Test Report

Run ID: 21342027602-py3.12 • Generated: 2026-01-26 00:14:25 • Duration: 120.64s
Plugin: v0.2.1 (a03dbe622cdc018f89b74731aed91adf1a582867) [dirty]
Repo: v0.2.1 (3145d53f452fcab2350c3fbbe2bf81eb0b563169) [dirty]
LLM: ollama / llama3.2:1b (minimal context, 620 annotated, 2 errors)
Token Usage: 135747 input, 75289 output (Total: 211036)

93.04%
Total Coverage

623	623	0	0
TOTAL TESTS	PASSED	FAILED	SKIPPED

0	0	0
XFAILED	XPASSED	ERRORS

Source Coverage Per Test Details Failures Only

Source Coverage

FILE	STMTS	MISS	COVER	%	COVERED LINES	MISSED LINES
src/pytest_llm_report/_git_info.py	2	0	2	100.0%	2-3	-
src/pytest_llm_report/aggregation.py	121	6	115	95.04%	13, 15-19, 21, 36, 39, 45, 47, 53-54, 56-58, 60, 62-65, 70, 74-75, 78-81, 85, 88-90, 94, 104, 110, 113-115, 117-121, 123-124, 129, 131-132, 134-135, 138-139, 145-147, 149, 152, 155, 158, 160, 162, 176, 178, 182, 184, 186, 196, 198-202, 204-205, 208, 210, 219, 231, 233-247, 249, 251, 259-260, 262-263, 265, 267-269, 273, 276-277,	67, 91-92, 111, 206, 217

					279-280, 283, 285-286, 288, 290-291, 295	
src/pytest_llm_report/cache.py	47	3	44	93.62%	13, 15-19, 21, 27, 33, 39-41, 43, 53, 55-56, 58, 60-62, 68-69, 78, 86, 88, 90, 92, 94, 97, 103, 107, 118-119, 121, 123, 129, 132-136, 141, 144, 153	64-65, 130
src/pytest_llm_report/collector.py	111	1	110	99.1%	19, 21-22, 24, 26-27, 33-34, 45-50, 52, 58, 60-62, 69, 78-79, 81, 90, 93-94, 96, 99-104, 106-107, 109-112, 114-119, 121-122, 124, 127-128, 130, 132-133, 135-137, 140-141, 143, 155, 163-164, 167-169, 171, 173, 181-182, 185-189, 191, 198-200, 202, 209-210, 212-214, 216, 218, 227-228, 230-236, 238, 241, 250-252, 254, 261, 264-265, 268-269, 271, 277, 279, 285	239
src/pytest_llm_report/context_util.py	53	3	50	94.34%	13-15, 18, 27, 29-31, 33, 35-36, 38-41, 47-49, 51-52, 55-59, 61-62, 64, 66-69, 72, 81-82, 86, 88-90, 93, 96, 108, 111, 124, 126-127, 129-130, 133, 135	53, 83-84

src/pytest_llm_report/coverage_map.py	135	6	129	95.56%	13, 15-17, 19-22, 30, 38, 44-45, 47, 58-60, 64, 72-73, 83, 86, 88-90, 92, 94-96, 98, 101-104, 106-108, 114, 116, 118, 121-122, 127-128, 131-135, 137-140, 144-146, 148, 150, 152-153, 156, 160-162, 165, 167-168, 173, 176, 178-184, 187-189, 191, 196, 199-200, 202, 204, 216-217, 220, 224-225, 228-234, 236, 239, 241, 243-244, 246-250, 252-254, 257, 259-260, 263-264, 271, 273-274, 276-279, 281-283, 285, 299-300, 302, 308	62, 123, 125, 157, 221, 251
src/pytest_llm_report/errors.py	36	0	36	100.0%	8-9, 12, 25-28, 31-36, 39-42, 45-46, 49-51, 54-55, 64-66, 68, 70, 73, 77-79, 83, 132, 142	-
src/pytest_llm_report/llm/__init__.py	3	0	3	100.0%	4-5, 7	-
src/pytest_llm_report/llm/annotator.py	154	21	133	86.36%	4, 6-10, 12-15, 21-22, 25-30, 33, 47-48, 50-52, 56, 58-59, 65, 67-68, 70, 73-74, 76, 84, 86-90, 95-96, 98-99, 106-107, 112-113, 116, 121-126, 130, 132, 134, 137, 144, 156, 181-182, 184, 186, 188-189, 199, 211, 213-216, 221-223, 226, 249-252, 254-255, 260, 262, 264-267, 269-270, 277-279, 281,	77-81, 160-168, 173, 286-287, 345, 364-365, 371

					283-284, 289-290, 292-293, 298-301, 303, 306, 329-332, 334, 336, 342, 344, 350-351, 353-354, 356-359, 361-362, 367-368, 370, 376-379, 381	
src/pytest_llm_report/llm/base.py	131	6	125	95.42%	13, 15-18, 20, 30, 33, 47, 50, 53, 59, 65-66, 68, 87-88, 96, 101, 103, 105, 128, 134-135, 137-138, 149, 155, 157, 163, 165, 174, 176, 185-186, 188, 191-198, 200, 202, 212, 214-217, 219-222, 224, 232, 243, 245, 247, 264, 266-267, 270-272, 274-275, 277, 279, 283, 286, 290-291, 294-295, 298-299, 305, 307-308, 310, 312, 314, 316, 325-326, 329-331, 333-334, 337-339, 342-347, 351, 353, 359-360, 363-364, 367-369, 372, 384, 386, 388-389, 391-392, 394, 396-397, 399, 401-402, 404, 406	91-92, 230, 284, 292, 296
src/pytest_llm_report/llm/batching.py	90	4	86	95.56%	8, 10-13, 20, 23-24, 27-29, 31-32, 34, 36-37, 39, 44, 53-55, 58, 67-68, 70, 73, 92-93, 95, 97, 103-106, 108-110, 112, 122-123, 126-128, 136, 139, 156-157, 160, 162, 164-167, 170-176, 181-185, 187-188, 190, 192-194,	158, 207, 211, 220

					196-197, 203-206, 209-210, 213-214, 216-218, 222, 224	
src/pytest_llm_report/llm/gemini.py	325	7	318	97.85%	7, 9-13, 15-16, 23-27, 30-34, 37-42, 44-46, 48-50, 52, 57-63, 65-70, 72-73, 75-78, 80-85, 87-89, 91-97, 99-114, 121-122, 125, 128, 134-135, 137-141, 143-144, 146, 164-166, 173-175, 178, 181-182, 184, 186-189, 191-192, 198-206, 208-210, 212-213, 215, 218, 221-230, 232-233, 235-237, 239-243, 246-247, 249-252, 254-255, 259, 261, 263, 268, 272-276, 279-281, 283, 288-293, 295, 299-305, 308-309, 311-312, 318-319, 322, 326, 332-333, 335, 339-343, 345-349, 352-353, 358-359, 366-367, 369, 383, 385-386, 390, 410, 413-415, 418-422, 424-427, 432, 434-435, 437, 441-444, 446, 449-463, 469, 471-473, 475-478, 480, 486, 488-491, 493, 495, 497-498, 502-508, 511, 514-516, 518-521, 523-528, 534, 537, 539-543, 547-548, 550-559, 562-564, 567-570, 574	115-117, 298, 310, 313-314

src/pytest_llm_report/llm/litellm_provider.py	77	1	76	98.7%	8, 10, 12-13, 21, 31, 37-38, 41-42, 44, 51, 60-62, 64, 82-83, 89, 92, 95-96, 98, 100-101, 104, 106-107, 112, 114, 116, 120, 122, 124-126, 129-130, 132, 135, 137, 139, 141-142, 144, 148, 170, 182-183, 186-188, 190, 192-193, 196-198, 204, 206, 211, 213, 215, 221-222, 224, 227-231, 234, 236, 242-243, 245	207
src/pytest_llm_report/llm/noop.py	13	0	13	100.0%	8, 10, 12-13, 20, 26, 32, 34, 51, 53, 59, 61, 67	-
src/pytest_llm_report/llm/ollama.py	72	1	71	98.61%	7, 9, 11-12, 18, 24, 42-43, 49, 52-53, 55, 58, 60-61, 63-67, 70, 74-77, 83, 85-86, 92, 94, 96-98, 100-101, 103, 107, 113-114, 116-118, 122, 128, 130, 138, 140, 142-144, 149-150, 156, 158, 160-162, 165-167, 172-173, 178, 180, 190, 192-193, 204, 209, 211-212	90
src/pytest_llm_report/llm/schemas.py	36	1	35	97.22%	8, 10-12, 16, 22, 38, 42-44, 46-47, 50-53, 55, 58-59, 62-65, 67-68, 77, 84, 90, 94-98, 102, 130	39

src/pytest_llm_report/llm/token_refresh.py	71	0	71	100.0%	7, 9-14, 17, 20, 23-24, 36-39, 41-43, 47, 59-60, 63-66, 69-72, 74, 83, 85-88, 90-91, 93, 101-103, 107-109, 111, 113-116, 120, 132-136, 139-140, 143-145, 148-150, 153-156, 158, 160-162	-
src/pytest_llm_report/llm/utils.py	33	2	31	93.94%	4, 6, 9, 20, 23, 42-43, 46-47, 51-53, 55-56, 66, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 90, 93-94, 96, 98	48, 78
src/pytest_llm_report/models.py	253	0	253	100.0%	17-18, 20, 23, 26-27, 36-38, 40, 42, 49-50, 59-61, 63, 65, 72-73, 86-92, 94, 96, 107-108, 120-126, 128, 130, 135-143, 146-147, 169-185, 187-188, 190, 192, 194, 201-224, 227-228, 236-237, 239, 241, 247-248, 257-259, 261, 263, 270-271, 280-282, 284, 286, 290-292, 295-296, 333-362, 364-372, 374, 376, 394-417, 419-437, 440-441, 455-463, 465, 467, 477-479, 482-483, 500-510, 512, 518, 520, 526-540	-

					122, 170, 199, 202-204, 209-211, 217-219, 225-227, 233-235, 241-242, 245-254, 257-259, 265-267, 271-274, 276, 284, 293, 308, 311-312, 320-325, 327, 332-337, 340-345, 348-349, 352-353, 356-357, 360-369, 372-375, 378-393, 396-397, 400-405, 408-409, 412-413, 416-421, 426-427, 430-431, 436-439, 444-447, 449, 451, 453, 460-461, 463-464, 466-467, 470-475, 479, 482-495, 498, 502-503, 507, 510, 514-515, 519-520, 524, 527, 531, 534-536, 540-541, 545-546, 550, 553, 557, 560, 564-565, 569, 572-574, 578, 581-584, 587, 591-592, 596, 599-608, 611, 613	
src/pytest_llm_report/options.py	268	57	211	78.73%		13-15, 21-22, 98-102, 105-107, 110-115, 118-121, 138-139, 142-149, 152-155, 158-160, 163-166, 169, 180-184, 187-188, 191, 193, 278, 287, 296
					41, 44, 50, 56, 62, 68, 74, 81, 90, 96, 102, 108, 114, 122, 128, 134, 142, 148, 155, 161, 169, 176, 185, 192, 199, 208, 215, 223, 229, 235, 241, 247, 254, 260, 268, 274, 283, 289, 297, 304, 311, 328, 332, 336, 342-343, 346-347, 349, 351, 354-356, 362-363, 371-372, 399-400, 403-404, 407, 410-411, 413-414, 417-418, 420, 422-426, 429-430, 432, 434, 437-	
src/pytest_llm_report/plugin.py	182	24	158	86.81%		13, 15-18, 20-21, 23, 29-32, 35, 319, 377, 481-482, 488, 548-549, 571, 595, 611-612

					438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-466, 468, 470-473, 476-477, 485-487, 491-494, 497, 499, 502-507, 509, 512-514, 516-521, 523, 534-535, 558-559, 562-563, 566-568, 579-580, 583, 586-587, 590-592, 602-603, 606-608, 619-620, 623, 626, 628-629	
src/pytest_llm_report/prompts.py	110	3	107	97.27%	13, 15-17, 24, 27, 33, 35, 49, 52, 55, 58-61, 63, 65, 67, 78-79, 82-84, 86-87, 92, 94-95, 98-101, 103-112, 114, 116, 118, 139-140, 142-144, 147, 152-153, 155-157, 159-161, 163-164, 166-167, 170-171, 173, 177, 180, 189, 192-194, 196-197, 201, 203, 216-217, 219-220, 223-228, 231-232, 235-237, 239-240, 242-247, 249, 251, 268, 275, 284-287	80, 185, 233
src/pytest_llm_report/render.py	65	6	59	90.77%	13, 15-16, 18, 24, 30-31, 34, 40, 42, 50-51, 53, 56, 65-67, 70, 79, 87, 90, 99, 101-102, 107, 110, 121-124, 126-129, 131-134, 140-142, 147, 155-157, 159, 172-177, 191, 210-211, 224, 267, 269, 285	148-149, 212, 217-218, 222

src/pytest_llm_report/report_writer.py	167	3	164	98.2%	13, 15-25, 27-29, 46, 55, 58, 67-68, 76, 83-84, 89, 98-100, 102, 105-108, 110, 113, 116, 127-128, 130, 142, 150, 156-158, 160, 186-189, 192, 197-199, 202-203, 211, 222-223, 226-227, 230-231, 233, 235, 254, 256-259, 262-264, 266, 268, 310, 319, 321-322, 324-335, 337, 339, 347, 350-352, 355-356, 359-361, 364, 367, 375, 383, 385-386, 389, 392, 395, 398, 406, 408-409, 415, 417, 419, 421-432, 439, 441-442, 444-446, 454-458, 460, 462, 465, 468-469, 471, 477-481, 487-488, 495, 502, 504, 506-508, 510, 513-514, 516, 522-523	135-137
src/pytest_llm_report/util/fs.py	34	1	33	97.06%	11, 13-14, 17, 30, 33, 36, 39, 42, 45, 55-56, 58-60, 63-65, 67, 70, 79, 82, 100, 103, 111-113, 116-117, 119-121, 123	40
src/pytest_llm_report/util/hashing.py	36	0	36	100.0%	12, 14-17, 23, 32, 35, 44-48, 51, 61, 64, 73-74, 76-78, 80-81, 86, 96, 103-104, 107, 113-114, 116-121	-

src/pytest_llm_report/util/ranges.py	33	0	33	100.0%	12, 15, 29-30, 33, 35-37, 39-40, 42, 45-47, 50, 52, 55, 65-67, 70, 81-82, 84-91, 93, 95	-
src/pytest_llm_report/util/time.py	16	0	16	100.0%	4, 6, 9, 15, 18, 27, 30, 39-44, 46-48	-

Per Test Details

📄 tests/test_adaptive_prompts.py

9 tests

PASSED `tests/test_adaptive_prompts.py::TestComplexityEstimation::test_complex_test_high_complexity` 1ms 🛡️ 5

AI ASSESSMENT

Scenario: tests/test_adaptive_prompts.py::TestComplexityEstimation::test_complex_test_high_complexity

Why Needed: This test is needed because it checks for complexity estimation in tests that have multiple assertions and mocks.

Key Assertions:

- {'name': 'assertion count', 'value': 5}
- {'name': 'mocks used', 'value': 3}

Confidence: 80%

Tokens: 118 input + 103 output = 221 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	17 lines (ranges: 65-66, 185, 188, 191-198, 200, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_adaptive_prompts.py::TestComplexityEstimation::test_empty_source_zero_complexity 1ms 5

AI ASSESSMENT

Scenario:
tests/test_adaptive_prompts.py::TestComplexityEstimation::test_empty_source_zero_complexity

Why Needed: The test is necessary because it checks the behavior of the `Config` class when given an empty source.

Key Assertions:

- {'description': "assert provider._estimate_test_complexity('') == 0", 'expected_value': 0, 'message': "Expected provider._estimate_test_complexity('') to return 0"}
- {'description': 'assert provider._estimate_test_complexity(None) == 0', 'expected_value': 0, 'message': 'Expected provider._estimate_test_complexity(None) to return 0'}

Confidence: 80%

Tokens: 136 input + 163 output = 299 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	8 lines (ranges: 65-66, 185-186, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_adaptive_prompts.py::TestComplexityEstimation::test_simple_test_low_complexity 2ms 5

AI ASSESSMENT

Scenario: tests/test_adaptive_prompts.py::TestComplexityEstimation::test_simple_test_low_complexity

Why Needed: This test is needed because it checks if Simple tests have low complexity scores.

Key Assertions:

- {'name': 'complexity_score', 'value': 'low'}

Confidence: 80%

Tokens: 115 input + 79 output = 194 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	17 lines (ranges: 65-66, 185, 188, 191-198, 200, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_adaptive_prompts.py::TestConfigValidation::test_invalid_prompt_tier` 1ms 3

AI ASSESSMENT

Scenario: Test invalid prompt tier

Why Needed: To test that the 'invalid' prompt tier is considered an error during validation.

Key Assertions:

- {'description': "The 'prompt_tier' field should be present in any error message.", 'expected_value': 'prompt_tier'}
- {'description': "Any error messages should contain the string 'prompt_tier'.", 'expected_value': 'prompt_tier'}

Confidence: 80%

Tokens: 126 input + 110 output = 236 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-261, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_adaptive_prompts.py::TestConfigValidation::test_valid_prompt_tiers` 1ms 3

AI ASSESSMENT

Scenario: Valid prompt tiers are validated

Why Needed: To ensure that the `prompt_tier` parameter is correctly validated and does not cause any issues.

Key Assertions:

- {'name': 'Expected errors for invalid prompt tiers', 'description': 'The validation should return an empty list of errors for valid prompt tiers'}

Confidence: 80%

Tokens: 142 input + 84 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	26 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_adaptive_prompts.py::TestPromptTierSelection::test_auto_tier_complex_test` 1ms 5

AI ASSESSMENT

Scenario:
tests/test_adaptive_prompts.py::TestPromptTierSelection::test_auto_tier_complex_test

Why Needed: Auto mode should use standard prompt for complex tests.

Key Assertions:

- {'name': 'standard_prompt_for_complex_tests', 'expected_value': 'standard prompt for complex tests'}

Confidence: 80%

Tokens: 122 input + 80 output = 202 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	23 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-220, 222, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_adaptive_prompts.py::TestPromptTierSelection::test_auto_tier_simple_test` 1ms 5

AI ASSESSMENT

Scenario:
tests/test_adaptive_prompts.py::TestPromptTierSelection::test_auto_tier_simple_test

Why Needed: To ensure that the auto mode selects a minimal prompt for simple tests, which is essential for efficient and effective test execution.

Key Assertions:

- {'name': 'selected_prompt_type', 'value': 'MINIMAL_SYSTEM_PROMPT'}

Confidence: 80%

Tokens: 155 input + 91 output = 246 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	23 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_adaptive_prompts.py::TestPromptTierSelection::test_minimal_tier_override` 1ms 5

AI ASSESSMENT

Scenario:
tests/test_adaptive_prompts.py::TestPromptTierSelection::test_minimal_tier_override

Why Needed: Config override to minimal should always use minimal prompt.

Key Assertions:

- {'name': 'provider', 'value': 'none'}
- {'name': 'prompt_tier', 'value': 'minimal'}

Confidence: 80%

Tokens: 122 input + 91 output = 213 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	9 lines (ranges: 65-66, 212, 214-215, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_adaptive_prompts.py::TestPromptTierSelection::test_standard_tier_override` 1ms 5

AI ASSESSMENT

Scenario: Config override to standard should always use standard prompt.

Why Needed: Because the config override is not necessary in this case, and using the standard prompt is a better practice.

Key Assertions:

- {'assertion_type': 'is', 'expected_value': 'standard_system_prompt'}

Confidence: 80%

Tokens: 148 input + 77 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	10 lines (ranges: 65-66, 212, 214, 216-217, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_aggregation.py

10 tests

PASSED tests/test_aggregation.py::TestAggregator::test_aggregate_all_policy 2ms 3

AI ASSESSMENT

Scenario: Test that the aggregate function correctly handles all policy when aggregating multiple test cases

Why Needed: This test prevents regression where an aggregation of multiple tests results in only one test being retained due to a missing or incomplete policy.

Key Assertions:

- The aggregated report should contain both test case information for each run.
- The number of tests in the aggregated report should be equal to the total number of runs.
- Each test case should have a corresponding run in the aggregated report.
- All retained test cases should be included in the aggregated report.
- Any missing or incomplete policy should not result in only one test being retained.
- The aggregate function should correctly handle all policy when aggregating multiple tests.
- The temporary directory created for the aggregation process should be deleted after the test is completed.

Confidence: 80%

Tokens: 364 input + 180 output = 544 total

COVERAGE

src/pytest_llm_report/aggregation.py	71 lines (ranges: 53, 56-57, 60, 62-64, 74-75, 78-81, 85, 88-90, 94-101, 110, 113-114, 117-121, 123, 129, 131-132, 134-135, 138, 145, 158, 160, 162-167, 169, 171-173, 184, 231, 233-237, 249, 259, 262-263, 265, 267, 290-293, 295)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_aggregation.py::TestAggregator::test_aggregate_dir_not_exists 4ms 3

AI ASSESSMENT

Scenario: tests/test_aggregation.py::TestAggregator::test_aggregate_dir_not_exists

Why Needed: The test is failing because the `aggregate` method of the Aggregator class does not check if the aggregation directory exists before aggregating data.

Key Assertions:

- {'name': 'Expected behavior', 'description': 'The `aggregate` method should return None when the aggregation directory does not exist.', 'expected_value': 'None'}

Confidence: 80%

Tokens: 104 input + 109 output = 213 total

COVERAGE

src/pytest_llm_report/aggregation.py	8 lines (ranges: 53, 56-58, 110, 113-115)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_aggregation.py::TestAggregator::test_aggregate_latest_policy` 4ms 3

AI ASSESSMENT

Scenario: Test that the `aggregate` method consistently picks the latest policy for a given test case across different times.

Why Needed: This test prevents regression where the latest policy is not picked correctly due to inconsistent timing of reports.

Key Assertions:

- The result of `aggregator.aggregate()` should contain only one test with an outcome of 'passed'.
- The number of tests in the result should be equal to 1.
- The outcome of the first test in the result should be 'passed'.
- The `run_meta` object should have a `is_aggregated` attribute set to True.
- The `run_meta.run_count` attribute should be equal to 2.
- The `summary.passed` attribute should be equal to 1 (i.e., the first test passed).
- The `summary.failed` attribute should be equal to 0 (i.e., no tests failed).

Confidence: 80%

Tokens: 477 input + 205 output = 682 total

COVERAGE

src/pytest_llm_report/aggregation.py	79 lines (ranges: 53, 56-57, 60, 65, 70, 74-75, 78-81, 85, 88-90, 94-101, 110, 113-114, 117-121, 123, 129, 131-132, 134-135, 138, 145, 158, 160, 162-167, 169, 171-173, 184, 196, 198-202, 204-205, 208, 231, 233-237, 249, 259, 262-263, 265, 267, 290-293, 295)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_aggregation.py::TestAggregator::test_aggregate_no_dir_configured` 1ms 3

AI ASSESSMENT

Scenario: tests/test_aggregation.py::TestAggregator::test_aggregate_no_dir_configured

Why Needed: The test is necessary because the aggregator requires a directory to aggregate data from.

Key Assertions:

- {'name': 'agg', 'type': 'NoneType', 'expected_type': 'NoneType'}

Confidence: 80%

Tokens: 110 input + 84 output = 194 total

COVERAGE

src/pytest_llm_report/aggregation.py	3 lines (ranges: 45, 53-54)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_aggregation.py::TestAggregator::test_aggregate_no_reports` 1ms 3

AI ASSESSMENT

Scenario: The `aggregate` method of the Aggregator class should not be called when there are no reports to aggregate.

Why Needed: This test prevents a potential bug where the `aggregate` method is called on an empty list of reports, causing it to return `None` instead of raising an error.

Key Assertions:

- The `aggregate` method should be called with an empty list of reports.
- The `aggregate` method should raise an error when given an empty list of reports.
- The `aggregate` method should not call the `is_file()` and `is_dir()` methods on empty lists of reports.

Confidence: 80%

Tokens: 201 input + 142 output = 343 total

COVERAGE

`src/pytest_llm_report/aggregation.py`	`10 lines (ranges: 53, 56-58, 110, 113-114, 117-118, 184)`
`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

PASSED `tests/test_aggregation.py::TestAggregator::test_aggregate_with_coverage_and_llm_annotations` 3ms 4

AI ASSESSMENT

Scenario: Test that coverage and LLM annotations are properly deserialized and can be re-serialized.

Why Needed: Prevents regression in core functionality by ensuring accurate token usage and coverage information is preserved during serialization.

Key Assertions:

- Coverage was correctly deserialized from the JSON report.
- LLM annotation was correctly deserialized with the expected scenario, why needed, and key assertions.
- Token usage was correctly serialized back into a JSON object.
- The test can be re-serialized to ensure it remains accurate.

Confidence: 80%

Tokens: 1002 input + 121 output = 1123 total

COVERAGE

src/pytest_llm_report/aggregation.py	87 lines (ranges: 53, 56-57, 60, 65, 70, 74-75, 78-81, 85, 88-90, 94-101, 110, 113-114, 117-121, 123, 129, 131-132, 134-135, 138-141, 145-147, 149-150, 152-153, 155, 158, 160, 162-167, 169, 171-173, 184, 196, 198-202, 208, 231, 233-237, 249, 259, 262-263, 265, 267, 290-293, 295)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	40 lines (ranges: 42-45, 65-68, 130-133, 135-137, 139, 141-143, 190, 194-199, 201, 203, 205, 207, 210-214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_aggregation.py::TestAggregator::test_aggregate_with_source_coverage` 2ms 3

AI ASSESSMENT

Scenario: Test source coverage summary deserialization when aggregate is configured.

Why Needed: This test prevents a potential bug where the source coverage summary is not correctly deserialized if the aggregate configuration does not match the report format.

Key Assertions:

- The `source_coverage` key exists in the `report1` dictionary and has the expected structure.
- The `file_path` value of the first `SourceCoverageEntry` is 'src/foo.py'.
- The `coverage_percent` value is 83.33%.
- The `covered_ranges` value is '1-5, 7-11'.
- The `missed_ranges` value is '6, 12'.

Confidence: 80%

Tokens: 395 input + 158 output = 553 total

COVERAGE

src/pytest_llm_report/aggregation.py	67 lines (ranges: 53, 56-57, 60, 65, 70, 74-75, 78-81, 85, 88-90, 94-101, 110, 113-114, 117-121, 123, 129, 131-132, 162-169, 171-173, 184, 196, 198-200, 208, 231, 233-234, 249, 259, 262-263, 265, 267, 290-293, 295)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_aggregation.py::TestAggregator::test_load_coverage_from_source` 3ms 3

AI ASSESSMENT

Scenario: Test loading coverage from configured source file when option is not set.

Why Needed: To prevent a potential bug where the test fails due to an incorrect assumption about the configuration.

Key Assertions:

- Verify that aggregator._load_coverage_from_source() returns None when llm_coverage_source is not set.
- Verify that aggregator._load_coverage_from_source() raises a UserWarning when llm_coverage_source does not exist.
- Verify that aggregator._load_coverage_from_source() correctly loads coverage from the configured source file (mocking coverage.py).
- Verify that mock_cov.report() returns 80.0 as expected after calling cov.report().
- Verify that mock_mapper.map_source_coverage() correctly maps the coverage data to entries.
- Verify that aggregator._load_coverage_from_source() correctly returns a tuple containing one entry when successful loading is achieved.

Confidence: 80%

Tokens: 584 input + 189 output = 773 total

COVERAGE

src/pytest_llm_report/aggregation.py	19 lines (ranges: 259-260, 262-263, 265, 267-271, 273, 276-277, 279-280, 283, 285-286, 288)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_aggregation.py::TestAggregator::test_recalculate_summary 1ms 3

AI ASSESSMENT

Scenario: test_recalculate_summary verifies that the aggregator recalculates the latest summary correctly and preserves coverage percentage.

Why Needed: This test prevents regression in the aggregation logic, ensuring that the latest summary is calculated accurately and coverage percentage remains preserved.

Key Assertions:

- The total count of tests passed should be equal to the original count.
- The number of failed tests should remain unchanged.
- The number of skipped tests should remain unchanged.
- The number of tests with unknown status (xfailed, xpassed) should remain unchanged.
- The error rate should be within a reasonable range (e.g., 0-100%).
- The coverage percentage should not decrease below the original value.
- The total duration of all tests should increase by at least 5 seconds to reflect the recalculated summary.
- The latest summary should have the same node IDs as the original test results.

Confidence: 80%

Tokens: 473 input + 198 output = 671 total

COVERAGE

src/pytest_llm_report/aggregation.py	17 lines (ranges: 231, 233-247, 249)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_aggregation.py::TestAggregator::test_skips_invalid_json` 3ms 3

AI ASSESSMENT

Scenario: Test that skipping an invalid JSON report prevents the aggregation from counting it as a valid report.

Why Needed: This test ensures that the `aggregate` function correctly handles reports with missing or malformed data, preventing it from incorrectly counting them as valid.

Key Assertions:

- The `aggregate` function should not count an invalid JSON report as a valid one.
- The `aggregate` function should only count the valid report (in this case, 'run1') in its run meta.
- The test should raise a warning when trying to aggregate a report with missing fields ('missing_fields.json').
- The test should assert that the aggregation result is not None and has exactly one run meta entry.
- The test should only count the valid report (in this case, 'run1') in its run meta.

Confidence: 80%

Tokens: 352 input + 182 output = 534 total

COVERAGE

src/pytest_llm_report/aggregation.py	72 lines (ranges: 53, 56-57, 60, 65, 70, 74-75, 78-81, 85, 88-90, 94-101, 110, 113-114, 117-121, 123-124, 129, 131-132, 162-167, 169, 171-173, 176, 178-180, 182, 184, 196, 198-200, 208, 231, 233-234, 249, 259, 262-263, 265, 267, 290-293, 295)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_aggregation_maximal.py

1 tests

PASSED `tests/test_aggregation_maximal.py::TestAggregationMaximal::test_recalculate_summary_coverage` 1ms 4

AI ASSESSMENT

Scenario: The test verifies that the aggregator recalculates the summary correctly when new tests are added and the latest summary is updated.

Why Needed: This test prevents regression in coverage calculation when new tests are added to the aggregation process, ensuring accuracy of the overall coverage percentage.

Key Assertions:

- summary.total == 2
- summary.passed == 1
- summary.failed == 1
- summary.coverage_total_percent == 88.5
- summary.total_duration == 3.0

Confidence: 80%

Tokens: 299 input + 125 output = 424 total

COVERAGE

src/pytest_llm_report/aggregation.py	10 lines (ranges: 45, 231, 233-239, 249)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_annotator.py

13 tests

PASSED `tests/test_annotator.py::TestAnnotateTests::test_batch_optimization_message` 2ms 5

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_batch_optimization_message

Why Needed: To test the batch optimization message functionality.

Key Assertions:

- {'name': 'mock_assembler.send_message', 'expected_output': 'optimized message'}
- {'name': 'mock_provider.get_messages', 'expected_output': ['optimized message']}

Confidence: 80%

Tokens: 112 input + 97 output = 209 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	98 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-91, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188, 199-206, 213-219, 221, 223, 249-252, 254-255, 257-258, 260, 262, 264, 269-274, 277-279, 281, 283-284, 289-290, 292-295, 298, 303)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_annotator.py::TestAnnotateTests::test_cached_progress_reporting` 1ms 6

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_cached_progress_reporting

Why Needed: To ensure that the progress reporting is cached correctly and not overwritten by subsequent requests.

Key Assertions:

- {'name': 'Mocked cache should be updated with correct data', 'expected_value': {'progress': 50, 'total': 100}, 'actual_value': {'progress': 0, 'total': 0}}
- {'name': 'Mocked cache should not overwrite previous progress reports', 'expected_value': {'progress': 50, 'total': 100}, 'actual_value': {'progress': 25, 'total': 75}}

Confidence: 80%

Tokens: 101 input + 163 output = 264 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	50 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-89, 95-96, 98-99, 106-108, 112-113, 116, 121-128, 130, 134, 156, 181-182, 184, 211, 213-219, 221, 223)
src/pytest_llm_report/llm/batching.py	18 lines (ranges: 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_annotator.py::TestAnnotateTests::test_cached_tests_are_skipped` 2ms 6

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_cached_tests_are_skipped

Why Needed: The test is necessary because it checks if cached tests are skipped. This is a critical functionality for ensuring the reliability and efficiency of the annotator.

Key Assertions:

- {'assertion_type': 'mocking', 'expected_mock_calls': [], 'actual_mock_calls': [], 'reason': "The test is testing if the mock objects are being called correctly, which is essential for verifying the correctness of the annotator's behavior."}

Confidence: 80%

Tokens: 102 input + 134 output = 236 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	95 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-87, 95-96, 98-99, 106-108, 112-113, 116, 121-124, 130, 132, 134, 137-141, 144-151, 156, 181-182, 184, 186, 188, 199-206, 213-219, 221, 223, 249-252, 254-255, 257-258, 260, 262, 264, 269-274, 277-279, 281, 283-284, 289-290, 292, 298, 303)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_annotator.py::TestAnnotateTests::test_concurrent_annotation 3ms 6

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_concurrent_annotation

Why Needed: To ensure that annotators can annotate data in a concurrent manner without causing any issues.

Key Assertions:

- {'assertion_type': 'concurrency', 'expected_result': 'No exceptions should be raised when multiple annotators are concurrently annotating the same dataset.'}
- {'assertion_type': 'threading', 'expected_result': 'All annotators should finish annotating their assigned tasks within a reasonable time frame (e.g., 10 seconds).'}

Confidence: 80%

Tokens: 98 input + 139 output = 237 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	90 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-87, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188-196, 213-219, 221, 223, 329-332, 334, 336-340, 342, 344, 350-351, 353-354, 356-359, 361-362, 367-368, 370, 376, 381)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_annotator.py::TestAnnotateTests::test_concurrent_annotation_handles_failures` 2ms 6

AI ASSESSMENT

Scenario:
tests/test_annotator.py::TestAnnotateTests::test_concurrent_annotation_handles_failures

Why Needed: This test is needed because the current implementation of `annotate` may not handle failures correctly when multiple annotators are working concurrently.

Key Assertions:

- {'name': 'Mock provider returns a valid annotation', 'expected_value': 'Annotation returned by mock provider'}
- {'name': 'Mock cache does not store annotations', 'expected_value': 'Annotations are not stored in the mock cache'}

Confidence: 80%

Tokens: 116 input + 128 output = 244 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	94 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-87, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188-196, 213-219, 221-223, 329-332, 334, 336-340, 342, 344, 350-351, 353-354, 356-359, 361-362, 367-368, 370, 376-379, 381)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_annotator.py::TestAnnotateTests::test_progress_reporting 3ms 6

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_progress_reporting

Why Needed: To ensure that the annotator reports progress accurately and consistently throughout the annotation process.

Key Assertions:

- {'name': 'mock_provider should be called with a valid task', 'expected_value': 'mock_provider.task', 'actual_value': 'mock_provider.get_task()'}
- {'name': 'mock_cache should not be modified during the annotation process', 'expected_value': 'None', 'actual_value': 'mock_cache'}

Confidence: 80%

Tokens: 96 input + 134 output = 230 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	96 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-89, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188, 199-206, 213-219, 221, 223, 249-252, 254-255, 257-258, 260, 262, 264, 269-274, 277-279, 281, 283-284, 289-290, 292-295, 298, 303)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_annotator.py::TestAnnotateTests::test_reports_progress_messages 2ms 6

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_reports_progress_messages

Why Needed: To ensure that the progress messages are returned correctly when annotating reports.

Key Assertions:

- {'name': 'mock_provider', 'expected_type': 'MockProvider', 'actual_type': 'MockProvider'}
- {'name': 'mock_cache', 'expected_type': 'MockCache', 'actual_type': 'MockCache'}
- {'name': 'mock_assembler', 'expected_type': 'MockAssembler', 'actual_type': 'MockAssembler'}

Confidence: 80%

Tokens: 101 input + 146 output = 247 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	96 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-89, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188, 199-206, 213-219, 221, 223, 249-252, 254-255, 257-258, 260, 262, 264, 269-274, 277-279, 281, 283-284, 289-290, 292-295, 298, 303)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_annotator.py::TestAnnotateTests::test_respects_opt_out_and_limit` 2ms 6

AI ASSESSMENT

Scenario: tests/test_annotator.py

Why Needed: The test is necessary to ensure that the annotator respects the opt-out and limit settings.

Key Assertions:

- {'name': 'mock_provider', 'expected_result': {'opt_out': [1, 2], 'limit': 5}, 'actual_result': {'opt_out': [1, 3], 'limit': 10}}

Confidence: 80%

Tokens: 104 input + 102 output = 206 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	91 lines (ranges: 47, 50-51, 58-59, 65, 67-68, 73-74, 76, 84, 86-87, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188, 199-206, 213-219, 221, 223, 249-252, 254-255, 257-258, 260, 262, 264, 269-274, 277-279, 281, 283-284, 289-290, 292, 298, 303)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_annotator.py::TestAnnotateTests::test_respects_rate_limit 2ms 6

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_respects_rate_limit

Why Needed: The test respects the rate limit for annotating a single document.

Key Assertions:

- {'name': 'mock_provider', 'expected_value': 1, 'actual_value': 2}
- {'name': 'mock_cache', 'expected_value': 1000, 'actual_value': 2000}

Confidence: 80%

Tokens: 112 input + 113 output = 225 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	94 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-87, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188, 199-206, 213-219, 221, 223, 249-252, 254-257, 260, 262, 264-267, 269-274, 277-279, 281, 283-284, 289-290, 292, 298, 303)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_annotator.py::TestAnnotateTests::test_sequential_annotation` 12.00s 6

AI ASSESSMENT

Scenario: Sequential annotation

Why Needed: To ensure that the annotator can correctly annotate sequential data.

Key Assertions:

- {'name': 'annotator behaves as expected', 'description': 'The annotator should be able to correctly identify and annotate sequential elements in the input data.'}
- {'name': 'sequential annotations are preserved', 'description': 'The sequential annotations should be preserved across multiple passes of the annotator, even if the input data is reordered or split into smaller chunks.'}

Confidence: 80%

Tokens: 98 input + 122 output = 220 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	94 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-87, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188, 199-206, 213-219, 221, 223, 249-252, 254-255, 257-258, 260, 262, 264-267, 269-274, 277-279, 281, 283-284, 289-290, 292, 298, 303)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_annotator.py::TestAnnotateTests::test_sequential_annotation_error_tracking` 24.00s 6

AI ASSESSMENT

Scenario:
tests/test_annotator.py::TestAnnotateTests::test_sequential_annotation_error_tracking

Why Needed: Error tracking in sequential annotation is necessary to ensure that errors are properly reported and handled by the annotator.

Key Assertions:

- {'name': 'Mock provider should be called with error message', 'expected_output': 'Mock provider was called with an error message', 'actual_output': 'Mock provider was not called with an error message'}
- {'name': 'Mock cache should be cleared after error is reported', 'expected_output': 'Mock cache should be cleared after error is reported', 'actual_output': 'Mock cache remains unchanged'}

Confidence: 80%

Tokens: 105 input + 159 output = 264 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	98 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-87, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188, 199-206, 213-219, 221-223, 249-252, 254-255, 257-258, 260, 262, 264-267, 269-274, 277-279, 281, 283-284, 289-290, 292, 298-301, 303)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_annotator.py::TestAnnotateTests::test_skips_if_disabled 1ms 4

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_skips_if_disabled

Why Needed: The test should be skipped when the LLM (Large Language Model) is not enabled.

Key Assertions:

- {'name': 'config', 'value': "Config(provider='none')"}

Confidence: 80%

Tokens: 108 input + 81 output = 189 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	2 lines (ranges: 47-48)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_annotator.py::TestAnnotateTests::test_skips_if_provider_unavailable` 1ms 4

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_skips_if_provider_unavailable

Why Needed: The test is skipped if the provider is unavailable because it prevents the test from running and potentially causing data loss.

Key Assertions:

- {'name': 'Provider is not available', 'expected': 'Provider is not available'}

Confidence: 80%

Tokens: 101 input + 90 output = 191 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	7 lines (ranges: 47, 50-54, 56)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_base_coverage_v2.py

2 tests

PASSED `tests/test_base_coverage_v2.py::test_base_parse_response_malformed_json_after_extract` 1ms 5

AI ASSESSMENT

Scenario: Test Base Parse Response Malformed JSON After Extract

Why Needed: To ensure that the `_parse_response` method correctly handles malformed JSON responses and raises a `JSONDecodeError` with a meaningful error message.

Key Assertions:

- {'assertion_type': 'Expected error message', 'expected_value': 'Failed to parse LLM response as JSON'}
- {'assertion_type': 'Expected error code', 'expected_value': 2}

Confidence: 80%

Tokens: 152 input + 114 output = 266 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	10 lines (ranges: 65-66, 325-326, 329-330, 333-334, 359-360)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_base_coverage_v2.py::test_base_parse_response_non_string_fields` 1ms 5

AI ASSESSMENT

Scenario: Verify that the `test_base_parse_response_non_string_fields` test case checks for non-string fields in the response data.

Why Needed: This test prevents a potential bug where the function does not handle cases with non-string fields in its response data.

Key Assertions:

- assert annotation.scenario == '123'
- assert annotation.why_needed == ['list']
- assert annotation.key_assertions == ['a']

Confidence: 80%

Tokens: 269 input + 103 output = 372 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	22 lines (ranges: 65-66, 325-326, 329-330, 333-334, 337-339, 342-346, 351, 353-357)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

tests/test_base_maximal.py

9 tests

PASSED `tests/test_base_maximal.py::TestGetProvider::test_get_gemini_provider` 1ms 5

AI ASSESSMENT

Scenario: tests/test_base_maximal.py::TestGetProvider::test_get_gemini_provider

Why Needed: To ensure the `get_gemini_provider` function is correctly creating a `GeminiProvider` instance.

Key Assertions:

- {'name': 'provider type', 'expected_type': 'GeminiProvider'}

Confidence: 80%

Tokens: 104 input + 83 output = 187 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	10 lines (ranges: 65-66, 384, 386, 388, 391, 396, 401-402, 404)
src/pytest_llm_report/llm/gemini.py	9 lines (ranges: 134-135, 137-141, 143-144)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_base_maximal.py::TestGetProvider::test_get_invalid_provider 2ms 4

AI ASSESSMENT

Scenario: tests/test_base_maximal.py::TestGetProvider::test_get_invalid_provider

Why Needed: To ensure that a ValueError is raised when an unknown LLM provider is specified.

Key Assertions:

- {'assertion_type': 'raises', 'message': 'Unknown LLM provider: invalid'}

Confidence: 80%

Tokens: 106 input + 80 output = 186 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	7 lines (ranges: 384, 386, 388, 391, 396, 401, 406)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_base_maximal.py::TestGetProvider::test_get_litellm_provider 1ms 5

AI ASSESSMENT

Scenario: tests/test_base_maximal.py::TestGetProvider::test_get_litellm_provider

Why Needed: To ensure the `get_litellm_provider` function returns a valid instance of `LiteLLMProvider`.

Key Assertions:

- {'name': 'provider type', 'expected': 'LiteLLMProvider'}

Confidence: 80%

Tokens: 109 input + 86 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	9 lines (ranges: 65-66, 384, 386, 388, 391, 396-397, 399)
src/pytest_llm_report/llm/litellm_provider.py	3 lines (ranges: 37-38, 41)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_base_maximal.py::TestGetProvider::test_get_noop_provider 1ms 5

AI ASSESSMENT

Scenario: tests/test_base_maximal.py::TestGetProvider::test_get_noop_provider

Why Needed: The test is necessary to ensure that the `get_provider` function returns a NoopProvider instance when no provider is specified.

Key Assertions:

- {'name': 'provider type', 'expected_value': 'NoopProvider'}

Confidence: 80%

Tokens: 104 input + 87 output = 191 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	6 lines (ranges: 65-66, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_base_maximal.py::TestGetProvider::test_get_ollama_provider 1ms 4

AI ASSESSMENT

Scenario: tests/test_base_maximal.py::TestGetProvider::test_get_ollama_provider

Why Needed: To verify the correctness of the OllamaProvider class.

Key Assertions:

- {'name': 'provider_type', 'expected': 'OllamaProvider', 'actual': 'isinstance(provider, OllamaProvider)'}

Confidence: 80%

Tokens: 108 input + 89 output = 197 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	8 lines (ranges: 65-66, 384, 386, 388, 391-392, 394)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_base_maximal.py::TestLlmProviderDefaults::test_available_caches_result` 1ms 4

AI ASSESSMENT

Scenario: Verify that the LlmProvider can be checked for availability without raising an exception.

Why Needed: This test prevents a potential bug where the LlmProvider raises an exception when checking its availability, causing the test to fail.

Key Assertions:

- The provider is not raised an exception when checking its availability.
- The provider's checks counter is incremented correctly.
- The provider returns True for availability checks.

Confidence: 80%

Tokens: 280 input + 99 output = 379 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	6 lines (ranges: 65-66, 134-135, 137-138)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_base_maximal.py::TestLlmProviderDefaults::test_get_model_name_defaults_to_config` 1ms 4

AI ASSESSMENT

Scenario: tests/test_base_maximal.py::TestLlmProviderDefaults

Why Needed: To ensure that the LLM model name is set to the default configuration when no explicit model name is provided.

Key Assertions:

- {'name': "provider.get_model_name() == 'test-model'", 'expected_value': 'test-model'}

Confidence: 80%

Tokens: 114 input + 86 output = 200 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	3 lines (ranges: 65-66, 163)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_base_maximal.py::TestLlmProviderDefaults::test_get_rate_limits_defaults_to_none` 1ms 4

AI ASSESSMENT

Scenario: tests/test_base_maximal.py

Why Needed: This test ensures that the `get_rate_limits` method returns `None` when no default rate limits are specified.

Key Assertions:

- {'name': 'assert get_rate_limits is None', 'description': 'The `get_rate_limits` method should return `None`.'}

Confidence: 80%

Tokens: 108 input + 87 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	3 lines (ranges: 65-66, 155)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_base_maximal.py::TestLlmProviderDefaults::test_is_local_defaults_to_false` 1ms 4

AI ASSESSMENT

Scenario: tests/test_base_maximal.py

Why Needed: To ensure the LLM provider defaults to false when in local mode.

Key Assertions:

- {'name': 'provider.is_local() should return False', 'expected_value': False}

Confidence: 80%

Tokens: 105 input + 68 output = 173 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	3 lines (ranges: 65-66, 174)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_batching.py

17 tests

PASSED `tests/test_batching.py::TestBuildBatchPrompt::test_context_files_included` 1ms 4

AI ASSESSMENT

Scenario: Verify that context files are included in the batch prompt.

Why Needed: This test prevents a potential issue where context files are not added to the prompt, potentially leading to incorrect usage of the `build_batch_prompt` function.

Key Assertions:

- The 'src/module.py' file is present in the prompt.
- The 'def helper()' function is present in the prompt.
- Context files should be added to the prompt according to their locations.

Confidence: 80%

Tokens: 261 input + 108 output = 369 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	35 lines (ranges: 34, 39, 156-157, 160, 162, 181-185, 187-188, 190, 192-194, 196-200, 203-206, 209-210, 213-214, 216-218, 222, 224)
src/pytest_llm_report/llm/utils.py	28 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestBuildBatchPrompt::test_parametrized_batch_prompt` 1ms 3

AI ASSESSMENT

Scenario: Verifies that the parametrized batch prompt includes all required information.

Why Needed: This test prevents a potential regression where the prompt is missing or incorrect, making it difficult to understand and use the `build_batch_prompt` function.

Key Assertions:

- Test Group: test.py::test_add[*]
- Parameterizations (2 variants)
- [1+1=2]
- [0+0=0]
- ONE annotation

Confidence: 80%

Tokens: 330 input + 107 output = 437 total

COVERAGE

`src/pytest_llm_report/collector.py`	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
`src/pytest_llm_report/llm/batching.py`	24 lines (ranges: 34, 39-40, 156-157, 160, 162, 164-168, 170-177, 187-188, 190, 224)
`src/pytest_llm_report/plugin.py`	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestBuildBatchPrompt::test_single_test_prompt` 1ms 3

AI ASSESSMENT

Scenario: Single test should generate normal prompt.

Why Needed: This test prevents a regression where the batched prompt is missing 'Test: test.py::test_foo' and incorrectly includes 'Parameterizations'.

Key Assertions:

- assert "Test: test.py::test_foo" in prompt
- assert "```python" in prompt
- assert source in prompt
- assert 'Parameterizations' not in prompt

Confidence: 80%

Tokens: 269 input + 108 output = 377 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	15 lines (ranges: 34, 39, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_batching.py::TestComputeSourceHash::test_consistent_hash 1ms 3

AI ASSESSMENT

Scenario: Verify that the same source code produces the same hash value.

Why Needed: Prevents a potential bug where different versions of the test function produce different hashes, leading to inconsistent results.

Key Assertions:

- The `source` variable is set to a string containing the test function definition.
- Two calls to `_compute_source_hash(source)` return the same hash value.
- The length of the returned hash value is 32 bytes as expected.
- _compute_source_hash(source) should not modify or alter the input source code.
- The `source` variable should be a string containing the test function definition.
- The hash value calculated by `_compute_source_hash(source)` should match the original hash value.
- If the same source code is used multiple times, the returned hash values should remain consistent.

Confidence: 80%

Tokens: 220 input + 181 output = 401 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	2 lines (ranges: 67, 70)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestComputeSourceHash::test_different_source_different_hash` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_batching.py::TestComputeSourceHash::test_different_source_different_hash

Why Needed: To ensure that different source code produces different hashes, which is a requirement for batch processing to work correctly.

Key Assertions:

- {'assertion_type': 'different', 'condition': 'hash1 != hash2'}

Confidence: 80%

Tokens: 127 input + 85 output = 212 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	2 lines (ranges: 67, 70)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_batching.py::TestComputeSourceHash::test_empty_source 1ms 3

AI ASSESSMENT

Scenario: tests/test_batching.py::TestComputeSourceHash::test_empty_source

Why Needed: The current implementation of ComputeSourceHash does not handle an empty source correctly.

Key Assertions:

- {'name': 'assert _compute_source_hash() returns an empty string for an empty input', 'expected_result': '', 'actual_result': '_compute_source_hash()'}

Confidence: 80%

Tokens: 94 input + 93 output = 187 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	2 lines (ranges: 67-68)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_batching.py::TestConfigValidation::test_batch_max_tests_minimum 1ms 3

AI ASSESSMENT

Scenario: tests/test_batching.py::TestConfigValidation::test_batch_max_tests_minimum

Why Needed: The test is necessary because the `batch_max_tests` configuration option must be at least 1.

Key Assertions:

- {'description': "The error message should contain 'batch_max_tests' in order to trigger this assertion.", 'expected_value': 'batch_max_tests', 'actual_value': '0'}

Confidence: 80%

Tokens: 126 input + 103 output = 229 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	27 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271-273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestConfigValidation::test_context_line_padding_non_negative` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_batching.py::TestConfigValidation::test_context_line_padding_non_negative

Why Needed: Context line padding must be non-negative.

Key Assertions:

- {'message': 'context_line_padding must be >= 0', 'description': 'The context line padding value is negative.'}

Confidence: 80%

Tokens: 126 input + 79 output = 205 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	27 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273-274, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestConfigValidation::test_invalid_context_compression` 1ms 3

AI ASSESSMENT

Scenario: Test invalid context compression

Why Needed: To test that an invalid context compression mode raises an error during validation.

Key Assertions:

- {'assertion_type': 'contains', 'value': 'context_compression', 'expected_value': 'invalid'}

Confidence: 80%

Tokens: 122 input + 72 output = 194 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-269, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestConfigValidation::test_valid_context_compression` 1ms 3

AI ASSESSMENT

Scenario: TestConfigValidation

Why Needed: Valid compression modes should pass.

Key Assertions:

- {'message': 'Context compression is enabled.', 'expected': 'None'}
- {'message': 'Context compression is disabled.', 'expected': 'lines'}

Confidence: 80%

Tokens: 133 input + 75 output = 208 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	26 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_batching.py::TestGetBaseNodeid::test_nested_params 1ms 3

AI ASSESSMENT

Scenario: tests/test_batching.py::TestGetBaseNodeid::test_nested_params

Why Needed: This test is necessary because the current implementation of _get_base_nodeid does not fully strip nested parameters.

Key Assertions:

- {'name': 'assert stripped params are correct', 'expected': 'test.py::test[a-b-c]', 'actual': '_get_base_nodeid('}

Confidence: 80%

Tokens: 109 input + 99 output = 208 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	2 lines (ranges: 53-54)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestGetBaseNodeid::test_parametrized_nodeid` 1ms 3

AI ASSESSMENT

Scenario: tests/test_batching.py::TestGetBaseNodeid::test_parametrized_nodeid

Why Needed: The test is necessary because it checks the behavior of `_get_base_nodeid` when a parameterized node id is passed.

Key Assertions:

- {'assertion': "_get_base_nodeid('tests/test_foo.py::test_add[1+1=2]'') == 'tests/test_foo.py::test_add'", 'expected_result': 'tests/test_foo.py::test_add'}

Confidence: 80%

Tokens: 133 input + 124 output = 257 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	2 lines (ranges: 53-54)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestGetBaseNodeid::test_simple_nodeid` 1ms 3

AI ASSESSMENT

Scenario: tests/test_batching.py::TestGetBaseNodeid::test_simple_nodeid

Why Needed: This test is needed because it checks the behavior of _get_base_nodeid when given a simple nodeid without any parameters.

Key Assertions:

- {'name': 'expected result', 'value': 'tests/test_foo.py::test_bar'}

Confidence: 80%

Tokens: 123 input + 91 output = 214 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	2 lines (ranges: 53, 55)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestGroupTestsForBatching::test_batch_max_size_respected` 1ms 4

AI ASSESSMENT

Scenario: Large groups should be split by batch_max_tests.

Why Needed: This test prevents regression where large groups are not split into batches of the correct size, potentially leading to performance issues or incorrect results.

Key Assertions:

- The number of tests in each batch is equal to `batch_max_tests`.
- Each batch has exactly two tests.
- Only one batch contains three tests.
- All tests are present in their respective batches.

Confidence: 80%

Tokens: 364 input + 106 output = 470 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	24 lines (ranges: 53-54, 67-68, 92-93, 95, 103-106, 108-110, 122-123, 126-132, 136)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestGroupTestsForBatching::test_batching_disabled` 1ms 4

AI ASSESSMENT

Scenario: Test case for batched tests

Why Needed: To ensure that each test is separate and not affected by the batching mechanism.

Key Assertions:

- {'name': 'Number of batches should be equal to number of tests', 'expected_value': 2, 'actual_value': 1}

Confidence: 80%

Tokens: 170 input + 81 output = 251 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	6 lines (ranges: 92-93, 95, 97-99)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestGroupTestsForBatching::test_parametrized_tests_grouped` 1ms 4

AI ASSESSMENT

Scenario: Test parametrized tests should be grouped together.

Why Needed: This test prevents a potential regression where the grouping of tests for batching is not considered.

Key Assertions:

- batches should contain exactly one group with parameterized tests.
- each group in batches should have exactly three tests.
- the first group in batches should be marked as parametrized and have its base nodeid as 'test.py::test_add'.
- all tests within a group should be from the same module (in this case, 'test.py').

Confidence: 80%

Tokens: 346 input + 129 output = 475 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	27 lines (ranges: 34, 39-40, 53-54, 67, 70, 92-93, 95, 103-106, 108-110, 122-123, 126-132, 136)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_batching.py::TestGroupTestsForBatching::test_single_tests_no_grouping` 1ms 4

AI ASSESSMENT

Scenario: Single tests should each be their own batch.

Why Needed: This test prevents a potential bug where multiple tests are grouped together and potentially interfere with each other's execution.

Key Assertions:

- Each test is in its own batch.
- There are two batches, one for each test.
- Each batch has exactly one test.
- The first batch contains only one test.
- The second batch also contains only one test.
- No tests are grouped together and interfere with each other's execution.

Confidence: 80%

Tokens: 278 input + 121 output = 399 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	18 lines (ranges: 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_cache.py

7 tests

PASSED `tests/test_cache.py::TestHashSource::test_consistent_hash` 1ms 3

AI ASSESSMENT

Scenario: tests/test_cache.py::TestHashSource::test_consistent_hash

Why Needed: To ensure that the cache is consistent across different runs of the test.

Key Assertions:

- {'name': 'Same source should produce same hash', 'expected_result': 'True'}

Confidence: 80%

Tokens: 107 input + 75 output = 182 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_cache.py::TestHashSource::test_different_source_different_hash` 1ms 3

AI ASSESSMENT

Scenario: Tests for cache functionality

Why Needed: To ensure that the cache is working correctly and producing unique hashes for different inputs.

Key Assertions:

- {'name': 'Hash of input 1 should not match hash of input 2', 'expected_result': 'different'}
- {'name': 'Hash of input 3 should match hash of input 4', 'expected_result': 'same'}

Confidence: 80%

Tokens: 108 input + 105 output = 213 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_cache.py::TestHashSource::test_hash_length 1ms 3

AI ASSESSMENT

Scenario: tests/test_cache.py::TestHashSource::test_hash_length

Why Needed: To ensure the hash value is of a fixed length (16 characters).

Key Assertions:

- {'description': 'The hash value should be 16 characters long.', 'value': 16}

Confidence: 80%

Tokens: 100 input + 77 output = 177 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_cache.py::TestLlmCache::test_clear 1ms 4

AI ASSESSMENT

Scenario: Test that clearing the cache removes all entries.

Why Needed: Prevents regression in case of large number of cache entries.

Key Assertions:

- The clear method should remove all cache entries.
- The get method should return None for non-existent keys.
- The count property should be updated correctly after clearing the cache.

Confidence: 80%

Tokens: 283 input + 82 output = 365 total

COVERAGE

src/pytest_llm_report/cache.py	26 lines (ranges: 39-41, 53, 55-56, 86, 90, 92, 94, 97-101, 103, 118-119, 121, 129, 132-136, 141)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_cache.py::TestLlmCache::test_does_not_cache_errors 1ms 4

AI ASSESSMENT

Scenario: tests/test_cache.py::TestLlmCache::test_does_not_cache_errors

Why Needed: To ensure that LLM annotations with errors are not cached.

Key Assertions:

- {'name': 'cache annotation for error', 'expected_value': 'None'}

Confidence: 80%

Tokens: 157 input + 73 output = 230 total

COVERAGE

src/pytest_llm_report/cache.py	11 lines (ranges: 39-41, 53, 55-56, 86, 88, 118-119, 121)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_cache.py::TestLlmCache::test_get_missing 1ms 4

AI ASSESSMENT

Scenario: tests/test_cache.py::TestLlmCache::test_get_missing

Why Needed: To test that the get method returns None for missing entries.

Key Assertions:

- {'name': 'result is None', 'expected_value': 'None'}

Confidence: 80%

Tokens: 128 input + 70 output = 198 total

COVERAGE

src/pytest_llm_report/cache.py	9 lines (ranges: 39-41, 53, 55-56, 118-119, 121)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_cache.py::TestLlmCache::test_set_and_get 1ms 4

AI ASSESSMENT

Scenario: Verify that annotations can be set and retrieved from the cache.

Why Needed: Prevents bypass attacks by ensuring that LLMCache stores and retrieves annotations in a secure manner.

Key Assertions:

- Check if the annotation is stored correctly in the cache.
- Check if the retrieved annotation matches the expected value.
- Verify that the confidence level of the retrieved annotation is correct.

Confidence: 80%

Tokens: 286 input + 93 output = 379 total

COVERAGE

src/pytest_llm_report/cache.py	28 lines (ranges: 39-41, 53, 55, 58, 60-62, 68-73, 86, 90, 92, 94, 97-101, 103, 118-119, 121)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_collector.py

11 tests

PASSED `tests/test_collector.py::TestCollectorCollectionErrors::test_collection_error_structure` 1ms 2

AI ASSESSMENT

Scenario:
tests/test_collector.py::TestCollectorCollectionErrors::test_collection_error_structure

Why Needed: The test is checking if the collection errors have the correct structure.

Key Assertions:

- {'assertion': {'message': 'nodeid should be a string'}, 'expected_result': 'nodeid should be a string'}
- {'assertion': {'message': 'message should be a string'}, 'expected_result': 'message should be a string'}

Confidence: 80%

Tokens: 124 input + 116 output = 240 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector.py::TestCollectorCollectionErrors::test_get_collection_errors_initially_empty` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_collector.py::TestCollectorCollectionErrors::test_get_collection_errors_initially_empty

Why Needed: To ensure that the `get_collection_errors` method returns an empty list when the collection is initially empty.

Key Assertions:

- {'name': 'assert collector.get_collection_errors() == []', 'expected_value': [], 'message': 'Expected get_collection_errors() to return an empty list'}

Confidence: 80%

Tokens: 114 input + 103 output = 217 total

COVERAGE

`src/pytest_llm_report/collector.py`	`15 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 285)`
`src/pytest_llm_report/options.py`	`2 lines (ranges: 123, 171)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

PASSED `tests/test_collector.py::TestCollectorMarkerExtraction::test_llm_context_override_default_none` 1ms 2

AI ASSESSMENT

Scenario: tests/test_collector.py::TestCollectorMarkerExtraction::test_llm_context_override_default_none

Why Needed: Default llm_context_override should be None.

Key Assertions:

- {'assertion_type': 'is None', 'expected_value': 'None'}

Confidence: 80%

Tokens: 136 input + 74 output = 210 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector.py::TestCollectorMarkerExtraction::test_llm_opt_out_default_false` 1ms 2

AI ASSESSMENT

Scenario: tests/test_collector.py::TestCollectorMarkerExtraction::test_llm_opt_out_default_false

Why Needed: The default value of llm_opt_out should be False.

Key Assertions:

- {'name': 'llm_opt_out is not equal to False', 'expected_value': False, 'actual_value': 'False'}

Confidence: 80%

Tokens: 136 input + 89 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector.py::TestCollectorOutputCapture::test_capture_enabled_by_default` 1ms 3

AI ASSESSMENT

Scenario: Tests for Collector Output Capture

Why Needed: Test that output capture is enabled by default.

Key Assertions:

- {'name': 'Output capture should be enabled by default', 'description': 'The output capture feature should be enabled by default.'}

Confidence: 80%

Tokens: 104 input + 70 output = 174 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector.py::TestCollectorOutputCapture::test_capture_max_chars_default` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_collector.py::TestCollectorOutputCapture::test_capture_max_chars_default

Why Needed: The default value for capture output max chars is not being tested.

Key Assertions:

- {'name': 'assert config.capture_output_max_chars == 4000', 'expected_result': 4000, 'actual_result': 'tests/test_collector.py::TestCollectorOutputCapture::test_capture_max_chars_default'}

Confidence: 80%

Tokens: 108 input + 105 output = 213 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector.py::TestCollectorXfailHandling::test_xfail_failed_is_xfailed` 1ms 3

AI ASSESSMENT

Scenario: tests/test_collector.py::TestCollectorXfailHandling::test_xfail_failed_is_xfailed

Why Needed: To ensure that xfail failures are correctly recorded as xfailed.

Key Assertions:

- {'expected_value': 'xfailed', 'actual_value': 'xfail'}

Confidence: 80%

Tokens: 206 input + 78 output = 284 total

COVERAGE

src/pytest_llm_report/collector.py	36 lines (ranges: 90, 93-94, 96, 99, 110-112, 114-118, 124, 127, 140, 155-159, 163, 167, 171, 209-210, 212, 216, 227-228, 230-234, 238)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector.py::TestCollectorXfailHandling::test_xfail_passed_is_xpassed` 1ms 3

AI ASSESSMENT

Scenario: tests/test_collector.py::TestCollectorXfailHandling::test_xfail_passed_is_xpassed

Why Needed: xfail passes should be recorded as xpassed.

Key Assertions:

- {'assertion_type': 'type', 'expected_value': 'xpassed'}

Confidence: 80%

Tokens: 205 input + 75 output = 280 total

COVERAGE

src/pytest_llm_report/collector.py	26 lines (ranges: 90, 93-94, 96, 99, 110-112, 114-115, 124, 127, 140, 155-159, 163, 167, 171, 209-210, 212-214)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_collector.py::TestTestCollector::test_create_collector 1ms 3

AI ASSESSMENT

Scenario: Test the `create_collector` method of `TestCollector` class.

Why Needed: This test prevents a potential bug where the collector does not initialize with empty results, potentially leading to incorrect data being collected.

Key Assertions:

- collector.results should be an empty dictionary.
- collector.collection_errors should be an empty list.
- collector.collected_count should be zero.
- assert collector.results == {}
- assert collector.collection_errors == []
- assert collector.collected_count == 0

Confidence: 80%

Tokens: 205 input + 121 output = 326 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_collector.py::TestTestCollector::test_get_results_sorted 1ms 3

AI ASSESSMENT

Scenario: tests/test_collector.py::TestTestCollector::test_get_results_sorted

Why Needed: The test is necessary because it checks if the results are sorted by nodeid.

Key Assertions:

- {'name': 'nodeids', 'expected': ['a_test.py::test_a', 'z_test.py::test_z'], 'actual': ['a_test.py::test_a', 'z_test.py::test_z']}

Confidence: 80%

Tokens: 227 input + 106 output = 333 total

COVERAGE

src/pytest_llm_report/collector.py	15 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 277)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector.py::TestTestCollector::test_handle_collection_finish` 1ms 3

AI ASSESSMENT

Scenario: Test the `handle_collection_finish` method to ensure it correctly tracks collected and deselected counts.

Why Needed: This test prevents a potential issue where the collected count is not updated correctly when items are deselected.

Key Assertions:

- The `collected_count` attribute should be set to 3 after collecting all items.
- The `deselected_count` attribute should be set to 1 after deselecting one item.

Confidence: 80%

Tokens: 256 input + 105 output = 361 total

COVERAGE

`src/pytest_llm_report/collector.py`	`16 lines (ranges: 78-79, 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/options.py`	`2 lines (ranges: 123, 171)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

📄 tests/test_collector_maximal.py

14 tests

PASSED `tests/test_collector_maximal.py::TestCollectorInternals::test_capture_output_disabled_via_handle_report` 2ms 3

AI ASSESSMENT

Scenario: tests/test_collector_maximal.py::TestCollectorInternals::test_capture_output_disabled_via_handle_re

Why Needed: To test that the collector does not capture output when config is disabled (integration via handle_runtest_logreport)

Key Assertions:

- {'name': 'collector.handle_runtest_logreport', 'expected_result': 'None'}

Confidence: 80%

Tokens: 211 input + 90 output = 301 total

COVERAGE

src/pytest_llm_report/collector.py	36 lines (ranges: 90, 93-94, 96, 99, 110-112, 114-118, 124, 127-128, 130, 140, 155-159, 163, 167, 171, 209-210, 227-228, 230-234, 238)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorInternals::test_capture_output_stderr` 1ms 3

AI ASSESSMENT

Scenario: TestCollectorInternals test

Why Needed: To ensure that the `collector._capture_output` method correctly captures stderr output.

Key Assertions:

- {'name': 'Expected captured stderr to be empty', 'expected_value': '', 'actual_value': 'Some error'}

Confidence: 80%

Tokens: 157 input + 76 output = 233 total

COVERAGE

src/pytest_llm_report/collector.py	18 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 261, 264, 268-269)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorInternals::test_capture_output_stdout` 1ms 3

AI ASSESSMENT

Scenario: TestCollectorInternals test capture_output_stdout

Why Needed: To verify that the `test_capture_output_stdout` method correctly captures stdout.

Key Assertions:

- {'name': 'captured_stdout', 'expected_value': 'Some output'}

Confidence: 80%

Tokens: 157 input + 69 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	18 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 261, 264-265, 268)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorInternals::test_capture_output_truncated` 1ms 3

AI ASSESSMENT

Scenario: tests/test_collector_internals

Why Needed: Truncating output exceeding max chars in TestCollectorInternals test.

Key Assertions:

- {'name': 'expected captured stdout length', 'value': 10}

Confidence: 80%

Tokens: 174 input + 65 output = 239 total

COVERAGE

src/pytest_llm_report/collector.py	18 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 261, 264-265, 268)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorInternals::test_create_result_with_item_markers` 2ms 3

AI ASSESSMENT

Scenario: Test creates a result with item markers.

Why Needed: Prevents regression where the collector does not extract item markers correctly when they are present in an item's callspec.

Key Assertions:

- item.get_closest_marker('llm_opt_out') returns MagicMock().
- item.get_closest_marker('llm_context_override') returns MagicMock() with 'complete' args.
- item.get_closest_marker('requirement') returns MagicMock().
- item.get_closest_marker('llm_opt_out') is not None.
- item.get_closest_marker('llm_context_override') is not None.
- item.get_closest_marker('requirement') is not None.
- result.param_id == 'param1' is True.
- result.llm_opt_out is True.
- result.llm_context_override == 'complete' is True.
- result.requirements == ['REQ-1', 'REQ-2'] is True.

Confidence: 80%

Tokens: 382 input + 213 output = 595 total

COVERAGE

src/pytest_llm_report/collector.py	35 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 155-159, 163-164, 167-169, 171, 181-182, 185-189, 198-200, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorInternals::test_extract_error_repr_crash` 1ms 3

COVERAGE

src/pytest_llm_report/collector.py	22 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 227-228, 230-234, 238)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorInternals::test_extract_error_string` 1ms 3

AI ASSESSMENT

Scenario: tests/test_collector_maximal.py::TestCollectorInternals::test_extract_error_string

Why Needed: To ensure the `_extract_error` method returns a string that can be used to recreate the original exception.

Key Assertions:

- {'name': 'assert _extract_error returns correct error message', 'expected_value': 'Some error occurred'}

Confidence: 80%

Tokens: 130 input + 90 output = 220 total

COVERAGE

src/pytest_llm_report/collector.py	22 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 227-228, 230-234, 238)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorInternals::test_extract_skip_reason_fallback` 1ms 3

AI ASSESSMENT

Scenario: tests/test_collector_maximal.py::TestCollectorInternals::test_extract_skip_reason_fallback

Why Needed: To ensure the `_extract_skip_reason` method returns `None` when no longrepr is provided.

Key Assertions:

- {'name': 'assert _extract_skip_reason returns None for no longrepr', 'expected_value': 'None'}

Confidence: 80%

Tokens: 130 input + 92 output = 222 total

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 250, 252)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorInternals::test_extract_skip_reason_string` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_collector_maximal.py::TestCollectorInternals::test_extract_skip_reason_string

Why Needed: To ensure the `extract_skip_reason` method returns a string as expected.

Key Assertions:

- {'name': 'expected return value', 'value': 'Just skipped'}

Confidence: 80%

Tokens: 133 input + 78 output = 211 total

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 250-251)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorInternals::test_extract_skip_reason_tuple` 1ms 3

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 250-251)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorReportHandling::test_handle_collection_report_failure` 1ms 3

AI ASSESSMENT

Scenario: Test the TestCollector handleCollectionReport function when a collection report fails.

Why Needed: This test prevents a regression where the TestCollector does not correctly record and log collection errors.

Key Assertions:

- The collector should have recorded one collection error with nodeid 'test_broken.py' and message 'SyntaxError'.
- The collected error should be of type 'TestCollectorReportFailure'.
- The collector's collection_errors list should contain the expected error object.

Confidence: 80%

Tokens: 273 input + 112 output = 385 total

COVERAGE

src/pytest_llm_report/collector.py	21 lines (ranges: 58, 60-65, 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorReportHandling::test_handle_runtest_rerun` 2ms 3

AI ASSESSMENT

Scenario: Test the `handle_runtest_rerun` method of TestCollector.

Why Needed: This test prevents regression in handling reruns for a specific test case.

Key Assertions:

- The 'rerun' attribute of the report should be set to 1 after a runtest call.
- The final outcome of the test should still be 'failed' even if the rerun count is 1.
- The `results` dictionary in the collector's results object should contain the expected data for the test case.

Confidence: 80%

Tokens: 281 input + 122 output = 403 total

COVERAGE

src/pytest_llm_report/collector.py	42 lines (ranges: 90, 93-94, 96, 99, 110-112, 114-118, 124, 127-128, 130, 140-141, 155-159, 163, 167, 171, 209-210, 227-228, 230-234, 238, 261, 264-265, 268-269)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorReportHandling::test_handle_runtest_setup_failure` 1ms 3

AI ASSESSMENT

Scenario: TestCollectorHandleRuntestSetupFailure verifies that the test collector correctly handles a setup failure in the runtest log report.

Why Needed: This test prevents regression by ensuring that the test collector correctly records and reports errors during setup.

Key Assertions:

- The 'setup' event is recorded with an outcome of 'error'.
- The phase is set to 'setup'.
- The error message is set to 'Setup failed'.

Confidence: 80%

Tokens: 300 input + 106 output = 406 total

COVERAGE

src/pytest_llm_report/collector.py	36 lines (ranges: 90, 93-94, 96, 99-103, 109-112, 114-115, 124, 127, 140, 155-159, 163, 167, 171, 209-210, 227-228, 230-234, 238)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_collector_maximal.py::TestCollectorReportHandling::test_handle_runtest_teardown_failure` 1ms 3

AI ASSESSMENT

Scenario: Test Collector should record error if teardown fails after pass.

Why Needed: To prevent regression in case of a teardown failure, where the test is expected to fail but actually succeeds due to a teardown issue.

Key Assertions:

- res.outcome == 'error'
- res.phase == 'teardown'
- res.error_message == 'Cleanup failed'

Confidence: 80%

Tokens: 391 input + 88 output = 479 total

COVERAGE

src/pytest_llm_report/collector.py	38 lines (ranges: 90, 93-94, 96, 99, 110-112, 114-115, 124, 127-128, 130, 132-133, 135-137, 140, 155-159, 163, 167, 171, 209-210, 227-228, 230-234, 238)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_context_compression.py

12 tests

PASSED tests/test_context_compression.py::TestConfigValidation::test_invalid_compression_mode 1ms 3

AI ASSESSMENT

Scenario: Test invalid context compression mode

Why Needed: To test that an invalid compression mode fails validation.

Key Assertions:

- {'message': 'Context compression must be one of "none", "gzip", or "lzma"', 'expected_value': 'invalid'}

Confidence: 80%

Tokens: 124 input + 74 output = 198 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-269, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_compression.py::TestConfigValidation::test_negative_padding_invalid` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_context_compression.py::TestConfigValidation::test_negative_padding_invalid

Why Needed: Negative padding should fail validation.

Key Assertions:

- {'column': 0, 'line': 1, 'message': 'context_line_padding is not a valid value for this context type.'}

Confidence: 80%

Tokens: 121 input + 81 output = 202 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	27 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273-274, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_compression.py::TestConfigValidation::test_valid_compression_modes` 1ms 3

AI ASSESSMENT

Scenario: TestConfigValidation

Why Needed: To ensure that valid compression modes pass validation.

Key Assertions:

- {'expected': {'context_compression': 'none'}, 'actual': {'context_compression': 'lines'}}

Confidence: 80%

Tokens: 135 input + 64 output = 199 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	26 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_compression.py::TestConfigValidation::test_zero_padding_valid 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_compression.py::TestConfigValidation::test_zero_padding_valid

Why Needed: Zero padding is a valid configuration option.

Key Assertions:

- {'name': 'config', 'type': 'Config', 'value': {'provider': 'none', 'context_line_padding': 0}}

Confidence: 80%

Tokens: 122 input + 83 output = 205 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	26 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_compression.py::TestContextCompression::test_compression_enabled_by_default 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_compression.py::TestContextCompression::test_compression_enabled_by_defaul

Why Needed: Context compression should be enabled by default ('lines').

Key Assertions:

- {'name': 'config.context_compression', 'expected_value': 'lines'}

Confidence: 80%

Tokens: 119 input + 74 output = 193 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_compression.py::TestContextCompression::test_compression_mode_lines` 1ms 3

AI ASSESSMENT

Scenario: Tests for Context Compression

Why Needed: To ensure that the context compression is working correctly and providing the expected behavior.

Key Assertions:

- {'name': 'Context Compression Mode', 'description': 'The context compression mode should be available.', 'expected_value': 'lines'}

Confidence: 80%

Tokens: 113 input + 77 output = 190 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_compression.py::TestContextCompression::test_line_padding_default 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_compression.py::TestContextCompression::test_line_padding_default

Why Needed: To ensure that line padding is correctly set to 2 when the context provider is 'none'.

Key Assertions:

- {'name': 'config.context_line_padding', 'expected_value': 2, 'actual_value': 0}

Confidence: 80%

Tokens: 106 input + 89 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_compression.py::TestExtractCoveredLines::test_contiguous_lines_no_gap 1ms 4

AI ASSESSMENT

Scenario: Contiguous covered lines should not have gap indicators.

Why Needed: Prevents regression where contiguous lines are separated by gaps, potentially misleading the user about coverage.

Key Assertions:

- The # ... indicator is present for contiguous lines.
- The line numbers L3, L4, and L5 are included in the result.
- No gap indicators are added to contiguous lines.

Confidence: 80%

Tokens: 293 input + 94 output = 387 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	23 lines (ranges: 33, 216, 219-220, 223-228, 231-232, 235-237, 239-240, 242, 244-247, 249)

PASSED tests/test_context_compression.py::TestExtractCoveredLines::test_empty_coverage 1ms 4

AI ASSESSMENT

Scenario: tests/test_context_compression.py::TestExtractCoveredLines::test_empty_coverage

Why Needed: Empty coverage is necessary to test the ContextAssembler's ability to extract covered lines from a test with no actual execution.

Key Assertions:

- {'expected_value': '', 'actual_value': ''}

Confidence: 80%

Tokens: 130 input + 79 output = 209 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	3 lines (ranges: 33, 216-217)

PASSED `tests/test_context_compression.py::TestExtractCoveredLines::test_extract_multiple_ranges` 1ms 4

AI ASSESSMENT

Scenario: Test that multiple covered ranges are extracted with gap indicators.

Why Needed: This test prevents regression where the output does not contain gap indicators between ranges of covered lines.

Key Assertions:

- The output should contain a '#' character followed by ' ...' to indicate gaps between ranges.
- The output should contain '# L3:' to indicate range starting at line 3.
- The output should contain '# L15:' to indicate range starting at line 15.
- The output should have gap indicators between the ranges of covered lines.

Confidence: 80%

Tokens: 274 input + 125 output = 399 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	24 lines (ranges: 33, 216, 219-220, 223-228, 231-232, 235-237, 239-240, 242-247, 249)

PASSED tests/test_context_compression.py::TestExtractCoveredLines::test_extract_single_line 1ms 4

AI ASSESSMENT

Scenario: Single covered line should be extracted with padding.

Why Needed: This test prevents a regression where single lines are not extracted correctly due to missing padding.

Key Assertions:

- The function `_extract_covered_lines` includes lines 2, 3, and 4 in the result.
- Lines 2, 3, and 4 have 1 line padding.
- The test asserts that '# L2:', '# L3:', and '# L4:' are present in the result.

Confidence: 80%

Tokens: 302 input + 118 output = 420 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	23 lines (ranges: 33, 216, 219-220, 223-228, 231-232, 235-237, 239-240, 242, 244-247, 249)

PASSED `tests/test_context_compression.py::TestExtractCoveredLines::test_padding_boundary` 1ms 4

AI ASSESSMENT

Scenario: Test Extracted Covered Lines: Padding should not go beyond file boundaries.

Why Needed: This test prevents a bug where padding exceeds the file boundary, potentially causing incorrect coverage metrics.

Key Assertions:

- assert '# L1:' in result
- assert '# L2:' in result
- assert '# L3:' in result

Confidence: 80%

Tokens: 288 input + 85 output = 373 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	23 lines (ranges: 33, 216, 219-220, 223-228, 231-232, 235-237, 239-240, 242, 244-247, 249)

📄 tests/test_context_limits.py

4 tests

PASSED tests/test_context_limits.py::test_no_truncation_needed 1ms 4

AI ASSESSMENT

Scenario: tests/test_context_limits.py::test_no_truncation_needed

Why Needed: This test is needed because the current implementation of `provider._build_prompt` truncates the context when it detects that a node's content exceeds the specified limit.

Key Assertions:

- {'assertion': "The 'prompt' variable should contain the full content of the node.", 'expected_value': 'short content'}
- {'assertion': "The 'truncated' key in the 'prompt' dictionary should be absent.", 'expected_value': ''}

Confidence: 80%

Tokens: 158 input + 133 output = 291 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	24 lines (ranges: 243, 245, 264, 266, 270-272, 274, 277, 279-280, 283, 286, 290-291, 294-295, 298-299, 305, 307-308, 312, 314)
src/pytest_llm_report/llm/utils.py	28 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_limits.py::test_smart_distribution 2ms 4

AI ASSESSMENT

Scenario: test_smart_distribution verifies that the smart distribution logic prevents a regression and optimizes F2's budget.

Why Needed: The test prevents a regression in the smart distribution logic by ensuring F2 gets at least their fair share of tokens, even if it means truncating some content to avoid waste.

Key Assertions:

- F1 should be full (A*160).
- F2 should get at least 110 tokens (100+80).
- F2's budget should not exceed the total available tokens (220).
- The smart split logic uses 40 + 180 = 220 tokens, which is zero waste.
- F2 should be truncated to avoid exceeding their fair share of tokens (480 < 536).

Confidence: 80%

Tokens: 773 input + 169 output = 942 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	25 lines (ranges: 243, 245, 264, 266, 270-272, 274, 277, 279-280, 283, 286, 290-291, 294-295, 298-299, 305, 307-308, 310, 312, 314)
src/pytest_llm_report/llm/utils.py	32 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 90-91, 93-94, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_limits.py::test_splitting_logic 1ms 4

AI ASSESSMENT

Scenario: The test verifies that the splitting logic correctly truncates strings and meets the expected requirements.

Why Needed: This test prevents a potential bug where the splitting logic does not truncate strings, leading to incorrect output or unexpected behavior.

Key Assertions:

- The string 'f1' should be truncated in the prompt.
- The string 'f2' should be truncated in the prompt.
- The keyword 'truncated' should be present in the prompt.

Confidence: 80%

Tokens: 317 input + 109 output = 426 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	24 lines (ranges: 243, 245, 264, 266, 270-272, 274, 277, 279-280, 283, 286, 290-291, 294-295, 298-299, 305, 307, 310, 312, 314)
src/pytest_llm_report/llm/utils.py	30 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 90-91, 93-94, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_limits.py::test_truncation_logic 1ms 4

AI ASSESSMENT

Scenario: The test verifies that the `provider._build_prompt` function truncates prompts to fit within a specified limit.

Why Needed: This test prevents regressions where large context files cause excessive prompting and context is not relevant.

Key Assertions:

- len(prompt) should be less than (limit * 5)
- [... truncated] should be present in the prompt or 'Relevant context' should not be present
- prompt should contain a header indicating truncation

Confidence: 80%

Tokens: 397 input + 111 output = 508 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	9 lines (ranges: 243, 245, 264, 266, 270-272, 274-275)
src/pytest_llm_report/llm/utils.py	1 lines (ranges: 20)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_context_util.py

28 tests

PASSED `tests/test_context_util.py::TestCollapseEmptyLines::test_collapse_three_empty_lines` 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestCollapseEmptyLines::test_collapse_three_empty_lines

Why Needed: The test is necessary because it checks the functionality of the `collapse_empty_lines` function when there are 3+ empty lines in a source string.

Key Assertions:

- {'expected': 'line1\n\nline2', 'actual': 'line1\n\n'}

Confidence: 80%

Tokens: 128 input + 97 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	1 lines (ranges: 108)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestCollapseEmptyLines::test_many_empty_lines` 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestCollapseEmptyLines::test_many_empty_lines

Why Needed: To test the functionality of collapsing many empty lines to one blank line.

Key Assertions:

- {'description': 'The collapsed source code should have only one newline character.', 'expected_value': '\n\nline1\n\nline2'}

Confidence: 80%

Tokens: 127 input + 88 output = 215 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	1 lines (ranges: 108)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestCollapseEmptyLines::test_preserve_two_empty_lines` 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestCollapseEmptyLines::test_preserve_two_empty_lines

Why Needed: Preserves up to 2 consecutive newlines in context.

Key Assertions:

- {'expected_result': 'line1\n\nline2', 'actual_result': 'line1\n\nline2'}

Confidence: 80%

Tokens: 125 input + 83 output = 208 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	1 lines (ranges: 108)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestCollapseEmptyLines::test_single_newline` 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestCollapseEmptyLines::test_single_newline

Why Needed: To test the functionality of collapsing empty lines in a single newline scenario.

Key Assertions:

- {'assertion': 'The result should be the original source string with no changes.', 'expected_result': 'line1\nline2\nline3'}

Confidence: 80%

Tokens: 121 input + 90 output = 211 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	1 lines (ranges: 108)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestOptimizeContext::test_always_collapses_empty_lines` 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestOptimizeContext::test_always_collapses_empty_lines

Why Needed: The test is necessary because it checks if the `optimize_context` function always collapses empty lines, regardless of the flags used.

Key Assertions:

- {'description': 'source should be modified to have only one line with no trailing whitespace', 'expected_result': 'line1\nline2'}

Confidence: 80%

Tokens: 137 input + 102 output = 239 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	6 lines (ranges: 108, 124, 126, 129, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestOptimizeContext::test_combined_optimization` 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestOptimizeContext::test_combined_optimization

Why Needed: To ensure that the combined optimization process is applied correctly to the test context.

Key Assertions:

- {'id': 'combined_optimization_process', 'expected_result': 'The combined optimization process should have been applied successfully.'}
- {'id': 'optimized_context', 'expected_result': 'The test context has been optimized correctly.'}

Confidence: 80%

Tokens: 96 input + 110 output = 206 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	45 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-59, 61-62, 64, 66-69, 81-82, 86, 88-90, 93, 108, 124, 126-127, 129-130, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestOptimizeContext::test_default_strips_docs_only 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestOptimizeContext::test_default_strips_docs_only

Why Needed: The default behavior of the `optimize_context` function should strip all docstrings, not just those in comments.

Key Assertions:

- {'name': 'docstring stripping', 'expected': True, 'actual': False}

Confidence: 80%

Tokens: 100 input + 88 output = 188 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	36 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-56, 58-59, 61-62, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestOptimizeContext::test_empty_source 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestOptimizeContext::test_empty_source

Why Needed: This test is needed because the current implementation of `optimize_context` does not handle empty sources correctly.

Key Assertions:

- {'assertion': 'result == ""', 'expected_result': ''}

Confidence: 80%

Tokens: 95 input + 78 output = 173 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestOptimizeContext::test_source_with_only_whitespace` 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestOptimizeContext::test_source_with_only_whitespace

Why Needed: This test is needed because the current implementation does not handle source code with only whitespace characters correctly.

Key Assertions:

- {'expected_value': ' \n\n \n', 'actual_value': ' \n\n \n'}

Confidence: 80%

Tokens: 115 input + 91 output = 206 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestOptimizeContext::test_strip_both 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestOptimizeContext::test_strip_both

Why Needed: To optimize the context by removing unnecessary docstrings and comments.

Key Assertions:

- {'assertion_type': 'strip both', 'expected_result': {'docstring': '', 'comment': ''}}

Confidence: 80%

Tokens: 95 input + 78 output = 173 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	44 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-56, 58-59, 61-62, 64, 66-69, 81-82, 86, 88-90, 93, 108, 124, 126-127, 129-130, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestOptimizeContext::test_strip_comments_only 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestOptimizeContext::test_strip_comments_only

Why Needed: To optimize the context by removing unnecessary comment lines that do not affect the functionality of the code.

Key Assertions:

- {'assertion_type': 'expected_string_length', 'expected_value': 0, 'actual_value': 1}

Confidence: 80%

Tokens: 95 input + 89 output = 184 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	14 lines (ranges: 81-82, 86, 88-90, 93, 108, 124, 126, 129-130, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestOptimizeContext::test_strip_neither 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestOptimizeContext::test_strip_neither

Why Needed: To ensure that the optimizer can correctly handle cases where neither strip nor optimize is requested.

Key Assertions:

- {'name': 'source', 'expected': 'def foo():'}

Confidence: 80%

Tokens: 94 input + 76 output = 170 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	6 lines (ranges: 108, 124, 126, 129, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestStripComments::test_comment_after_string_with_hash 1ms 3

AI ASSESSMENT

Scenario:
tests/test_context_util.py::TestStripComments::test_comment_after_string_with_hash

Why Needed: To ensure that the function correctly strips comments from strings containing a hash (#) symbol.

Key Assertions:

- {'description': 'The expected output of the test is compared to the actual output.', 'expected_output': "'url = ", 'actual_output': '"http://example.com#anchor"'}

Confidence: 80%

Tokens: 134 input + 103 output = 237 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	7 lines (ranges: 81-82, 86, 88-90, 93)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestStripComments::test_escaped_quotes 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripComments::test_escaped_quotes

Why Needed: To ensure that the context utility correctly handles escaped quotes in strings.

Key Assertions:

- {'description': 'The function should not return a string containing an escaped quote.', 'expected_result': '# comment'}

Confidence: 80%

Tokens: 133 input + 80 output = 213 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	7 lines (ranges: 81-82, 86, 88-90, 93)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestStripComments::test_mixed_quotes 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripComments::test_mixed_quotes

Why Needed: To strip quotes from code that contains both single and double quotes.

Key Assertions:

- {'expected': '"don\'t # worry"', 'actual': '"don\'t \\# worry"'}

Confidence: 80%

Tokens: 101 input + 77 output = 178 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	7 lines (ranges: 81-82, 86, 88-90, 93)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestStripComments::test_no_comments 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripComments::test_no_comments

Why Needed: To strip comments from the source code, ensuring that only relevant information is included in the output.

Key Assertions:

- {'assertion_type': 'contains', 'pattern': '.*#.*', 'expected_result': ''}

Confidence: 80%

Tokens: 91 input + 84 output = 175 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	7 lines (ranges: 81-82, 86, 88-90, 93)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestStripComments::test_preserve_hash_in_double_quoted_string` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_context_util.py::TestStripComments::test_preserve_hash_in_double_quoted_string

Why Needed: Preserves # inside double-quoted strings in source code.

Key Assertions:

- {'assertion': 'strip_comments() removes leading and trailing whitespace from the entire string, including comments.', 'expected_result': '', 'actual_result': 'url = "http://example.com#anchor"'}

Confidence: 80%

Tokens: 135 input + 104 output = 239 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/context_util.py`	`7 lines (ranges: 81-82, 86, 88-90, 93)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

PASSED `tests/test_context_util.py::TestStripComments::test_preserve_hash_in_single_quoted_string` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_context_util.py::TestStripComments::test_preserve_hash_in_single_quoted_string

Why Needed: To ensure that comments are stripped from single-quoted strings, preserving the hash character (#) to maintain code readability.

Key Assertions:

- assert result == 'url = 'http://example.com#anchor',
- # Expected: 'url = 'http://example.com#anchor'

Confidence: 80%

Tokens: 135 input + 96 output = 231 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	7 lines (ranges: 81-82, 86, 88-90, 93)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestStripComments::test_strip_simple_comment` 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripComments::test_strip_simple_comment

Why Needed: To remove simple end-of-line comments from the source code.

Key Assertions:

- {'expected_value': 'x = 1', 'actual_value': 'strip_comments(source)', 'message': "Expected strip_comments(source) to return 'x = 1', but got '{}' instead."}

Confidence: 80%

Tokens: 119 input + 98 output = 217 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	7 lines (ranges: 81-82, 86, 88-90, 93)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestStripComments::test_strip_standalone_comment 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripComments::test_strip_standalone_comment

Why Needed: To strip standalone comments from the test source code.

Key Assertions:

- {'expected': "The line should be stripped of any leading or trailing whitespace and then removed if it's a standalone comment.", 'actual': 'The line remains unchanged.'}

Confidence: 80%

Tokens: 99 input + 89 output = 188 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	7 lines (ranges: 81-82, 86, 88-90, 93)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestStripDocstrings::test_handles_syntax_error_gracefully` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_context_util.py::TestStripDocstrings::test_handles_syntax_error_gracefully

Why Needed: The test is necessary because it checks that the function does not modify the original source code when a syntax error occurs.

Key Assertions:

- {'description': 'The function should return the original source code without any modifications.', 'expected_result': 'def foo(unclosed paren', 'actual_result': 'def foo(unclosed paren'}

Confidence: 80%

Tokens: 119 input + 112 output = 231 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	4 lines (ranges: 27, 29-31)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_context_util.py::TestStripDocstrings::test_multiple_docstrings 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripDocstrings::test_multiple_docstrings

Why Needed: This test is needed because it checks if the context_util module can strip out multiple docstrings from a given string.

Key Assertions:

- {'name': 'strip_multiple_docstrings', 'expected_result': 'Module docstring.'}

Confidence: 80%

Tokens: 95 input + 86 output = 181 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	30 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-59, 61-62, 64, 66-69)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestStripDocstrings::test_preserves_multiline_data_strings` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_context_util.py::TestStripDocstrings::test_preserves_multiline_data_strings

Why Needed: Preserve multiline data strings in docstrings.

Key Assertions:

- {'name': 'docstring triple quotes are preserved', 'expected_value': 'def foo():\n """\n def bar():\n pass\n \\\n return \'Hello, World!\'\n"""', 'actual_value': '\'"""def foo():\n """def bar():\n pass\n \\\n return \'Hello, World!\'\n"""\''}

Confidence: 80%

Tokens: 103 input + 151 output = 254 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	29 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-56, 58-59, 61-62, 64, 66-69)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestStripDocstrings::test_preserves_regular_strings` 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripDocstrings::test_preserves_regular_strings

Why Needed: Preserve regular strings in test context

Key Assertions:

- {'description': 'Regular string is preserved', 'condition': "source.contains('hello world') == True", 'expected_result': True}

Confidence: 80%

Tokens: 102 input + 83 output = 185 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	25 lines (ranges: 27, 29, 33, 35-36, 38-45, 49, 51-52, 55-56, 58, 61, 64, 66-69)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestStripDocstrings::test_preserves_strings_in_structures` 1ms 3

AI ASSESSMENT

LLM error: Failed to parse LLM response as JSON

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	27 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-56, 58, 61, 64, 66-69)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestStripDocstrings::test_strip_multiline_docstring` 1ms 3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripDocstrings::test_strip_multiline_docstring

Why Needed: To ensure that the context utility function works correctly when dealing with multiline docstrings.

Key Assertions:

- {'assertion_type': 'remove_all', 'expected_result': 'str', 'actual_result': 'str'}

Confidence: 80%

Tokens: 97 input + 87 output = 184 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	29 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-56, 58-59, 61-62, 64, 66-69)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestStripDocstrings::test_strip_triple_double_quoted_docstring` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_context_util.py::TestStripDocstrings::test_strip_triple_double_quoted_docstring

Why Needed: To ensure that the context manager works correctly, especially when dealing with triple double-quoted docstrings.

Key Assertions:

- The function `strip_triple_double_quoted_docstring` should remove all triple double-quoted docstrings from the given source code.

Confidence: 80%

Tokens: 106 input + 93 output = 199 total

COVERAGE

`src/pytest_llm_report/collector.py`	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
`src/pytest_llm_report/context_util.py`	29 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-56, 58-59, 61-62, 64, 66-69)
`src/pytest_llm_report/plugin.py`	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_context_util.py::TestStripDocstrings::test_strip_triple_single_quoted_docstring` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_context_util.py::TestStripDocstrings::test_strip_triple_single_quoted_docstring

Why Needed: The test is necessary because the current implementation does not correctly strip triple single-quoted docstrings.

Key Assertions:

- {'name': 'strip_triple_single_quoted_docstring', 'expected_value': '', 'message': 'Expected to return an empty string after stripping triple single-quoted docstrings.'}

Confidence: 80%

Tokens: 106 input + 107 output = 213 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/context_util.py`	`29 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-56, 58-59, 61-62, 64, 66-69)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

📄 tests/test_coverage_boosters.py

3 tests

PASSED `tests/test_coverage_boosters.py::TestCoverageBoosters::test_gemini_model_parsing_edge_cases` 1ms 5

AI ASSESSMENT

Scenario: Tests the parsing of preferred models for a Gemini configuration with edge cases.

Why Needed: This test prevents regression in case the 'm1' or 'm2' model is not available, as it would cause an error when trying to parse them.

Key Assertions:

- assert 'm1' in models
- assert 'm2' in models
- assert provider._parse_preferred_models() == []
- assert provider._parse_preferred_models() == ['All']

Confidence: 80%

Tokens: 273 input + 118 output = 391 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	19 lines (ranges: 134-135, 137-141, 143-144, 476, 478, 524-531)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_boosters.py::TestCoverageBoosters::test_gemini_rate_limiter_edge_math` 1ms 3

AI ASSESSMENT

Scenario: Verify that the rate limiter prevents over and under token limits when recording tokens but not requests.

Why Needed: This test prevents a potential bug where the rate limiter allows excessive token usage without preventing the request from being processed.

Key Assertions:

- The next_available_in() method should return an error (0) for both over and under token limits when recording tokens but not requests.
- The next_available_in() method should return a valid time in seconds for both over and under token limits when recording tokens but not requests.
- The rate limiter should prevent excessive token usage without preventing the request from being processed.

Confidence: 80%

Tokens: 273 input + 142 output = 415 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	35 lines (ranges: 39-42, 45-46, 48, 52-54, 66, 68-70, 81-82, 84, 87-88, 92-93, 95-96, 100-101, 103, 105, 107-114)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_boosters.py::TestCoverageBoosters::test_models_to_dict_variants` 1ms 3

AI ASSESSMENT

Scenario: Verify that the `to_dict()` method of `SourceCoverageEntry` and `LlmAnnotation` classes returns the expected values for coverage percent, error message, and duration.

Why Needed: This test prevents a regression where the coverage percent is not correctly calculated for `SourceCoverageEntry` instances with multiple statements covered.

Key Assertions:

- The value of `d['coverage_percent']` should be equal to 50.0.
- The value of `ann.to_dict()['error']` should be equal to 'timeout'.
- The value of `meta.to_dict()['duration']` should be equal to 1.0.

Confidence: 80%

Tokens: 318 input + 147 output = 465 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	47 lines (ranges: 96-103, 130-133, 135, 137-139, 141, 143, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_coverage_map.py

7 tests

PASSED tests/test_coverage_map.py::TestCoverageMapper::test_create_mapper 1ms 4

AI ASSESSMENT

Scenario: tests/test_coverage_map.py::TestCoverageMapper::test_create_mapper

Why Needed: To ensure the Mapper class initializes with a valid configuration.

Key Assertions:

- {'name': 'mapper.config is config', 'expected_value': 'config'}
- {'name': 'mapper.warnings == []', 'expected_value': []}

Confidence: 80%

Tokens: 109 input + 91 output = 200 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	2 lines (ranges: 44-45)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_coverage_map.py::TestCoverageMapper::test_get_warnings 1ms 4

AI ASSESSMENT

Scenario: tests/test_coverage_map.py::TestCoverageMapper::test_get_warnings

Why Needed: To ensure the get_warnings method returns a list of warnings as expected.

Key Assertions:

- {'name': 'Type of warnings is correct', 'expected_type': 'list', 'actual_type': 'list'}

Confidence: 80%

Tokens: 110 input + 82 output = 192 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	3 lines (ranges: 44-45, 308)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map.py::TestCoverageMapper::test_map_coverage_no_coverage_file` 2ms 5

AI ASSESSMENT

Scenario: Test that the `map_coverage` function returns an empty dictionary when no coverage file exists.

Why Needed: Prevents regression in case of missing coverage files, ensuring accurate coverage reporting.

Key Assertions:

- The `map_coverage()` method should return a dictionary with all keys set to False (indicating no coverage).
- The `map_coverage()` method should have at least one warning (indicating potential issues).

Confidence: 80%

Tokens: 277 input + 102 output = 379 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map.py::TestCoverageMapperContextExtraction::test_extract_nodeid_all_phases` 1ms 4

AI ASSESSMENT

Scenario: The test verifies that the `CoverageMapper` correctly extracts node IDs for all phases when the `include_phase` parameter is set to 'all'.

Why Needed: This test prevents a potential bug where the `CoverageMapper` does not extract node IDs for certain phases, leading to incorrect coverage reports.

Key Assertions:

- The `_extract_nodeid` method returns the expected node ID for each phase.
- The `_extract_nodeid` method returns the same node ID across all phases.
- The `assert` statements check that the extracted node IDs match the expected values.

Confidence: 80%

Tokens: 279 input + 134 output = 413 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	11 lines (ranges: 44-45, 216, 220, 224-225, 228-229, 231, 233, 236)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map.py::TestCoverageMapperContextExtraction::test_extract_nodeid_empty_context` 1ms 4

AI ASSESSMENT

Scenario: tests/test_coverage_map.py::TestCoverageMapperContextExtraction::test_extract_nodeid_empty_c

Why Needed: To handle the case when the context is empty.

Key Assertions:

- {'assertion': 'assert mapper._extract_nodeid([]) == None', 'expected_result': 'None'}
- {'assertion': 'assert mapper._extract_nodeid(None) == None', 'expected_result': 'None'}

Confidence: 80%

Tokens: 128 input + 111 output = 239 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	4 lines (ranges: 44-45, 216-217)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map.py::TestCoverageMapperContextExtraction::test_extract_nodeid_filters_setup` 1ms 4

AI ASSESSMENT

Scenario: tests/test_coverage_map.py::TestCoverageMapperContextExtraction::test_extract_nodeid_filters_se

Why Needed: To filter out setup phase when include_phase=run.

Key Assertions:

- {'assertion_type': 'is_none', 'expected_result': 'None'}

Confidence: 80%

Tokens: 139 input + 76 output = 215 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	9 lines (ranges: 44-45, 216, 220, 224-225, 228-230)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map.py::TestCoverageMapperContextExtraction::test_extract_nodeid_with_run_phase` 1ms 4

AI ASSESSMENT

Scenario: tests/test_coverage_map.py::TestCoverageMapperContextExtraction::test_extract_nodeid_with_rur

Why Needed: To extract the correct node ID from the run phase context.

Key Assertions:

- {'assertion': "nodeid == 'test.py::test_foo'", 'expected_result': 'test.py::test_foo'}

Confidence: 80%

Tokens: 145 input + 90 output = 235 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	11 lines (ranges: 44-45, 216, 220, 224-225, 228-229, 231, 233, 236)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_coverage_map_coverage.py

17 tests

PASSED `tests/test_coverage_map_coverage.py::TestExtractContexts::test_contexts_by_lineno_exception` 1ms 5

AI ASSESSMENT

Scenario: Test 'test_contexts_by_lineno_exception' verifies that the test_contexts_by_lineno function handles exceptions correctly.

Why Needed: The test prevents a potential regression where the test_contexts_by_lineno function fails to handle an exception when accessing contexts for files with multiple lines of code.

Key Assertions:

- The mock_data.contexts_by_lineno.side_effect is set to raise an Exception when it encounters the first file ('file.py') in the measured_files list.
- The test asserts that the result of calling mapper._extract_contexts(mock_data) is an empty dictionary, indicating that the function handled the exception correctly.

Confidence: 80%

Tokens: 332 input + 140 output = 472 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	29 lines (ranges: 44-45, 118, 121-122, 127, 131-135, 137-140, 144, 148, 150, 152, 156, 160-162, 167-170, 199, 202)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED `tests/test_coverage_map_coverage.py::TestExtractContexts::test_no_measured_files` 1ms 4

AI ASSESSMENT

Scenario: Test Extract Contexts

Why Needed: When no measured files are present in the coverage data, an empty dictionary should be returned.

Key Assertions:

- {'name': 'result is a dictionary', 'expected_value': '{}', 'actual_value': 'assert result == {}'}

Confidence: 80%

Tokens: 136 input + 78 output = 214 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	7 lines (ranges: 44-45, 118, 121-122, 127-128)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestExtractContexts::test_skip_non_python_files` 2ms 5

AI ASSESSMENT

Scenario: tests/test_coverage_map_coverage.py::TestExtractContexts::test_skip_non_python_files

Why Needed: To skip non-Python files from coverage reports.

Key Assertions:

- {'assertion_type': 'equals', 'expected_value': {}, 'actual_value': {}}

Confidence: 80%

Tokens: 154 input + 76 output = 230 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	14 lines (ranges: 44-45, 118, 121-122, 127, 131-135, 144-146)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestLoadCoverageData::test_coverage_not_installed` 1ms 4

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	2 lines (ranges: 44-45)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestLoadCoverageData::test_no_coverage_file` 1ms 5

AI ASSESSMENT

Scenario: TestLoadCoverageData

Why Needed: To test the scenario when no .coverage file exists.

Key Assertions:

- {'name': 'result is None', 'expected_value': 'None'}
- {'name': "assert any('W001' in w.code for w in mapper.warnings)", 'expected_value': 'True'}

Confidence: 80%

Tokens: 153 input + 93 output = 246 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	9 lines (ranges: 44-45, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestMapSourceCoverage::test_analysis_exception_handling` 1ms 5

AI ASSESSMENT

Scenario: Test that the test_analysis_exception_handling function prevents regression by catching Analysis2 exceptions and adding warnings.

Why Needed: This test verifies that the test_analysis_exception_handling function handles analysis2 exceptions correctly, preventing potential regressions.

Key Assertions:

- The function should return an empty list when analyzing a source code with no coverage data.
- The function should add a warning to the warnings list for each Analysis2 exception it catches.
- The function should not add any COVERAGE_ANALYSIS_FAILED warnings in this specific test case.

Confidence: 80%

Tokens: 286 input + 122 output = 408 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	22 lines (ranges: 44-45, 243-244, 246-248, 250, 252-254, 259, 261, 263-268, 271, 299-300)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED `tests/test_coverage_map_coverage.py::TestMapSourceCoverage::test_empty_statements` 1ms 5

AI ASSESSMENT

Scenario: Test handling when file has no statements.

Why Needed: To ensure that the coverage map is correctly handled when a file contains no statements.

Key Assertions:

- {'expected_result': [], 'actual_result': []}

Confidence: 80%

Tokens: 178 input + 75 output = 253 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	18 lines (ranges: 44-45, 243-244, 246-248, 250, 252-254, 259-261, 273-274, 299-300)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED `tests/test_coverage_map_coverage.py::TestMapSourceCoverage::test_include_test_files_when_not_configured` 2ms 6

AI ASSESSMENT

Scenario: Test that test files are included when omit_tests_from_coverage is False.

Why Needed: This test prevents regression in case the `omit_tests_from_coverage` configuration flag is set to True without specifying a custom list of test files.

Key Assertions:

- The `covered` attribute of the first result should be equal to 2 (i.e., all test files are included).
- The `missed` attribute of the first result should be equal to 1 (i.e., only one test file is missed).

Confidence: 80%

Tokens: 322 input + 123 output = 445 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	32 lines (ranges: 44-45, 243-244, 246-248, 250, 252, 259-261, 273, 276-279, 281-283, 285-293, 295, 299-300)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	17 lines (ranges: 30, 33, 36, 39, 42, 55, 58-60, 63-64, 100, 103, 111-112, 116, 123)
src/pytest_llm_report/util/ranges.py	13 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65-67)

PASSED `tests/test_coverage_map_coverage.py::TestMapSourceCoverage::test_skip_non_python_files` 1ms 4

AI ASSESSMENT

Scenario: test_skip_non_python_files

Why Needed: Skip non-Python files to ensure accurate coverage reporting.

Key Assertions:

- {'expected_result': {}, 'actual_result': {}}

Confidence: 80%

Tokens: 154 input + 56 output = 210 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	10 lines (ranges: 44-45, 243-244, 246-249, 299-300)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestMapSourceCoverage::test_skip_test_files_when_configured` 1ms 5

AI ASSESSMENT

Scenario: Test that test files are skipped when omit_tests_from_coverage is True.

Why Needed: The test is necessary to ensure that test files are correctly skipped from the coverage report when `omit_tests_from_coverage` is set to `True`. This helps prevent false positives and ensures accurate reporting of covered code.

Key Assertions:

- {'assertion_type': 'contains', 'expected_value': 'Test files are skipped when omit_tests_from_coverage is True.', 'actual_value': 'The test files are included in the coverage report.'}

Confidence: 80%

Tokens: 182 input + 126 output = 308 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	15 lines (ranges: 44-45, 243-244, 246-248, 250, 252-255, 257, 299-300)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED `tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_all_phase_config` 1ms 4

AI ASSESSMENT

Scenario: Test that all phases are accepted when configured.

Why Needed: Prevents regression in phase filtering functionality.

Key Assertions:

- The mapper should return the nodeid for any phase.
- The mapper should match the expected nodeids with the given paths.
- All phases should be included in the coverage report.

Confidence: 80%

Tokens: 305 input + 80 output = 385 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	11 lines (ranges: 44-45, 216, 220, 224-225, 228-229, 231, 233, 236)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_empty_string` 1ms 4

AI ASSESSMENT

Scenario: tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_empty_string

Why Needed: To test that an empty string does not return a node ID.

Key Assertions:

- {'name': 'assert mapper._extract_nodeid() returns None for empty string', 'expected_value': 'None'}

Confidence: 80%

Tokens: 115 input + 85 output = 200 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	4 lines (ranges: 44-45, 216-217)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_none` 1ms 4

AI ASSESSMENT

Scenario:
tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_none

Why Needed: To ensure that the `_extract_nodeid` method handles `None` inputs correctly and returns `None` as expected.

Key Assertions:

- {'assertion': 'mapper._extract_nodeid(None) == None', 'expected_result': 'None'}

Confidence: 80%

Tokens: 114 input + 93 output = 207 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	4 lines (ranges: 44-45, 216-217)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_run_phase_default` 1ms 4

AI ASSESSMENT

Scenario: Test that run phase is the default filter.

Why Needed: This test prevents a regression where the default filter does not match the expected node ID.

Key Assertions:

- mapper._extract_nodeid('test_foo.py::test_bar|run') == 'test_foo.py::test_bar'
- mapper._extract_nodeid('test_foo.py::test_bar|setup') is None
- mapper._extract_nodeid('test_foo.py::test_bar|teardown') is None

Confidence: 80%

Tokens: 297 input + 120 output = 417 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 216, 220, 224-225, 228-231, 233, 236)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_setup_phase_config` 1ms 4

AI ASSESSMENT

Scenario: Test that setup phase is correctly filtered when configured.

Why Needed: Prevents a regression where the test might fail due to incorrect filtering of nodeids in the setup phase.

Key Assertions:

- mapper._extract_nodeid('test_foo.py::test_bar|setup') == 'test_foo.py::test_bar'
- mapper._extract_nodeid('test_foo.py::test_bar|run') is None
- mapper._extract_nodeid('test_foo.py::test_bar|teardown') is None

Confidence: 80%

Tokens: 293 input + 125 output = 418 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 216, 220, 224-225, 228-229, 231-233, 236)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_teardown_phase_config` 1ms 4

AI ASSESSMENT

Scenario: Test that teardown phase is correctly filtered when configured.

Why Needed: This test prevents a potential bug where the teardown phase is not properly filtered, leading to incorrect coverage reporting.

Key Assertions:

- mapper._extract_nodeid('test_foo.py::test_bar|teardown') == 'test_foo.py::test_bar'
- mapper._extract_nodeid('test_foo.py::test_bar|run') is None
- mapper._extract_nodeid('test_foo.py::test_bar|setup') is None

Confidence: 80%

Tokens: 296 input + 125 output = 421 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 216, 220, 224-225, 228-229, 231, 233-234, 236)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_without_pipe` 1ms 4

AI ASSESSMENT

Scenario:
tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_without_pipe

Why Needed: This test is necessary because the current implementation of `CoverageMapper` does not correctly handle node IDs without phase delimiters.

Key Assertions:

- {'assertion_type': 'value', 'expected_value': 'test_foo.py::test_bar'}

Confidence: 80%

Tokens: 136 input + 93 output = 229 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	6 lines (ranges: 44-45, 216, 220, 224, 239)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_coverage_map_maximal.py

9 tests

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_extract_contexts_full_logic` 2ms 6

AI ASSESSMENT

Scenario: Should exercise all paths in _extract_contexts to ensure full logic coverage.

Why Needed: This test prevents regression by verifying that the mapper extracts all necessary contexts for full logic coverage.

Key Assertions:

- assert 'test_app.py::test_one' in result
- assert 'test_app.py::test_two' in result
- assert len(one_cov) == 1 and one_cov[0].line_count == 2
- # Verify app.py is in test_one's coverage

Confidence: 80%

Tokens: 413 input + 118 output = 531 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	57 lines (ranges: 44-45, 118, 121-122, 127, 131-135, 137-140, 144, 148, 150, 152-153, 156, 160-163, 165, 167-168, 173, 176, 178-184, 187-189, 191-194, 196, 199-200, 202, 216, 220, 224-225, 228-229, 231, 233, 236)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	17 lines (ranges: 30, 33, 36, 39, 42, 55, 58-60, 63-64, 100, 103, 111-112, 116, 123)
src/pytest_llm_report/util/ranges.py	13 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65-67)

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_extract_contexts_no_contexts` 1ms 5

AI ASSESSMENT

Scenario: tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_extract_contexts_no_con

Why Needed: To test the coverage mapper's behavior when there are no test contexts.

Key Assertions:

- {'expected': {}, 'actual': {}}

Confidence: 80%

Tokens: 174 input + 73 output = 247 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	14 lines (ranges: 44-45, 118, 121-122, 127, 131-135, 144-146)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_extract_nodeid_variants` 1ms 4

AI ASSESSMENT

Scenario: The test verifies that the `CoverageMapper` correctly extracts node IDs for a scenario where there are missing lines in the code.

Why Needed: This test prevents a potential regression where the coverage map is not accurately reporting the number of covered lines due to missing or filtered code.

Key Assertions:

- The `_extract_nodeid` method should return the expected node ID for each line in the given phase.
- If the line is filtered, the method should return `None`.
- The method should handle cases where there are no lines in the given phase.
- The method should correctly extract node IDs from code without a pipe (|) separating different phases.
- The `_extract_nodeid` method should be able to distinguish between missing and filtered lines.

Confidence: 80%

Tokens: 323 input + 171 output = 494 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	14 lines (ranges: 44-45, 216, 220, 224-225, 228-229, 231-234, 236, 239)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_load_coverage_data_no_files` 1ms 5

AI ASSESSMENT

Scenario: Test that the function correctly handles the case when no coverage files exist.

Why Needed: This test prevents a potential bug where the function would silently fail to load coverage data without raising an error.

Key Assertions:

- The function should return None for _load_coverage_data() when no .coverage files are found.
- There should be exactly one warning message with code 'W001' in mapper.warnings.
- All warnings should have the same code 'W001'.

Confidence: 80%

Tokens: 276 input + 113 output = 389 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	9 lines (ranges: 44-45, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_load_coverage_data_read_error` 2ms 4

AI ASSESSMENT

Scenario: Test that the test_load_coverage_data_read_error function handles errors reading coverage files correctly.

Why Needed: This test prevents a potential regression where the CoverageMapper class fails to handle errors when loading coverage data from corrupted or invalid files.

Key Assertions:

- The function should return None when trying to load coverage data from a corrupt .coverage file.
- Any warnings generated by the mapper should contain 'Failed to read coverage data' in their message.
- The mapper's warnings should not be None, indicating that no errors were found during loading.
- The function should raise an exception when trying to load coverage data from a corrupted or invalid file.
- A mock CoverageData instance with a side effect of raising an Exception on read should be used for mocking purposes.
- The mocked CoverageData instance's read method should return an Exception object.
- Any assertions made within the try block should not raise any AssertionError, indicating that no errors were found during loading.

Confidence: 80%

Tokens: 343 input + 211 output = 554 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	17 lines (ranges: 44-45, 72-73, 83, 86, 88, 92, 94-96, 107-111, 114)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_load_coverage_data_with_parallel_files` 3ms 4

AI ASSESSMENT

Scenario: Test should handle parallel coverage files from xdist and verify that the CoverageMapper correctly updates its data.

Why Needed: This test prevents regression where the CoverageMapper does not update its data when loading coverage files from parallel directories.

Key Assertions:

- The mock instances of `CoverageData` returned by `mock_data_cls.side_effect` should have been updated with at least two calls to `update()`.
- The `update()` method should be called on the mock instance of `CoverageData` that was created in the test fixture.
- The `update()` method should not be called on any other mock instances of `CoverageData` returned by `mock_data_cls.side_effect`.
- Any mock instances of `CoverageData` that were not part of the parallel data (i.e., `mock_parallel_data1` and `mock_parallel_data2`) should not have been updated.
- The `update()` method should be called on any mock instance of `CoverageData` that was created in the test fixture, even if it is not a part of the parallel data.
- Any mock instances of `CoverageData` that were created in the test fixture but are not part of the parallel data should not have been updated.
- The `update()` method should be called on any mock instance of `CoverageData` that was created in the test fixture, even if it is a part of the parallel data.

Confidence: 80%

Tokens: 378 input + 304 output = 682 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	15 lines (ranges: 44-45, 72-73, 83, 86, 88, 92, 94, 98, 101-104, 106)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_map_coverage_no_data` 1ms 4

AI ASSESSMENT

Scenario: Test that the `map_coverage` method returns an empty dictionary when `_load_coverage_data` returns None.

Why Needed: Prevents a potential bug where the `map_coverage` method does not handle cases where there is no coverage data to map.

Key Assertions:

- The `_load_coverage_data` method of the `CoverageMapper` class should be called with `None` as its argument when no data is available.
- The `map_coverage` method should return an empty dictionary (`{}`) when no data is found.
- No exception should be raised if no coverage data is loaded by `_load_coverage_data`.
- The `map_coverage` method should not throw any errors or exceptions when there is no data to map.

Confidence: 80%

Tokens: 228 input + 167 output = 395 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	5 lines (ranges: 44-45, 58-60)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_map_source_coverage_analysis_error` 1ms 5

AI ASSESSMENT

Scenario: Test that the CoverageMapper handles analysis errors during source coverage analysis.

Why Needed: This test prevents a regression where an error in analysis2 causes all files to be skipped without any meaningful output.

Key Assertions:

- The mock_cov.analysis2 side effect is called with an exception 'Analysis failed'.
- mock_data.measured_files returns ['app.py'] as expected.
- mock_cov.get_data returns the mocked data.
- entries is empty after calling mapper.map_source_coverage(mock_cov).

Confidence: 80%

Tokens: 274 input + 118 output = 392 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	22 lines (ranges: 44-45, 243-244, 246-248, 250, 252-254, 259, 261, 263-268, 271, 299-300)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_map_source_coverage_comprehensive` 2ms 6

AI ASSESSMENT

Scenario: Verify that the test covers all paths in map_source_coverage with comprehensive coverage.

Why Needed: This test prevents regression by ensuring that all possible source files are covered under the given configuration.

Key Assertions:

- The function mapper.map_source_coverage returns exactly one entry for 'app.py'.
- The file path of the first entry is correct ('app.py').
- The number of statements in the first entry is correct (3).
- The coverage percentage of the first entry is correct (66.67%).

Confidence: 80%

Tokens: 345 input + 123 output = 468 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	32 lines (ranges: 44-45, 243-244, 246-248, 250, 252, 259-261, 273, 276-279, 281-283, 285-293, 295, 299-300)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	17 lines (ranges: 30, 33, 36, 39, 42, 55, 58-60, 63-64, 100, 103, 111-112, 116, 123)
src/pytest_llm_report/util/ranges.py	14 lines (ranges: 29, 33, 35-37, 39-40, 45-47, 50, 52, 65-66)

📄 tests/test_errors.py

3 tests

PASSED tests/test_errors.py::test_make_warning 1ms 3

AI ASSESSMENT

Scenario: Test the `make_warning` factory function to verify it returns a WarningCode.W001_NO_COVERAGE warning with the specified detail.

Why Needed: This test prevents a potential bug where the `make_warning` function does not correctly identify warnings without coverage files.

Key Assertions:

- The function `make_warning` should return a WarningCode.W001_NO_COVERAGE warning with the specified detail.
- The message of the warning should contain 'No .coverage file found'.
- The detail of the warning should be set to 'test-detail'.

Confidence: 80%

Tokens: 236 input + 128 output = 364 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_errors.py::test_warning_code_values 1ms 2

AI ASSESSMENT

Scenario: Test that warning codes have correct values.

Why Needed: Prevents a potential bug where the warning code values are incorrect, potentially leading to unexpected behavior or errors in the application.

Key Assertions:

- {'message': 'Assertion failed: WarningCode.W001_NO_COVERAGE.value == "W001"'}
- {'message': 'Assertion failed: WarningCode.W101_LLM_ENABLED.value == "W101"'}
- {'message': 'Assertion failed: WarningCode.W201_OUTPUT_PATH_INVALID.value == "W201"'}
- {'message': 'Assertion failed: WarningCode.W301_INVALID_CONFIG.value == "W301"'}
- {'message': 'Assertion failed: WarningCode.W401_AGGREGATE_DIR_MISSING.value == "W401"'}

Confidence: 80%

Tokens: 240 input + 167 output = 407 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_errors.py::test_warning_to_dict 1ms 3

AI ASSESSMENT

Scenario: Test ReportWarning.to_dict() method.

Why Needed: Prevents a warning that is not properly formatted in the report.

Key Assertions:

- The 'code' key should contain the correct value.
- The 'message' key should contain the correct value.
- The 'detail' key should be present and have the correct value.
- The 'code' key should match the expected value of ReportWarning.W001_NO_COVERAGE.
- The 'message' key should match the expected value of ReportWarning.W001_NO_COVERAGE.
- The 'detail' key should match the expected value of some/path.

Confidence: 80%

Tokens: 276 input + 144 output = 420 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	8 lines (ranges: 70-71, 73-75, 77-79)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_errors_maximal.py

6 tests

PASSED `tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_known_code` 1ms 3

AI ASSESSMENT

Scenario: Test verifies that a warning is created with the correct code and message for known code.

Why Needed: To prevent a regression where warnings are not correctly generated when using known code.

Key Assertions:

- The function `make_warning` returns an instance of `WarningCode.W101_LLM_ENABLED` with the correct code.
- The warning message is set to `WARNING_MESSAGES[WarningCode.W101_LLM_ENABLED]`.
- The detail attribute is not provided for warnings with code `WarningCode.W101_LLM_ENABLED`.

Confidence: 80%

Tokens: 222 input + 125 output = 347 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_unknown_code` 1ms 3

AI ASSESSMENT

Scenario: tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_unknown_code

Why Needed: To handle unknown WarningCode values that are not part of the enum.

Key Assertions:

- {'name': 'missing_code', 'expected_type': 'WarningCode', 'actual_type': 'str'}

Confidence: 80%

Tokens: 202 input + 83 output = 285 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_with_detail` 1ms 3

AI ASSESSMENT

Scenario: tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_with_detail

Why Needed: To test the creation of a warning with detail.

Key Assertions:

- {'name': 'w.code == WarningCode.W301_INVALID_CONFIG', 'expected_result': 'True'}
- {'name': 'w.detail == "Bad value"', 'expected_result': 'True'}

Confidence: 80%

Tokens: 127 input + 102 output = 229 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_errors_maximal.py::TestWarningCodes::test_codes_are_strings 1ms 2

AI ASSESSMENT

Scenario: Tests failed

Why Needed: The test 'test_codes_are_strings' is expected to pass. However, it has raised a warning.

Key Assertions:

- {'message': 'Enum values should be strings.', 'expected_type': 'str'}
- {'message': "WarningCode.value.startswith('W').", 'expected_type': 'str'}

Confidence: 80%

Tokens: 109 input + 95 output = 204 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_errors_maximal.py::TestWarningDataClass::test_warning_to_dict_no_detail` 1ms 3

AI ASSESSMENT

Scenario: Tests for ReportWarning class

Why Needed: To ensure that the warning is correctly serialized to a dictionary without any additional details.

Key Assertions:

- {'assertion': "data == {'code': 'W001', 'message': 'No coverage'}", 'expected_result': {'code': 'W001', 'message': 'No coverage'}}

Confidence: 80%

Tokens: 147 input + 91 output = 238 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_errors_maximal.py::TestWarningDataClass::test_warning_to_dict_with_detail` 1ms 3

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	8 lines (ranges: 70-71, 73-75, 77-79)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_fs.py

12 tests

PASSED tests/test_fs.py::TestIsPythonFile::test_non_python_file 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestIsPythonFile::test_non_python_file

Why Needed: The test is checking if the function correctly identifies non-.py files.

Key Assertions:

- {'name': 'is_python_file() should return False for non-.py files', 'description': 'The function should return False for non-.py files', 'expected_result': False}
- {'name': 'is_python_file() should return False for non-.pyc files', 'description': 'The function should return False for non-.pyc files', 'expected_result': False}

Confidence: 80%

Tokens: 115 input + 144 output = 259 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	1 lines (ranges: 79)

PASSED tests/test_fs.py::TestIsPythonFile::test_python_file 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestIsPythonFile::test_python_file

Why Needed: The function `is_python_file()` should be able to identify .py files.

Key Assertions:

- {'name': 'assertion', 'description': 'The function `is_python_file()` should return True for .py files.'}

Confidence: 80%

Tokens: 98 input + 85 output = 183 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	1 lines (ranges: 79)

PASSED tests/test_fs.py::TestMakeRelative::test_makes_path_relative 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestMakeRelative::test_makes_path_relative

Why Needed: To test the functionality of making a path relative to the current working directory.

Key Assertions:

- {'expected_output': 'subdir/file.py', 'actual_output': 'subdir/file.py'}

Confidence: 80%

Tokens: 144 input + 79 output = 223 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 55, 58-60, 63-64)

PASSED `tests/test_fs.py::TestMakeRelative::test_returns_normalized_with_no_base` 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestMakeRelative::test_returns_normalized_with_no_base

Why Needed: To ensure that the `make_relative` function returns a normalized path when no base is provided.

Key Assertions:

- {'expected_value': 'foo/bar', 'actual_value': 'foo/bar'}

Confidence: 80%

Tokens: 107 input + 80 output = 187 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`
`src/pytest_llm_report/util/fs.py`	`7 lines (ranges: 30, 33, 36, 39, 42, 55-56)`

PASSED tests/test_fs.py::TestNormalizePath::test_already_normalized 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestNormalizePath::test_already_normalized

Why Needed: The current implementation of `normalize_path` does not correctly handle already-normalized paths.

Key Assertions:

- {'expected': 'foo/bar', 'actual': 'foo/bar'}
- {'expected': 'normalized', 'actual': 'already normalized'}

Confidence: 80%

Tokens: 96 input + 82 output = 178 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	5 lines (ranges: 30, 33, 36, 39, 42)

PASSED `tests/test_fs.py::TestNormalizePath::test_forward_slashes` 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestNormalizePath::test_forward_slashes

Why Needed: To ensure that the `normalize_path` function correctly handles paths with forward slashes.

Key Assertions:

- {'expected_value': '/foo/bar', 'actual_value': 'foo\\bar'}

Confidence: 80%

Tokens: 100 input + 76 output = 176 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	5 lines (ranges: 30, 33, 36, 39, 42)

PASSED `tests/test_fs.py::TestNormalizePath::test_strips_trailing_slash` 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestNormalizePath::test_strips_trailing_slash

Why Needed: To ensure that the `normalize_path` function correctly removes trailing slashes from file paths.

Key Assertions:

- {'expected_value': '/foo/bar/', 'actual_value': 'foo/bar'}

Confidence: 80%

Tokens: 102 input + 79 output = 181 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	5 lines (ranges: 30, 33, 36, 39, 42)

PASSED tests/test_fs.py::TestShouldSkipPath::test_custom_exclude_patterns 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_custom_exclude_patterns

Why Needed: This test ensures that the `should_skip_path` function correctly handles custom pattern exclusion.

Key Assertions:

- {'name': 'custom patterns should be excluded', 'description': "The path 'tests/conftest.py' should be skipped due to custom patterns."}
- {'name': 'module.py should not be skipped', 'description': "The path 'src/module.py' should not be skipped due to custom patterns."}

Confidence: 80%

Tokens: 126 input + 129 output = 255 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	15 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116-117, 119-121, 123)

PASSED tests/test_fs.py::TestShouldSkipPath::test_normal_path 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_normal_path

Why Needed: The test should be able to pass without skipping any path.

Key Assertions:

- {'name': 'should not skip normal paths', 'expected': True, 'actual': False}

Confidence: 80%

Tokens: 96 input + 76 output = 172 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED tests/test_fs.py::TestShouldSkipPath::test_skips_git 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_skips_git

Why Needed: The current implementation of `should_skip_path` does not correctly handle `.git` directories.

Key Assertions:

- {'name': 'assert should_skip_path is True for .git/objects/foo', 'expected_result': True, 'message': 'Expected should_skip_path to return True for the path .git/objects/foo'}

Confidence: 80%

Tokens: 99 input + 106 output = 205 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED tests/test_fs.py::TestShouldSkipPath::test_skips_pycache 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_skips_pycache

Why Needed: Because the test case is testing the functionality of skipping __pycache__ directories.

Key Assertions:

- {'assertion_type': 'is True', 'expected_value': True}

Confidence: 80%

Tokens: 109 input + 77 output = 186 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED tests/test_fs.py::TestShouldSkipPath::test_skips_venv 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_skips_venv

Why Needed: Because the test case `should_skip_path` is trying to check if a specific directory should be skipped, but it's actually causing an issue with the venv directories.

Key Assertions:

- {'name': 'Should skip venv directories', 'description': 'The function should return True for venv directories and False otherwise'}
- {'name': '.venv/lib/python/site.py', 'description': '.venv is a virtual environment directory, it should be skipped by the function.'}

Confidence: 80%

Tokens: 121 input + 144 output = 265 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

📄 tests/test_fs_coverage.py

15 tests

PASSED `tests/test_fs_coverage.py::TestIsPythonFile::test_is_python_file_false` 1ms 3

AI ASSESSMENT

Scenario: Verifies that a non-.py file does not match the expected behavior.

Why Needed: Prevents potential bugs where a non-.py file is incorrectly identified as Python code.

Key Assertions:

- The function `is_python_file` correctly returns False for non-.py files.
- The function `is_python_file` correctly returns False for non-.py files with the `.pyc` extension.
- The function `is_python_file` does not incorrectly identify a valid Python file as non-Python code.

Confidence: 80%

Tokens: 210 input + 121 output = 331 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	1 lines (ranges: 79)

PASSED tests/test_fs_coverage.py::TestIsPythonFile::test_is_python_file_true 1ms 3

AI ASSESSMENT

Scenario: Testing if a file is a Python file.

Why Needed: Prevents a potential bug where a non-Python file is incorrectly identified as such.

Key Assertions:

- {'message': 'The function is_python_file() should return True for .py files.', 'description': 'The function should correctly identify .py files.'}
- {'message': 'The function is_python_file() should return True for path/to/.py files.', 'description': 'The function should correctly identify path-to/.py files.'}

Confidence: 80%

Tokens: 212 input + 118 output = 330 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	1 lines (ranges: 79)

PASSED `tests/test_fs_coverage.py::TestMakeRelative::test_make_relative_path_not_under_base` 1ms 3

AI ASSESSMENT

Scenario: Test makes a relative path not under the base directory when it is not.

Why Needed: Prevents regression where make_relative fails to return normalized absolute paths for non-relative paths.

Key Assertions:

- The function should return a normalized absolute path as expected.
- The 'project1' should be present in the result.
- The 'file.py' should also be present in the result.
- The relative_to parameter should not cause make_relative to fail for non-relative paths.
- make_relative should correctly normalize the absolute path when the input is not under the base directory.

Confidence: 80%

Tokens: 301 input + 135 output = 436 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	12 lines (ranges: 30, 33, 36, 39, 42, 55, 58-60, 63, 65, 67)

PASSED tests/test_fs_coverage.py::TestMakeRelative::test_make_relative_success 1ms 3

AI ASSESSMENT

Scenario: Test Make Relative

Why Needed: To test the functionality of making a relative path to a file.

Key Assertions:

- {'name': 'Expected output', 'value': 'subdir/file.py'}

Confidence: 80%

Tokens: 147 input + 62 output = 209 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 55, 58-60, 63-64)

PASSED tests/test_fs_coverage.py::TestMakeRelative::test_make_relative_with_none_base 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestMakeRelative::test_make_relative_with_none_base

Why Needed: To ensure that the `make_relative` function correctly handles cases where the base is None.

Key Assertions:

- {'expected': 'path/to/file.py', 'actual': 'path/to/file.py'}

Confidence: 80%

Tokens: 116 input + 82 output = 198 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	7 lines (ranges: 30, 33, 36, 39, 42, 55-56)

PASSED `tests/test_fs_coverage.py::TestNormalizePath::test_normalize_path_backslashes` 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestNormalizePath::test_normalize_path_backslashes

Why Needed: To ensure that backslashes are correctly converted to forward slashes in file paths.

Key Assertions:

- {'expected_result': 'path/to/file.py', 'actual_result': 'path/to/file.py'}

Confidence: 80%

Tokens: 114 input + 80 output = 194 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	5 lines (ranges: 30, 33, 36, 39, 42)

PASSED tests/test_fs_coverage.py::TestNormalizePath::test_normalize_path_path_object 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestNormalizePath::test_normalize_path_path_object

Why Needed: To ensure the `normalize_path` function correctly normalizes path objects, specifically when dealing with file paths.

Key Assertions:

- {'assertion': {'message': "Expected normalized path to be 'path/to/file.py'"}, 'expected_result': 'path/to/file.py'}

Confidence: 80%

Tokens: 110 input + 96 output = 206 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	5 lines (ranges: 30, 33, 36, 39, 42)

PASSED `tests/test_fs_coverage.py::TestNormalizePath::test_normalize_path_trailing_slash` 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestNormalizePath::test_normalize_path_trailing_slash

Why Needed: To ensure that the `normalize_path` function correctly removes trailing slashes from file paths.

Key Assertions:

- {'expected_value': '/path/to/dir/', 'actual_value': 'path/to/dir'}

Confidence: 80%

Tokens: 111 input + 82 output = 193 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	5 lines (ranges: 30, 33, 36, 39, 42)

PASSED `tests/test_fs_coverage.py::TestShouldSkipPath::test_should_not_skip_regular_path` 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestShouldSkipPath::test_should_not_skip_regular_path

Why Needed: Regular paths are not skipped by default.

Key Assertions:

- {'assertion': "should_skip_path('src/module.py') is False", 'expected_result': True}
- {'assertion': "should_skip_path('tests/test_foo.py') is False", 'expected_result': True}

Confidence: 80%

Tokens: 120 input + 107 output = 227 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_git 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_git

Why Needed: The test should skip the .git directory because it contains a Git hook that may be causing issues with the test.

Key Assertions:

- {'name': 'should skip .git directory', 'expected': True}

Confidence: 80%

Tokens: 102 input + 83 output = 185 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED `tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_path_starting_with_skip_dir` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_path_starting_with_skip_dir

Why Needed: To ensure that the function correctly handles paths starting with a skip directory name.

Key Assertions:

- {'name': 'should be True for venv path', 'expected_value': True, 'message': 'The function should return True for paths starting with a skip directory name.'}
- {'name': 'should be True for .venv path', 'expected_value': True, 'message': 'The function should return True for paths starting with a skip directory name.'}

Confidence: 80%

Tokens: 124 input + 146 output = 270 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`
`src/pytest_llm_report/util/fs.py`	`10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)`

PASSED `tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_pycache` 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_pycache

Why Needed: Because the test module __pycache__ is being tested.

Key Assertions:

- The function should skip the __pycache__ directory.

Confidence: 80%

Tokens: 116 input + 62 output = 178 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED `tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_site_packages` 1ms 3

AI ASSESSMENT

Scenario: /usr/lib/python3.12/site-packages/pkg/mod.py

Why Needed: Because it's a site-package directory.

Key Assertions:

- {'name': 'is_site_package_directory', 'value': 'True'}

Confidence: 80%

Tokens: 111 input + 64 output = 175 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_venv 1ms 3

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_venv

Why Needed: The test is checking if venv directories are skipped by the `should_skip_path` function.

Key Assertions:

- {'path': 'venv/lib/python3.12/site.py', 'expected_result': True}
- {'path': '.venv/lib/python3.12/site.py', 'expected_result': True}

Confidence: 80%

Tokens: 130 input + 112 output = 242 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED `tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_with_exclude_patterns` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_with_exclude_patterns

Why Needed: Custom exclude patterns are needed to skip certain files that contain sensitive information.

Key Assertions:

- {'name': 'should_skip_path', 'expected_result': True}
- {'name': 'assert_path', 'expected_result': False, 'message': 'Expected path to be excluded, but it was not.'}

Confidence: 80%

Tokens: 132 input + 110 output = 242 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`
`src/pytest_llm_report/util/fs.py`	`15 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116-117, 119-121, 123)`

📄 tests/test_gemini_provider.py

25 tests

PASSED tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_annotate_loop_daily_limit_hit 3ms 🛡️ 4

AI ASSESSMENT

Scenario: Test that the test_annotate_loop_daily_limit_hit function prevents a daily limit hit when the provider is configured with an empty _models list and a mock limiter that returns None for the daily limit.

Why Needed: This test prevents a potential issue where the provider would exceed its daily limit of requests per day, potentially causing unexpected behavior or errors in downstream applications.

Key Assertions:

- The function should return an error message indicating that the Gemini requests-per-day limit has been reached when the provider is configured with an empty _models list and a mock limiter that returns None for the daily limit.
- The function should not attempt to annotate the internal node as 'passed' when the provider is configured with an empty _models list and a mock limiter that returns None for the daily limit.
- The function should return an error message indicating that the Gemini requests-per-day limit has been reached when the provider is configured with an empty _models list and a mock limiter that returns None for the daily limit.
- The function should not attempt to annotate the internal node as 'passed' when the provider is configured with an empty _models list and a mock limiter that returns None for the daily limit.
- The function should return an error message indicating that the Gemini requests-per-day limit has been reached when the provider is configured with an empty _models list and a mock limiter that returns None for the daily limit.
- The function should not attempt to annotate the internal node as 'passed' when the provider is configured with an empty _models list and a mock limiter that returns None for the daily limit.
- The function should return an error message indicating that the Gemini requests-per-day limit has been reached when the provider is configured with an empty _models list and a mock limiter that returns None for the daily limit.
-

Confidence: 80%

Tokens: 367 input + 393 output = 760 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	22 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)

src/pytest_llm_report/llm/gemini.py	50 lines (ranges: 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-227, 232-233, 318-320, 340, 343, 471-473)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_annotation_exceptions_coverage` 3ms 4

AI ASSESSMENT

Scenario: Test that _GeminiRateLimitExceeded is raised when a request exceeds the limit.

Why Needed: To prevent regression where a request exceeds the rate limit and causes the model to be exhausted.

Key Assertions:

- Mocking the _call_gemini function with a side effect of raising _GeminiRateLimitExceeded.
- Setting the mock_call.side_effect to _GeminiRateLimitExceeded.
- Asserting that res.error contains 'requests-per-day' or 'rate limits reached'.
- _GeminiRateLimitExceeded is called with the correct arguments ('requests_per_day')
- The model is marked as exhausted and its status is updated correctly.
- The _model_exhausted_at dictionary contains the key 'm1' with a value set to True.

Confidence: 80%

Tokens: 730 input + 181 output = 911 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	22 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	100 lines (ranges: 32-34, 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-210, 221-224, 228-230, 232-233, 235-236, 239-244, 263-265, 268, 293, 295, 299-303, 318-320, 340, 343, 471-473)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_coverage_gaps` 167ms 5

AI ASSESSMENT

Scenario: Prevents regression in coverage gaps by ensuring that the rate limiters are correctly configured and annotated for different scenarios.

Why Needed: This test prevents regression in coverage gaps because it ensures that the rate limiters are correctly configured and annotated for different scenarios, such as prompt_override, context too long error, RPD in parse_rate_limits, fallback models, input limits logic (Flash vs Pro).

Key Assertions:

- The _GeminiRateLimiter is correctly configured with a requests_per_minute value of 100.
- The _GeminiRateLimiter is annotated with the correct internal nodeid and outcome for each test scenario.
- The _parse_rate_limits function returns the correct number of requests per day.
- The mock_config.model is set to 'fallback' correctly after patching the provider._fetch_available_models function.
- The input limits logic (Flash vs Pro) works as expected, with the correct value being passed to the models.
- The _models attribute of the provider is correctly updated with the fallback models after patching the provider._ensure_models_and_limits function.

Confidence: 80%

Tokens: 821 input + 238 output = 1059 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	27 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-331)
src/pytest_llm_report/llm/gemini.py	173 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181-182, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246, 249-250, 252, 254-255, 259, 340, 343, 346, 348-356, 358-361, 363-364, 366-367, 435, 437-439, 441-442, 449-455, 457, 459, 461-466, 471-473, 476-

	478, 497-498, 502-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-564, 574)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-52, 55)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_parse_preferred_models_coverage` 1ms 4

AI ASSESSMENT

Scenario: TestGeminiProvider

Why Needed: To ensure the GeminiProvider correctly handles cases where the `model` parameter is not provided or is set to 'ALL'.

Key Assertions:

- {'expected_result': [], 'actual_result': ['[]', []]}

Confidence: 80%

Tokens: 156 input + 73 output = 229 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	13 lines (ranges: 134-135, 137-141, 143-144, 524-527)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_prune_daily_requests` 1ms 3

AI ASSESSMENT

Scenario: TestGeminiProvider

Why Needed: To ensure the Gemini provider is correctly pruning daily requests that are older than 24 hours.

Key Assertions:

- {'assertion': 'The length of limiter._daily_requests is equal to 0 after calling _prune(time.time())', 'expected_result': 0, 'actual_result': 'True'}

Confidence: 80%

Tokens: 157 input + 94 output = 251 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	10 lines (ranges: 39-42, 81-82, 84, 87-89)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_tpm_available_fallback 1ms 3

AI ASSESSMENT

Scenario: Verify that the test_tpm_available_fallback function waits for a sufficient time before allowing token requests.

Why Needed: This test prevents regression where the Gemini provider may not wait long enough for token requests to be processed after a previous request was made.

Key Assertions:

- The total remaining tokens should decrease by at least 30 seconds when the last requested token is used up.
- The function should return before `remaining` decreases below `tokens_used + 1` if `request_tokens <= limit`.
- If `request_tokens` is massive, the function should return immediately and not wait for a sufficient time.
- The remaining tokens should start at `tokens_used` when no requests are made.
- The function should handle cases where `request_tokens > limit` without returning prematurely.
- The test should pass even if `remaining + request_tokens <= limit` is true, indicating that the loop will not return.
- The last requested token should be used up before the function returns.

Confidence: 80%

Tokens: 524 input + 220 output = 744 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	4 lines (ranges: 39-42)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProvider::test_annotate_import_error` 1ms 5

AI ASSESSMENT

Scenario: Test that the test_annotate_import_error function verifies when google-generativeai is not installed.

Why Needed: This test prevents a potential import error caused by missing required dependencies.

Key Assertions:

- The function `provider._annotate_internal` should return an error message indicating 'google-generativeai not installed' when the module is flagged as None in sys.modules.
- The function `provider._annotate_internal` should include the string 'google-generativeai' in its error message.
- The function `provider._annotate_internal` should raise a ValueError with the message 'google-generativeai not installed' when the module is not found in sys.modules.
- The function `provider._annotate_internal` should return None as expected when the module is flagged as None in sys.modules.
- The function `provider._annotate_internal` should include the nodeid and outcome of the test case in its error message.
- The function `provider._annotate_internal` should not raise an exception when the module is installed correctly.

Confidence: 80%

Tokens: 259 input + 224 output = 483 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	14 lines (ranges: 134-135, 137-141, 143-144, 164-165, 167-169)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProvider::test_annotate_no_token` 3ms 5

AI ASSESSMENT

Scenario: Test that annotation fails when token is missing from the environment.

Why Needed: Prevents a potential bug where the Gemini provider throws an error due to an unprovided GEMINI_API_TOKEN.

Key Assertions:

- The 'GEMINI_API_TOKEN' key in the environment should be present.
- The error message should contain 'GEMINI_API_TOKEN is not set'.
- The annotation should fail with this error message when the token is missing.
- The provider should raise an exception instead of throwing a specific error.
- The error message should include the full path to the environment variable.
- The error message should be more informative than just 'GEMINI_API_TOKEN is not set'.
- The test should fail with this error message when the token is missing.

Confidence: 80%

Tokens: 313 input + 178 output = 491 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	22 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	21 lines (ranges: 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-188)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProvider::test_annotate_rate_limit_retry` 5ms 4

AI ASSESSMENT

Scenario: Test that the GeminiProvider correctly annotates a rate limit retry scenario.

Why Needed: This test prevents regression in the GeminiProvider's ability to handle rate limit retries.

Key Assertions:

- The annotation returned by the provider matches the expected scenario.
- The mock_post call count is correct, indicating that the provider successfully retried the API after the first failure.
- The _parse_response method of the provider returns a Mock object with the correct scenario and error status code.
- The provider's internal state is updated correctly to reflect the retry attempt.
- The annotation does not contain any additional or incorrect information about the source.
- No other critical checks are performed by this test, but it ensures that the provider behaves as expected in this specific scenario.

Confidence: 80%

Tokens: 636 input + 171 output = 807 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	19 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221)
src/pytest_llm_report/llm/gemini.py	214 lines (ranges: 32-34, 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-237, 239-244, 246, 249-250, 252, 261, 263-265, 299-300, 304-306, 308-309, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413-416, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-452, 463-466, 471-473, 476-478, 497-498, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531,

	537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569, 574)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProvider::test_annotate_success` 404ms 4

AI ASSESSMENT

Scenario: Verify that the _annotate_success method returns a correct annotation when successful.

Why Needed: This test prevents regression in the GeminiProvider's _annotate_internal method, which may return an incorrect annotation if the response from _call_gemini is not in the expected format.

Key Assertions:

- The annotation returned by _annotate_internal has the correct scenario 'Success Scenario'.
- The annotation does not have any error.
- The annotation's outcome is correctly set to 'passed'.
- _parse_response returns the correct text and tokens when called with a successful response from _call_gemini.
- The _build_prompt method is not mocked, which could potentially cause issues if it has complex dependencies.
- _annotate_internal calls _parse_response where the expected format is used.
- The annotation's error is None when it should be an error.
- The annotation's outcome is correctly set to 'passed' even though there was no error.

Confidence: 80%

Tokens: 649 input + 212 output = 861 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	19 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221)
src/pytest_llm_report/llm/gemini.py	208 lines (ranges: 39-42, 45-46, 48, 52-54, 66, 68-70, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246-247, 249-252, 261, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-430, 432, 435, 437-439, 441-444, 449-

	452, 463-466, 471-473, 476-478, 497-498, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567-568, 574)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_gemini_provider.py::TestGeminiProvider::test_availability 3ms 5

AI ASSESSMENT

Scenario: Verifies that the GeminiProvider class correctly checks for availability based on environment variables.

Why Needed: This test prevents a potential bug where the provider may not be available due to missing or incorrect environment variables.

Key Assertions:

- The `GeminiProvider` instance is created with the correct `provider` set to 'gemini' when environment variable 'GEMINI_API_TOKEN' is present.
- The `_check_availability()` method returns False when environment variable 'GEMINI_API_TOKEN' is not present.
- The `_check_availability()` method returns True when environment variable 'GEMINI_API_TOKEN' is present and correct.

Confidence: 80%

Tokens: 235 input + 147 output = 382 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	12 lines (ranges: 134-135, 137-141, 143-144, 332-333, 335)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_annotate_retry_exceptions` 60.00s 4

AI ASSESSMENT

Scenario: Test that the GeminiProvider class correctly handles retry exceptions and model exhaustion when calling _annotate_internal with a mock ResourceExhausted exception.

Why Needed: This test prevents regression in the GeminiProvider class, where it may not handle retry exceptions or model exhaustion correctly when calling _annotate_internal.

Key Assertions:

- Verify that the provider's _model_exhausted_at dictionary contains 'm1' after a mock ResourceExhausted exception is raised.
- Verify that the provider's _cooldowns dictionary contains 'm1' after a mock ResourceExhausted exception is raised and the cooldown period exceeds 5.5 seconds.
- Verify that the provider correctly handles retry after cleanup by checking if the cooldown period for 'm1' has exceeded 5.5 seconds.
- Verify that the provider's _rate_limiters are properly set up to avoid network calls when calling _get_rate_limiter → _ensure_rate_limits.
- Verify that the provider raises a MockResourceExhausted exception with the correct message and error code when calling _call_gemini.
- Verify that the provider correctly handles model exhaustion by checking if 'm1' is in the _model_exhausted_at dictionary after a mock ResourceExhausted exception is raised.
- Verify that the provider's cooldowns are properly set up to handle retry after cleanup.
- Verify that the provider does not raise an error when calling _call_gemini with a mock ResourceExhausted exception.

Confidence: 80%

Tokens: 651 input + 317 output = 968 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	22 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	111 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-210, 221-224, 228-230, 232-233, 235-237, 239-244, 263-265, 268, 272-276, 279-281, 283-286,

	288-292, 318-320, 322-323, 340, 343, 471-473)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_annotate_retry_loop_coverage` 3ms 5

AI ASSESSMENT

Scenario: Test that the GeminiProvider correctly clears _model_exhausted_at when annotating with a successful call to _call_gemini.

Why Needed: The test prevents regression where the _model_exhausted_at is not cleared after a successful annotation, potentially leading to incorrect assertions in other tests.

Key Assertions:

- provider._model_exhausted_at[model] = None
- provider._models[model] == [model]
- provider._rate_limiters[model].next_available_in.return_value == 0
- mock_call.return_value[0][0] == 'response'
- mock_call.return_value[1][0] == LlmTokenUsage(10, 10, 20)
- assert mock_limiter.next_available_in.called_once_with(0)
- assert provider._rate_limiters[model].next_available_in.called_once_with(0)
- assert provider._model_exhausted_at[model] is None

Confidence: 80%

Tokens: 482 input + 210 output = 692 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	27 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-331)
src/pytest_llm_report/llm/gemini.py	97 lines (ranges: 39-42, 45-46, 48, 52-54, 66, 68-70, 73, 76-78, 81-82, 84, 87-88, 92-94, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-210, 212-213, 215-216, 218, 222-224, 228-230, 232, 235-236, 239-244, 246-247, 249-252, 254, 259, 340, 343, 471-473)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-52, 55)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_ensure_rate_limits_error` 1ms 4

AI ASSESSMENT

Scenario: TestGeminiProviderDetailed::test_ensure_rate_limits_error

Why Needed: To test that the ` GeminiProvider` raises an exception when rate limiting is attempted with a non-numeric value.

Key Assertions:

- {'name': 'Expected error to be raised', 'description': 'The test should expect an exception to be raised when attempting to limit rate with a non-numeric value.'}

Confidence: 80%

Tokens: 156 input + 100 output = 256 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	27 lines (ranges: 134-135, 137-141, 143-144, 346, 348-356, 358-361, 363-364, 366-367)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_fetch_available_models_error` 1ms 5

AI ASSESSMENT

Scenario: TestGeminiProviderDetailed.test_fetch_available_models_error

Why Needed: To test the error handling of fetching available models when a network error occurs.

Key Assertions:

- {'name': 'models', 'expected_value': [], 'type': 'list'}
- {'name': 'limit_map', 'expected_value': {}, 'type': 'dict'}

Confidence: 80%

Tokens: 132 input + 98 output = 230 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	15 lines (ranges: 134-135, 137-141, 143-144, 537, 539-541, 544-545)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_fetch_available_models_invalid_json` 2ms 5

AI ASSESSMENT

Scenario: Test that fetching available models with invalid JSON data prevents a bug related to model validation.

Why Needed: This test verifies that the GeminiProvider class correctly handles invalid JSON input when fetching available models.

Key Assertions:

- The 'm1' model should not be included in the list of available models.
- The 'm2' model should not be included in the list of available models.
- The 'm3' model should be included in the list of available models.
- The 'inputTokenLimit' field of the 'm3' model should match the expected value.
- The 'supportedGenerationMethods' field of the 'm3' model should only contain 'generateContent'.
- The 'limitMap' dictionary should not contain the 'm1', 'm2', or 'm3' models.

Confidence: 80%

Tokens: 340 input + 187 output = 527 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	34 lines (ranges: 134-135, 137-141, 143-144, 476-477, 537, 539-543, 547-548, 550-559, 562-563, 567, 569, 574)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_get_max_context_tokens_calls_ensure` 2ms 4

AI ASSESSMENT

Scenario: tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_get_max_context_tokens_calls_ensu

Why Needed: To ensure that the `get_max_context_tokens` method of the GeminiProvider class calls the mock function correctly.

Key Assertions:

- {'name': 'mock_ensure was called once', 'expected_value': 1}

Confidence: 80%

Tokens: 144 input + 91 output = 235 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	3 lines (ranges: 65-66, 163)
src/pytest_llm_report/llm/gemini.py	15 lines (ranges: 134-135, 137-141, 143-144, 486, 488-491, 493)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_parse_rate_limits_types` 1ms 5

AI ASSESSMENT

Scenario:
tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_parse_rate_limits_types

Why Needed: This test ensures that the GeminiProvider can correctly parse rate limits from a JSON configuration.

Key Assertions:

- {'name': 'config.requests_per_minute', 'expected_value': 'None'}
- {'name': 'config.tokens_per_minute', 'expected_value': 100}

Confidence: 80%

Tokens: 156 input + 103 output = 259 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	24 lines (ranges: 134-135, 137-141, 143-144, 449-457, 459-460, 463-466)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiRateLimiter::test_prune_logic` 1ms 3

AI ASSESSMENT

Scenario: Verify that the `prune_logic` method correctly removes old requests and updates token usage when a new request is added.

Why Needed: This test prevents regression in the `prune_logic` method, which may cause outdated data to be returned for existing requests.

Key Assertions:

- The length of `_request_times` should be equal to 1 after pruning.
- The length of `_token_usage` should be equal to 1 after pruning.
- The value of `now - 10.0` in `_request_times[0]` should match the current time.
- _request_times[-1] == now - 61.0
- _token_usage[-1][0] == now - 61.0
- _token_usage[-1][1] == 10

Confidence: 80%

Tokens: 323 input + 180 output = 503 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	11 lines (ranges: 39-42, 81-85, 87-88)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiRateLimiter::test_record_tokens_invalid` 1ms 3

AI ASSESSMENT

Scenario: tests/test_gemini_provider.py::TestGeminiRateLimiter::test_record_tokens_invalid

Why Needed: The test is failing because the rate limiter is not correctly handling invalid token records.

Key Assertions:

- {'name': 'len(limiter._token_usage) == 0', 'expected_result': 0, 'actual_result': 1}

Confidence: 80%

Tokens: 127 input + 95 output = 222 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	6 lines (ranges: 39-42, 66-67)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiRateLimiter::test_rpd_limit` 1ms 3

AI ASSESSMENT

Scenario: Test that the rate limiter does not exceed the limit when no requests are made.

Why Needed: The test ensures that the rate limiter does not allow more requests than allowed by the configuration, which would result in a 'RateLimitExceeded' error.

Key Assertions:

- {'name': 'limiter.record_request() should not be called when no requests are made', 'expected_result': 'None'}
- {'name': 'next_available_in() should return None for 100 requests', 'expected_result': 'None'}

Confidence: 80%

Tokens: 129 input + 133 output = 262 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	18 lines (ranges: 39-42, 45-46, 48-50, 73, 76-78, 81-82, 84, 87-88)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_gemini_provider.py::TestGeminiRateLimiter::test_rpm_limit 1ms 3

AI ASSESSMENT

Scenario: Verify that the rate limiter does not block the third request after two successful requests.

Why Needed: This test prevents a potential issue where the third request is blocked due to insufficient available time for subsequent requests.

Key Assertions:

- The `next_available_in` method returns 0.0 when there are no more available slots.
- The `next_available_in` method returns a value between 0 and 60.0 when there are still available slots.
- The `record_request` method is called before the third request waits for an available slot.

Confidence: 80%

Tokens: 280 input + 130 output = 410 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	27 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-97, 100-102)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiRateLimiter::test_seconds_until_tpm_available_branches` 1ms 3

AI ASSESSMENT

Scenario: Verify that the rate limiter correctly handles requests exceeding the limit when there are no tokens available.

Why Needed: This test prevents a potential bug where the rate limiter does not properly handle scenarios where there are no tokens available and more than one minute has passed since the last request.

Key Assertions:

- assert wait > 0
- assert wait <= 60.0 + 1e-9
- # Tokens have been exhausted, but usage is still within limit

Confidence: 80%

Tokens: 377 input + 114 output = 491 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	18 lines (ranges: 39-42, 100-101, 103-114)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiRateLimiter::test_wait_for_slot_daily_limit_exceeded` 1ms 3

AI ASSESSMENT

Scenario: Verify that the `wait_for_slot` method raises an exception when the daily limit is exceeded.

Why Needed: This test prevents a potential bug where the rate limiter does not raise an exception when the daily limit is exceeded, potentially causing unexpected behavior or errors in downstream systems.

Key Assertions:

- The `wait_for_slot` method raises an instance of `_GeminiRateLimitExceeded` with the correct `limit_type` attribute.
- The `limit_type` attribute of the exception raised by `wait_for_slot` is set to `

Confidence: 80%

Tokens: 263 input + 126 output = 389 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	24 lines (ranges: 32-34, 39-42, 45-46, 48-50, 58-60, 73, 76-78, 81-82, 84, 87-88)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_gemini_provider.py::TestGeminiRateLimiter::test_wait_for_slot_sleeps` 2ms 3

AI ASSESSMENT

Scenario: Test that the `wait_for_slot` method sleeps for a sufficient amount of time when waiting for an available slot.

Why Needed: This test prevents regression where the rate limiter does not sleep long enough to allow subsequent requests to wait their turn, potentially leading to performance issues or errors.

Key Assertions:

- ...
- ...
- ...

Confidence: 80%

Tokens: 325 input + 86 output = 411 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/llm/gemini.py`	`18 lines (ranges: 39-42, 58-59, 61-63, 73, 76-78, 81-82, 84, 87-88)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

📄 tests/test_hashing.py

13 tests

PASSED tests/test_hashing.py::TestComputeConfigHash::test_different_config 1ms 4

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestComputeConfigHash::test_different_config

Why Needed: To ensure that different configurations of the Compute API produce different hashes.

Key Assertions:

- {'assertion': 'compute_config_hash(config1) != compute_config_hash(config2)', 'expected_result': 'different'}

Confidence: 80%

Tokens: 119 input + 82 output = 201 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	8 lines (ranges: 96-101, 103-104)

PASSED tests/test_hashing.py::TestComputeConfigHash::test_returns_short_hash 1ms 4

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestComputeConfigHash::test_returns_short_hash

Why Needed: To ensure the computed hash is short and does not exceed 16 characters.

Key Assertions:

- {'message': 'The length of the hash should be equal to 16'}
- {'message': 'The hash should not be longer than 16 characters'}

Confidence: 80%

Tokens: 109 input + 87 output = 196 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	8 lines (ranges: 96-101, 103-104)

PASSED tests/test_hashing.py::TestComputeFileSha256::test_consistent_with_bytes 1ms 3

AI ASSESSMENT

Scenario: File hashing consistency

Why Needed: To ensure that the hash of a file matches its content.

Key Assertions:

- {'name': 'file_hash', 'expected_value': 'content_hash'}

Confidence: 80%

Tokens: 144 input + 60 output = 204 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	6 lines (ranges: 32, 44-48)

PASSED `tests/test_hashing.py::TestComputeFileSha256::test_hashes_file` 1ms 3

AI ASSESSMENT

Scenario: Hashing a file

Why Needed: To test the correctness of the hash computation function.

Key Assertions:

- {'description': 'The computed SHA-256 hash should be 64 bytes long.', 'expected_value': 64, 'actual_value': 0}
- {'description': 'The file contents should not be modified during the test.', 'expected_value': 'hello world', 'actual_value': 'hello world'}

Confidence: 80%

Tokens: 124 input + 114 output = 238 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`
`src/pytest_llm_report/util/hashing.py`	`5 lines (ranges: 44-48)`

PASSED `tests/test_hashing.py::TestComputeHmac::test_different_key` 1ms 3

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestComputeHmac::test_different_key

Why Needed: To ensure that different keys produce different signatures.

Key Assertions:

- {'expected': 'different', 'actual': 'same'}

Confidence: 80%

Tokens: 125 input + 65 output = 190 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`
`src/pytest_llm_report/util/hashing.py`	`1 lines (ranges: 61)`

PASSED `tests/test_hashing.py::TestComputeHmac::test_with_key` 1ms 3

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestComputeHmac::test_with_key

Why Needed: To verify the correctness of HMAC computation with a key.

Key Assertions:

- {'name': 'expected length of signature', 'value': 64}

Confidence: 80%

Tokens: 108 input + 70 output = 178 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	1 lines (ranges: 61)

PASSED `tests/test_hashing.py::TestComputeSha256::test_consistent` 1ms 3

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestComputeSha256::test_consistent

Why Needed: To ensure that the hash function is consistent and produces the same output for the same input.

Key Assertions:

- {'expected': {'hash': '...'}, 'actual': {'hash': '...'}}

Confidence: 80%

Tokens: 115 input + 80 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	1 lines (ranges: 32)

PASSED `tests/test_hashing.py::TestComputeSha256::test_length` 1ms 3

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestComputeSha256::test_length

Why Needed: To ensure the length of the computed SHA-256 hash is 64 characters (64 hexadecimal digits).

Key Assertions:

- {'assertion': 'The length of the hash should be equal to 64.', 'expected_result': 64}

Confidence: 80%

Tokens: 103 input + 88 output = 191 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	1 lines (ranges: 32)

PASSED `tests/test_hashing.py::TestGetDependencySnapshot::test_includes_pytest` 81ms 3

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestGetDependencySnapshot::test_includes_pytest

Why Needed: To ensure that the 'pytest' package is included in the dependency snapshot.

Key Assertions:

- {'name': 'includes pytest package', 'description': "The 'pytest' package should be present in the dependency snapshot."}

Confidence: 80%

Tokens: 102 input + 86 output = 188 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	8 lines (ranges: 113-114, 116-121)

PASSED tests/test_hashing.py::TestGetDependencySnapshot::test_returns_dict 82ms 3

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestGetDependencySnapshot::test_returns_dict

Why Needed: To ensure that the `get_dependency_snapshot` function returns a dictionary as expected.

Key Assertions:

- {'name': 'snapshot is a dict', 'expected': 'dict', 'actual': 'True'}

Confidence: 80%

Tokens: 98 input + 82 output = 180 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	8 lines (ranges: 113-114, 116-121)

PASSED tests/test_hashing.py::TestLoadHmacKey::test_loads_key 1ms 4

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestLoadHmacKey::test_loads_key

Why Needed: To test that the hmac.load function correctly loads a key from a file.

Key Assertions:

- {'expected_value': 'my-secret-key\n', 'actual_value': '', 'error_message': ''}

Confidence: 80%

Tokens: 145 input + 82 output = 227 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	5 lines (ranges: 73, 76-77, 80-81)

PASSED tests/test_hashing.py::TestLoadHmacKey::test_missing_key_file 1ms 4

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestLoadHmacKey::test_missing_key_file

Why Needed: The test should return None if the key file does not exist.

Key Assertions:

- {'message': 'Expected the function to return None when the key file does not exist.'}

Confidence: 80%

Tokens: 126 input + 76 output = 202 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	4 lines (ranges: 73, 76-78)

PASSED tests/test_hashing.py::TestLoadHmacKey::test_no_key_file 1ms 4

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestLoadHmacKey::test_no_key_file

Why Needed: Because the test case requires a valid HMAC key to be loaded.

Key Assertions:

- {'name': 'assert is None', 'expected_result': 'None'}

Confidence: 80%

Tokens: 110 input + 74 output = 184 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	2 lines (ranges: 73-74)

📄 tests/test_integration_gate.py

16 tests

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_aggregation_defaults` 1ms 🛡️ 3

AI ASSESSMENT

Scenario: Verify aggregation configuration defaults.

Why Needed: Prevents a potential bug where aggregation settings are not properly initialized with default values.

Key Assertions:

- config.aggregate_dir is None
- config.aggregate_policy == 'latest'
- config.aggregate_include_history is False

Confidence: 80%

Tokens: 201 input + 70 output = 271 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 293)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_capture_failed_output_default_true` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_integration_gate.py::TestConfigDefaults::test_capture_failed_output_default_true

Why Needed: The test captures failed output by default.

Key Assertions:

- {'name': 'assert config.capture_failed_output is True', 'expected_result': 'True'}

Confidence: 80%

Tokens: 107 input + 73 output = 180 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 293)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_context_mode_default_minimal` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_integration_gate.py::TestConfigDefaults::test_context_mode_default_minimal

Why Needed: To ensure the context mode is set to 'minimal' by default.

Key Assertions:

- {'name': 'config.llm_context_mode', 'expected_value': 'minimal'}

Confidence: 80%

Tokens: 107 input + 77 output = 184 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/options.py`	`3 lines (ranges: 123, 171, 293)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_llm_not_enabled_by_default` 1ms 3

AI ASSESSMENT

Scenario: tests/test_integration_gate.py::TestConfigDefaults::test_llm_not_enabled_by_default

Why Needed: LLM is currently disabled by default.

Key Assertions:

- {'name': 'is_llm_enabled', 'expected_value': False, 'actual_value': 'not_llm_enabled'}

Confidence: 80%

Tokens: 109 input + 81 output = 190 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	4 lines (ranges: 123, 171, 284, 293)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_omit_tests_default_true` 1ms 3

AI ASSESSMENT

Scenario: tests/test_integration_gate.py::TestConfigDefaults::test_omit_tests_default_true

Why Needed: The test is necessary because it checks the default behavior of omitting tests from coverage.

Key Assertions:

- {'name': 'config.omit_tests_from_coverage', 'value': 'True'}

Confidence: 80%

Tokens: 109 input + 80 output = 189 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/options.py`	`3 lines (ranges: 123, 171, 293)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_provider_default_none` 1ms 3

AI ASSESSMENT

Scenario: tests/test_integration_gate.py::TestConfigDefaults::test_provider_default_none

Why Needed: The provider is set to 'none' by default, which may not be suitable for all use cases.

Key Assertions:

- config.provider == 'none'

Confidence: 80%

Tokens: 101 input + 64 output = 165 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 293)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_secret_exclude_globs` 1ms 3

AI ASSESSMENT

Scenario: Integration test of gate configuration

Why Needed: To ensure that secret files are excluded from the LLM context.

Key Assertions:

- {'name': 'Exclude secret files by default', 'description': "The function should exclude 'secret' files and '.env' files from the list of excludes."}
- {'name': 'Exclude .env files', 'description': ".env files are a common source of sensitive information, so it's essential to exclude them."}

Confidence: 80%

Tokens: 132 input + 118 output = 250 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/options.py`	`3 lines (ranges: 123, 171, 293)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

PASSED `tests/test_integration_gate.py::TestFullPipeline::test_deterministic_output` 11ms 5

AI ASSESSMENT

Scenario: The test verifies that the deterministic output of the integration gate is correctly reported.

Why Needed: This test prevents a regression where the deterministic output may not be reported correctly due to changes in the test data or configuration.

Key Assertions:

- nodeids are sorted by nodeid
- tests are passed
- output is deterministic

Confidence: 80%

Tokens: 313 input + 84 output = 397 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	80 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	122 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_integration_gate.py::TestFullPipeline::test_empty_test_suite` 11ms 6

AI ASSESSMENT

Scenario: Test that an empty test suite produces a valid report.

Why Needed: This test prevents regression in case the test suite is empty.

Key Assertions:

- The total count of tests should be zero.
- The summary section should have no data.
- No error messages or warnings should be printed.
- The output file should not contain any report content.
- The report writer should not raise an exception when writing to a non-existent file.
- The test suite should not produce any invalid JSON data.

Confidence: 80%

Tokens: 240 input + 121 output = 361 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	62 lines (ranges: 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	123 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202-206, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_integration_gate.py::TestFullPipeline::test_html_report_generation` 45ms 6

AI ASSESSMENT

Scenario: Test that the full pipeline generates an HTML report.

Why Needed: This test prevents regression where the pipeline does not generate an HTML report even when all tests pass.

Key Assertions:

- The file named `report.html` exists in the temporary directory.
- The string '' is present in the content of the `report.html` file.
- The string 'test_pass' is present in the content of the `report.html` file.

Confidence: 80%

Tokens: 270 input + 108 output = 378 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	118 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_integration_gate.py::TestFullPipeline::test_json_report_generation` 67ms 7

AI ASSESSMENT

Scenario: Verify that a full pipeline generates a valid JSON report with the correct schema version, summary statistics, and skipped tests.

Why Needed: This test prevents regression in the integration gate by ensuring that the full pipeline correctly generates a JSON report with the expected structure and content.

Key Assertions:

- The 'schema_version' key is present in the JSON report and has the correct value.
- The 'summary' key contains the correct total, passed, failed, and skipped counts.
- The 'passed', 'failed', and 'skipped' keys contain the expected values for each type of test.
- The number of tests with a status of 'skipped' is equal to the number of tests that were not executed (i.e., 'skipped' tests).
- The total count of all tests is 3, as specified in the test results.
- The passed count is 1, as specified in the test results.
- The failed count is 1, as specified in the test results.
- The skipped count is 1, as specified in the test results.

Confidence: 80%

Tokens: 419 input + 240 output = 659 total

COVERAGE

src/pytest_llm_report/_git_info.py	2 lines (ranges: 2-3)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	80 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	138 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130,

	156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-329, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_integration_gate.py::TestSchemaCompatibility::test_report_root_has_required_fields` 1ms 3

AI ASSESSMENT

Scenario: Tests ReportRoot has required fields.

Why Needed: This test ensures that the report root contains all necessary fields for a valid schema.

Key Assertions:

- 'schema_version' is present in `data`
- 'run_meta' is present in `data`
- 'summary' is present in `data`
- 'tests' is present in `data`

Confidence: 80%

Tokens: 250 input + 92 output = 342 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	54 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_integration_gate.py::TestSchemaCompatibility::test_run_meta_has_aggregation_fields` 1ms 3

AI ASSESSMENT

Scenario: {'description': 'RunMeta has aggregation fields.', 'expected_result': {'schema': {'type': 'object', 'properties': {'is_aggregated': {'type': 'boolean'}, 'run_count': {'type': 'integer'}}}, 'assertions': [{'key_assertion': ['is_aggregated'], 'expected_result': True}, {'key_assertion': ['run_count'], 'expected_result': 0}]}}

Why Needed: The test is necessary because the RunMeta object does not have an 'aggregation_fields' property. The presence of this property in the schema indicates that aggregation fields are required for a RunMeta object to be valid.

Key Assertions:

- {'name': 'is_aggregated', 'expected_result': True}
- {'name': 'run_count', 'expected_result': 0}

Confidence: 80%

Tokens: 136 input + 246 output = 382 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	29 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_integration_gate.py::TestSchemaCompatibility::test_run_meta_has_status_fields` 1ms 3

AI ASSESSMENT

Scenario: Test 'RunMeta has run status fields' verifies that the RunMeta object contains status fields.

Why Needed: This test prevents a potential regression where the RunMeta object is missing required status fields.

Key Assertions:

- The 'exit_code' field should be present in the data.
- The 'interrupted' field should be present in the data.
- The 'collect_only' field should be present in the data.
- The 'collected_count' field should be present in the data.
- The 'selected_count' field should be present in the data.

Confidence: 80%

Tokens: 237 input + 135 output = 372 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	29 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_integration_gate.py::TestSchemaCompatibility::test_schema_version_defined` 1ms 2

AI ASSESSMENT

Scenario:
tests/test_integration_gate.py::TestSchemaCompatibility::test_schema_version_defined

Why Needed: The schema version is defined to ensure compatibility with gate APIs.

Key Assertions:

- {'name': 'SCHEMA_VERSION', 'type': 'string'}
- {'name': '.', 'type': 'boolean'}

Confidence: 80%

Tokens: 103 input + 87 output = 190 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

PASSED `tests/test_integration_gate.py::TestSchemaCompatibility::test_test_case_has_required_fields` 1ms 3

AI ASSESSMENT

Scenario: Test 'test_case_has_required_fields' verifies that the TestCaseResult object has required fields.

Why Needed: This test prevents a potential bug where a TestCaseResult object is created without all necessary fields (nodeid, outcome, duration).

Key Assertions:

- The 'nodeid' field should be present in the TestCaseResult object.
- The 'outcome' field should be present in the TestCaseResult object.
- The 'duration' field should be present in the TestCaseResult object.

Confidence: 80%

Tokens: 223 input + 116 output = 339 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	19 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_litellm_retry_coverage.py

4 tests

PASSED `tests/test_litellm_retry_coverage.py::TestLiteLLMTokenRefreshRetry::test_all_retries_exhausted` 2.00s 5

AI ASSESSMENT

Scenario: Test that all retries are exhausted when API calls fail.

Why Needed: Prevents regression where LiteLLMProvider fails to retry after exhausting all retries.

Key Assertions:

- The `result` variable should be `None`.
- The `error` attribute of the `result` object should not be `None`.
- The `test_source` field of the `result` object should be an empty string.
- The `context_files` dictionary of the `result` object should be an empty dictionary.
- The `provider.annotate()` method should raise an exception when API calls fail.

Confidence: 80%

Tokens: 346 input + 140 output = 486 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/litellm_provider.py	39 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 135, 139, 141-142, 144-145, 170-174, 176-178, 182, 186-187, 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_litellm_retry_coverage.py::TestLiteLLMTokenRefreshRetry::test_non_401_error_no_force_refresh` 1ms 5

AI ASSESSMENT

Scenario: Test that non-401 errors don't force token refresh.

Why Needed: Prevents regression in case of non-401 error without forcing token refresh.

Key Assertions:

- The API call should fail with a 500 status code.
- The annotation should contain an error message.
- The annotation should not have any retries set.
- The number of retries should be less than or equal to the maximum retries (1 in this case).
- No force refresh is triggered when the API call fails with a 500 status code.
- The test source does not contain any function that triggers token refresh.
- The context files are empty, which means no external dependencies are being refreshed.

Confidence: 80%

Tokens: 367 input + 159 output = 526 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/litellm_provider.py	38 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 135, 139, 141, 144-145, 170-174, 176-178, 182, 186-187, 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_litellm_retry_coverage.py::TestLiteLLMTokenRefreshRetry::test_retry_succeeds_after_transient_error` 6.00s 6

AI ASSESSMENT

Scenario: Test that retry succeeds after transient error.

Why Needed: To ensure the LLM can recover from transient errors and still complete successfully.

Key Assertions:

- The API call is mocked to fail twice, then succeed with a successful response.
- The test scenario is asserted as 'test scenario'.
- The error message is None after the retry attempt.
- The LLM completes successfully with a valid response.

Confidence: 80%

Tokens: 433 input + 102 output = 535 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	47 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 135, 139, 141-142, 170-174, 176-178, 182, 186-187, 190, 192-193, 196-201, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_litellm_retry_coverage.py::TestLiteLLMTokenRefreshRetry::test_token_refresh_on_401` 6.40s 7

AI ASSESSMENT

Scenario: Test that 401 error triggers token refresh when API call fails first, then succeeds.

Why Needed: To ensure the LLMTokenRefreshRetry test suite covers cases where the API call fails before a retry attempt.

Key Assertions:

- The test verifies that the LLM is retried after a 401 error has been encountered.
- The test verifies that the LLM will not be retried if the API call succeeds first.
- The test verifies that the LLMTokenRefreshRetry test suite catches and reports the retry attempt.
- The test verifies that the LLMTokenRefreshRetry test suite ensures the token is refreshed after a 401 error has been encountered.
- The test verifies that the LLMTokenRefreshRetry test suite prevents regression in API call handling.
- The test verifies that the LLMTokenRefreshRetry test suite handles 401 errors correctly and does not retry without a valid token.
- The test verifies that the LLMTokenRefreshRetry test suite reports the retry attempt to the user or logs an error message.

Confidence: 80%

Tokens: 473 input + 229 output = 702 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	54 lines (ranges: 37-38, 41-42, 44-48, 60-61, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 135, 139, 141-142, 170-174, 176-178, 182, 186-188, 190, 192-193, 196-201, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)

src/pytest_llm_report/llm/token_refresh.py	28 lines (ranges: 59-60, 63-66, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_llm.py

9 tests

PASSED `tests/test_llm.py::TestGetProvider::test_gemini_returns_provider` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm.py::TestGetProvider::test_gemini_returns_provider

Why Needed: The test is necessary because the Gemini model requires a specific provider to be used.

Key Assertions:

- {'name': 'provider.__class__.__name__', 'expected': 'GeminiProvider'}

Confidence: 80%

Tokens: 131 input + 80 output = 211 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	10 lines (ranges: 65-66, 384, 386, 388, 391, 396, 401-402, 404)
src/pytest_llm_report/llm/gemini.py	9 lines (ranges: 134-135, 137-141, 143-144)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm.py::TestGetProvider::test_litellm_returns_provider 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm.py::TestGetProvider::test_litellm_returns_provider

Why Needed: To ensure that the LiteLLMProvider class is correctly instantiated when a specific provider is used.

Key Assertions:

- {'name': 'provider.__class__.__name__', 'expected_value': 'LiteLLMProvider'}

Confidence: 80%

Tokens: 140 input + 86 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	9 lines (ranges: 65-66, 384, 386, 388, 391, 396-397, 399)
src/pytest_llm_report/llm/litellm_provider.py	3 lines (ranges: 37-38, 41)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm.py::TestGetProvider::test_none_returns_noop 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm.py::TestGetProvider::test_none_returns_noop

Why Needed: This test is necessary because the LLM's GetProvider method returns a NoopProvider when the provider is None.

Key Assertions:

- {'name': 'provider should be NoneType', 'description': 'The provider returned by the GetProvider method should be NoneType.'}
- {'name': 'provider should not be an instance of LLMProvider', 'description': 'The provider returned by the GetProvider method should not be an instance of LLMProvider.'}

Confidence: 80%

Tokens: 115 input + 138 output = 253 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	6 lines (ranges: 65-66, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm.py::TestGetProvider::test_ollama_returns_provider 1ms 4

AI ASSESSMENT

Scenario: tests/test_llm.py::TestGetProvider::test_ollama_returns_provider

Why Needed: To ensure that the OllamaProvider is correctly returned when a specific provider is specified.

Key Assertions:

- {'name': 'provider', 'expected_value': 'OllamaProvider'}

Confidence: 80%

Tokens: 154 input + 80 output = 234 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	8 lines (ranges: 65-66, 384, 386, 388, 391-392, 394)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm.py::TestGetProvider::test_unknown_raises 1ms 4

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	7 lines (ranges: 384, 386, 388, 391, 396, 401, 406)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm.py::TestLlmProviderContract::test_noop_implements_interface` 1ms 5

AI ASSESSMENT

Scenario: Test that NoopProvider implements LlmProvider contract.

Why Needed: Prevents a potential bug where the NoopProvider does not implement all required methods of LlmProvider.

Key Assertions:

- provider should have annotate method
- provider should have is_available method
- provider should have get_model_name method
- provider should have config attribute

Confidence: 80%

Tokens: 232 input + 90 output = 322 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm.py::TestNoopProvider::test_annotate_returns_empty 1ms 5

AI ASSESSMENT

Scenario: The test verifies that the NoopProvider returns an empty annotation when no annotation is specified.

Why Needed: This test prevents a regression where the NoopProvider does not return any annotation for a function with no annotations.

Key Assertions:

- annotation is of type LlmAnnotation
- annotation scenario is an empty string
- annotation why_needed is an empty string
- annotation key_assertions are an empty list

Confidence: 80%

Tokens: 249 input + 103 output = 352 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	8 lines (ranges: 65-66, 87-89, 97-98, 105)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 51)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm.py::TestNoopProvider::test_get_model_name_empty 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm.py::TestNoopProvider::test_get_model_name_empty

Why Needed: This test is needed because the model name is not being returned correctly when an empty string is passed to get_model_name.

Key Assertions:

- {'name': "assert get_model_name() == ''", 'expected_result': '', 'actual_result': 'NoopProvider.get_model_name()'}

Confidence: 80%

Tokens: 114 input + 100 output = 214 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 67)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm.py::TestNoopProvider::test_is_available 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm.py::TestNoopProvider::test_is_available

Why Needed: The LLM is not available.

Key Assertions:

- {'name': 'provider.is_available()', 'expected_value': True, 'actual_value': 'True'}

Confidence: 80%

Tokens: 108 input + 74 output = 182 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	5 lines (ranges: 65-66, 134, 137-138)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 59)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_llm_contract.py

13 tests

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_required_fields` 1ms 2

AI ASSESSMENT

Scenario: This test is designed to ensure that the `ANNOTATION_JSON_SCHEMA` correctly requires certain fields.

Why Needed: The purpose of this test is to verify that the schema does not allow optional fields and that required fields are present.

Key Assertions:

- {'name': 'required', 'description': "The field 'scenario' must be present in the annotation.", 'type': 'assertion'}
- {'name': 'why_needed', 'description': 'The schema does not allow optional fields, and required fields are present.', 'type': 'assertion'}

Confidence: 80%

Tokens: 115 input + 160 output = 275 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_schema_from_dict` 1ms 3

AI ASSESSMENT

Scenario: Test that AnnotationSchema.from_dict parses a dictionary correctly.

Why Needed: Prevents incorrect parsing of user data from a dict.

Key Assertions:

- checks password
- checks username

Confidence: 80%

Tokens: 274 input + 55 output = 329 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/schemas.py	5 lines (ranges: 77-81)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_schema_handles_empty` 1ms 3

AI ASSESSMENT

Scenario: This test checks if the AnnotationSchema can handle an empty input.

Why Needed: The test is necessary because the AnnotationSchema requires a non-empty string for the scenario and why-needeeds fields.

Key Assertions:

- {'name': 'schema.scenario', 'value': '', 'expected_value': ''}
- {'name': 'schema.why_needed', 'value': '', 'expected_value': ''}

Confidence: 80%

Tokens: 109 input + 108 output = 217 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/schemas.py	5 lines (ranges: 77-81)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm_contract.py::TestAnnotationSchema::test_schema_handles_partial 1ms 3

AI ASSESSMENT

Scenario: Test case for testing AnnotationSchema

Why Needed: This test is necessary to ensure the AnnotationSchema handles partial input correctly.

Key Assertions:

- {'assertion': "schema.scenario should be equal to 'Partial only'", 'expected_result': 'Partial only'}
- {'assertion': 'schema.why_needed should be empty', 'expected_result': ''}

Confidence: 80%

Tokens: 119 input + 98 output = 217 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/schemas.py	5 lines (ranges: 77-81)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_schema_has_required_fields` 1ms 2

AI ASSESSMENT

Scenario: The test verifies that the schema has required fields.

Why Needed: This test prevents a bug where the schema is missing required fields, potentially leading to errors or inconsistencies.

Key Assertions:

- assert 'scenario' in ANNOTATION_JSON_SCHEMA['properties']
- assert 'why_needed' in ANNOTATION_JSON_SCHEMA['properties']
- assert 'key_assertions' in ANNOTATION_JSON_SCHEMA['properties']
- assert isinstance(ANNOTATION_JSON_SCHEMA, dict)
- assert len(ANNOTATION_JSON_SCHEMA) > 0
- assert all(key in ANNOTATION_JSON_SCHEMA for key in ['scenario', 'why_needed', 'key_assertions'])

Confidence: 80%

Tokens: 215 input + 157 output = 372 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_schema_to_dict` 1ms 3

AI ASSESSMENT

Scenario: TestAnnotationSchema::test_schema_to_dict verifies that the AnnotationSchema instance correctly serializes to a dictionary.

Why Needed: This test prevents regression by ensuring that the AnnotationSchema instance can be serialized and deserialized correctly.

Key Assertions:

- assertion 1: The 'scenario' key in the data dictionary matches the expected value.
- assertion 2: The 'why_needed' key in the data dictionary matches the expected value.
- assertion 3: The 'key_assertions' list in the data dictionary contains all expected assertions.

Confidence: 80%

Tokens: 247 input + 129 output = 376 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 90-92, 94-96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm_contract.py::TestNoopProvider::test_noop_from_factory 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_contract.py::TestNoopProvider::test_noop_from_factory

Why Needed: The test is necessary to ensure that the factory returns a NoopProvider for provider='none'.

Key Assertions:

- {'name': 'provider', 'expected': 'None', 'got': 'None'}

Confidence: 80%

Tokens: 118 input + 86 output = 204 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	6 lines (ranges: 65-66, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_contract.py::TestNoopProvider::test_noop_is_llm_provider` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_contract.py::TestNoopProvider::test_noop_is_llm_provider

Why Needed: To ensure that the NoopProvider class correctly implements the LlmProvider interface.

Key Assertions:

- {'name': 'isinstance(provider, LlmProvider)', 'expected_result': 'True'}

Confidence: 80%

Tokens: 117 input + 84 output = 201 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_contract.py::TestNoopProvider::test_noop_returns_empty_annotation` 1ms 5

AI ASSESSMENT

Scenario: The NoopProvider returns an empty annotation when the test function does not have any annotations.

Why Needed: This test prevents a regression where the NoopProvider incorrectly returns an empty annotation for tests with no annotations.

Key Assertions:

- assert result.scenario == "" (empty string)
- assert result.why_needed == "" (empty string)
- assert result.key_assertions == [] (no key assertions performed)

Confidence: 80%

Tokens: 253 input + 104 output = 357 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	8 lines (ranges: 65-66, 87-89, 97-98, 105)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 51)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_contract.py::TestProviderContract::test_annotate_returns_annotation` 1ms 5

AI ASSESSMENT

Scenario: Verify that the `annotate` method returns a `TestCaseResult` object with the correct attributes.

Why Needed: This test prevents regression where the `annotate` method does not return an expected `TestCaseResult` object.

Key Assertions:

- The `result` attribute is present and has the correct name (`scenario`, `why_needed`, and `key_assertions`) on the `TestCaseResult` object.
- The `result` attribute has the correct type (e.g., a dictionary or an instance of `TestCaseResult`)
- The `outcome` attribute is set to `

Confidence: 80%

Tokens: 263 input + 138 output = 401 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	8 lines (ranges: 65-66, 87-89, 97-98, 105)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 51)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_contract.py::TestProviderContract::test_provider_handles_empty_code` 1ms 5

AI ASSESSMENT

Scenario: Provider handles empty code gracefully

Why Needed: Test case to ensure the provider can handle scenarios with empty code.

Key Assertions:

- {'description': 'The provider should return a valid TestCaseResult object even when the test has an empty code.', 'expected_result': "TestCaseResult(nodeid='test::nodeid', outcome='passed')", 'actual_result': {'nodeid': 'test::nodeid', 'outcome': 'passed'}}

Confidence: 80%

Tokens: 145 input + 111 output = 256 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	8 lines (ranges: 65-66, 87-89, 97-98, 105)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 51)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_contract.py::TestProviderContract::test_provider_handles_none_context` 1ms 5

AI ASSESSMENT

Scenario: Provider handles None context gracefully

Why Needed: To ensure the provider can handle None context without throwing an error.

Key Assertions:

- {'description': 'The annotate method should return a non-None result even when the test case has no outcome.', 'expected_result': 'True'}

Confidence: 80%

Tokens: 148 input + 78 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	8 lines (ranges: 65-66, 87-89, 97-98, 105)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 51)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_contract.py::TestProviderContract::test_provider_has_annotate_method` 1ms 7

AI ASSESSMENT

Scenario:
tests/test_llm_contract.py::TestProviderContract::test_provider_has_annotate_method

Why Needed: To ensure that all providers have an annotate method.

Key Assertions:

- {'name': "has attribute 'annotate'", 'expected_result': 'True'}
- {'name': 'is callable on provider.annotate', 'expected_result': 'True'}

Confidence: 80%

Tokens: 145 input + 96 output = 241 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	15 lines (ranges: 65-66, 384, 386, 388-389, 391-392, 394, 396-397, 399, 401-402, 404)
src/pytest_llm_report/llm/gemini.py	9 lines (ranges: 134-135, 137-141, 143-144)
src/pytest_llm_report/llm/litellm_provider.py	3 lines (ranges: 37-38, 41)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_llm_providers.py

52 tests

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_handles_context_too_large` 1ms 5

AI ASSESSMENT

Scenario:
tests/test_llm_providers.py::TestGeminiProvider::test_annotate_handles_context_too_large

Why Needed: Because the annotation process is too resource-intensive for large contexts.

Key Assertions:

- {'assertion_type': 'expected_exception', 'message': 'An exception was raised when annotating a context that was too large.'}

Confidence: 80%

Tokens: 98 input + 90 output = 188 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	187 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 263-265, 299, 311-312, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 435, 437-439, 441-444, 449-452, 463-466, 471-473, 476-478, 497-498, 502-505, 507-508, 511, 514-516, 518-521, 524-525, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_missing_dependency` 1ms 5

AI ASSESSMENT

Scenario: Test that the LiteLLMProvider annotates a missing dependency correctly.

Why Needed: This test prevents a bug where the provider does not report an error for missing dependencies.

Key Assertions:

- The annotation message contains the correct error message indicating that 'litellm' is missing and how to install it.
- The annotation message includes the correct path to install 'litellm'.
- The annotation message includes the correct provider name ('litellm') in the error message.

Confidence: 80%

Tokens: 270 input + 116 output = 386 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	34 lines (ranges: 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-195, 471-473, 497-498, 502-503, 537)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_missing_token` 1ms 5

AI ASSESSMENT

Scenario: Test that the `annotate` method of a GeminiProvider object raises an error when no API token is provided.

Why Needed: To prevent a potential bug where the `annotate` method fails to raise an error when an API token is missing, allowing the test case to pass even if the provider is not properly configured.

Key Assertions:

- The `error` attribute of the annotation object should be set to 'GEMINI_API_TOKEN is not set'.
- The `annotation.error` attribute should contain the string 'GEMINI_API_TOKEN is not set'.
- The `provider.annotate` method should raise an error with the message 'GEMINI_API_TOKEN is not set' when no API token is provided.
- The `annotation.error` attribute should be a string containing the message 'GEMINI_API_TOKEN is not set'.
- The `annotation.error` attribute should contain the exact phrase 'GEMINI_API_TOKEN is not set'.
- The `provider.annotate` method should raise an exception with the specified error message when no API token is provided.
- The `annotation.error` attribute should be a string that starts with 'GEMINI_API_TOKEN is not set'.
- The `annotation.error` attribute should contain the exact phrase 'GEMINI_API_TOKEN is not set' followed by a colon and a space.

Confidence: 80%

Tokens: 440 input + 295 output = 735 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	21 lines (ranges: 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-188)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)

src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_records_tokens` 1ms 6

AI ASSESSMENT

Scenario: Verify that the `annotate_records_tokens` test prevents regressions by ensuring tokens are recorded correctly.

Why Needed: To prevent regressions caused by a change in how token usage is handled.

Key Assertions:

- The `.annotate` method of the `GeminiProvider` instance records the specified number and type of tokens.
- The `annotate_records_tokens` test verifies that the `annotate` method does not raise an exception when called with a valid input.
- The rate limits logic is tested to ensure it runs without error.

Confidence: 80%

Tokens: 783 input + 124 output = 907 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	220 lines (ranges: 39-42, 45-46, 48, 52-54, 66, 68-70, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-101, 103, 105, 107-109, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246-247, 249-252, 261, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-430, 432, 435, 437-439, 441-444, 449-455, 457, 459-460, 463-466, 471-473, 476-478, 497-498, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552,

	554-555, 557-559, 562-563, 567, 569-571, 574)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_retries_on_rate_limit` 1ms 6

AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: To ensure that the LLM provider can correctly handle rate limiting and retry logic.

Key Assertions:

- {'name': 'Mocked API call with rate limit exceeded error', 'expected_output': {'error_code': 429, 'message': 'Rate limit exceeded'}, 'actual_output': {}}
- {'name': 'Mocked API call with retry logic executed', 'expected_output': {'retry_count': 1}, 'actual_output': {}}

Confidence: 80%

Tokens: 98 input + 128 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)

src/pytest_llm_report/llm/gemini.py	216 lines (ranges: 32-34, 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246, 249-250, 252, 261, 263-265, 299-300, 304-306, 308-309, 340-343, 346-349, 352, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413-416, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457-458, 463-466, 471-473, 476-478, 497-498, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_rotates_models_on_daily_limit` 1ms 6

AI ASSESSMENT

Scenario: tests/test_llm_providers.py::TestGeminiProvider::test_annotate_rotates_models_on_daily_limit

Why Needed: To ensure that the model rotation is applied correctly on a daily limit.

Key Assertions:

- {'name': 'model_rotation', 'expected_value': 'True'}
- {'name': 'daily_limit', 'expected_value': 1}

Confidence: 80%

Tokens: 100 input + 100 output = 200 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	210 lines (ranges: 39-42, 45-46, 48-50, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-230, 232, 235-236, 239-244, 246, 249-250, 252, 261, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457, 459, 461-466, 471-473, 476-478, 497-499, 502-505, 507-508, 511, 514-516, 518-521, 524, 526-527, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567-571, 574)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)

src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_skips_on_daily_limit` 1ms 6

AI ASSESSMENT

Scenario: tests/test_llm_providers.py::TestGeminiProvider::test_annotate_skips_on_daily_limit

Why Needed: Because the `annotate` method is skipping annotations on a daily limit.

Key Assertions:

- {'name': 'The annotate method should not skip any annotations when there are no annotations to skip.', 'expected_result': {}, 'actual_result': {}}
- {'name': 'The annotate method should skip all annotations when the daily limit is reached.', 'expected_result': {'annotations_skipped': []}, 'actual_result': {'annotations_skipped': ['annotation_1', 'annotation_2']}}

Confidence: 80%

Tokens: 98 input + 151 output = 249 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	47 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	216 lines (ranges: 39-42, 45-46, 48-50, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-230, 232-233, 235-236, 239-244, 246, 249-250, 252, 261, 318-320, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457, 459, 461-466, 471-473, 476-478, 497-499, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)

src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_success_with_mock_response` 1ms 6

AI ASSESSMENT

Scenario: Test that LiteLLM provider annotates a successful response with the correct key assertions and confidence level.

Why Needed: Prevents regression by ensuring the annotation is accurate for valid responses.

Key Assertions:

- status ok
- redirect

Confidence: 80%

Tokens: 474 input + 65 output = 539 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	209 lines (ranges: 39-42, 45-46, 48-49, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-101, 103, 105, 107-109, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246, 249-250, 252, 261, 340-343, 346-349, 352, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457-466, 471-473, 476-478, 497-498, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)

src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_exhausted_model_recovers_after_24h` 1ms 6

AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: The LLM provider's model should recover from being exhausted after 24 hours.

Key Assertions:

- {'name': 'model recovered within 24h', 'description': 'The model should be available and functional again after 24 hours.'}

Confidence: 80%

Tokens: 104 input + 81 output = 185 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	47 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	222 lines (ranges: 39-42, 45-46, 48-50, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-210, 212-213, 215-216, 218, 222-230, 232-233, 235-236, 239-244, 246, 249-250, 252, 261, 318-320, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457, 459, 461-466, 471-473, 476-478, 497-499, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)

src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_fetch_available_models_error` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: To ensure that the `fetch_available_models` method raises an error when no models are available.

Key Assertions:

- {'name': 'Expected a KeyError to be raised', 'description': 'When no models are available, the `fetch_available_models` method should raise a KeyError.'}

Confidence: 80%

Tokens: 92 input + 89 output = 181 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	68 lines (ranges: 134-135, 137-141, 143-144, 346, 348-349, 352-356, 358-361, 363-364, 366-367, 435, 437-439, 441-444, 449-452, 463-466, 476, 478, 497-498, 502-508, 511, 514-516, 518-521, 524-525, 537, 539-541, 544-545)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_model_list_refreshes_after_interval` 1ms 6

AI ASSESSMENT

Scenario: Tests for LLM providers

Why Needed: To ensure that the model list refreshes after a specified interval.

Key Assertions:

- {'name': 'Model list should be refreshed after interval', 'description': 'The model list should be updated with new data after the specified interval.'}

Confidence: 80%

Tokens: 96 input + 78 output = 174 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	201 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246, 249-250, 252, 261, 340-343, 346-349, 352, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457-458, 463-466, 471-473, 476-478, 497-499, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)

src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_401_retry_with_token_refresh` 1ms 7

AI ASSESSMENT

Scenario: Test that LiteLLM provider retries on 401 after refreshing token.

Why Needed: Reason: The current implementation does not retry when the token is refreshed, causing a failure in test_401_retry_with_token_refresh.

Key Assertions:

- Verify that the provider retries after refreshing the token.
- Verify that the first failed attempt is due to a 401 Unauthorized error.
- Verify that the second successful attempt is obtained with the refreshed token.
- Verify that the captured keys match the expected values (token-1 and token-2).
- Verify that the provider correctly sets the `litellm_token_refresh_command` and `litellm_token_refresh_interval` attributes.
- Verify that the `fake_completion` function is called with a valid API key for the first attempt, and an error for the second attempt.

Confidence: 80%

Tokens: 580 input + 186 output = 766 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	50 lines (ranges: 37-38, 41-42, 44-48, 60-61, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 122, 124-127, 170-174, 176-178, 182, 186-188, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/llm/token_refresh.py	28 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156, 160-162)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)

src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_handles_completion_error` 1ms 5

AI ASSESSMENT

Scenario: The test verifies that the LiteLLMProvider annotates completion errors correctly.

Why Needed: This test prevents a regression where the provider does not surface completion errors in annotations.

Key Assertions:

- The annotation should contain an error message indicating a completion error.
- The error message should contain the string 'boom'.
- The annotation should raise a RuntimeError exception when trying to annotate a test with a completion error.
- The provider should be able to surface completion errors in annotations by setting the litellm module in sys.modules.
- The config should be created with the correct provider (in this case, LiteLLMProvider).
- The LiteLLMProvider should create an instance of SimpleNamespace with a completion function set to fake_completion.
- The annotation should contain the correct key ('def test_case(): assert True').

Confidence: 80%

Tokens: 307 input + 188 output = 495 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/litellm_provider.py	34 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116, 120, 135, 137, 170-174, 176-178, 182, 186-187, 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_invalid_key_assertions` 1ms 6

AI ASSESSMENT

Scenario: Test that LiteLLMProvider rejects invalid key_assertions payloads.

Why Needed: To prevent regression and ensure the correct behavior of LiteLLMProvider when receiving invalid key_assertions payloads.

Key Assertions:

- The 'response_data' dictionary should be a list.
- The 'response_data' dictionary should contain only string keys.
- Invalid response: key_assertions must be a list
- The 'response_data' dictionary should not have any non-string keys.
- The 'response_data' dictionary should not have any non-string values.
-

Confidence: 80%

Tokens: 346 input + 135 output = 481 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	43 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346-348)
src/pytest_llm_report/llm/litellm_provider.py	35 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-187, 190, 192-193, 196, 204, 206, 211)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_missing_dependency` 1ms 5

AI ASSESSMENT

Scenario: The LiteLLMProvider annotates the missing dependency 'litellm' in the test case.

Why Needed: This test prevents a potential bug where the provider reports an error due to a missing required library, causing the test to fail or produce incorrect results.

Key Assertions:

- assert annotation.error == 'litellm not installed. Install with: pip install litellm'
- assert annotation.provider is None
- # The provider should be None when annotating a missing dependency

Confidence: 80%

Tokens: 271 input + 116 output = 387 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	9 lines (ranges: 65-66, 87-89, 97-99, 105)
src/pytest_llm_report/llm/litellm_provider.py	8 lines (ranges: 37-38, 41, 82-86)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_success_with_mock_response` 1ms 6

AI ASSESSMENT

Scenario: Test that the LiteLLMProvider annotates a valid response payload successfully.

Why Needed: Prevents regressions by ensuring the annotation is correct for successful responses.

Key Assertions:

- status ok
- redirect

Confidence: 80%

Tokens: 475 input + 61 output = 536 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	34 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-187, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_with_prompt_override 1ms 6

AI ASSESSMENT

Scenario: Test that LiteLLMProvider overrides the prompt when provided.

Why Needed: To ensure that the LiteLLM provider uses the custom prompt instead of the default one.

Key Assertions:

- The 'prompt_override' key in the annotation should be set to 'CUSTOM PROMPT'.
- The content of the 'messages' field in the captured messages should match the expected value.
- The error message should not have any issues or warnings.
- The custom prompt should override the default one used by LiteLLMProvider.
- The key 'why_needed' in the annotation should contain the expected message.
- The key 'key_assertions' in the annotation should contain the expected assertion.

Confidence: 80%

Tokens: 373 input + 159 output = 532 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	37 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	34 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95-96, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-187, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_with_token_usage` 1ms 6

AI ASSESSMENT

Scenario: Test LiteLLM provider to annotate with token usage.

Why Needed: Prevents regression in token usage extraction from responses.

Key Assertions:

- The `token_usage` attribute of the annotation returned by `provider.annotate(test, 'src')` is not None.
- The value of `prompt_tokens` in `annotation.token_usage` is set to 100.
- The value of `completion_tokens` in `annotation.token_usage` is set to 50.
- The total number of tokens extracted from the response is 150.

Confidence: 80%

Tokens: 426 input + 127 output = 553 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	39 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-187, 190, 192-193, 196-201, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_api_base_pass through` 1ms 6

AI ASSESSMENT

Scenario: Test that the LiteLLM provider passes `api_base` to completion call.

Why Needed: This test prevents regression in case where `api_base` is not set correctly.

Key Assertions:

- The value of `api_base` should be 'https://proxy.corp.com/v1'.
- The `litellm_api_base` attribute should have been updated with the correct value.
- The `api_base` key in the response data should match the expected value.
- The `key_assertions` list should contain only one assertion.
- The `response_data` dictionary should have a single key-value pair.
- The `json.dumps(response_data)` function should return a string representation of the dictionary.
- The `FakeLiteLLMResponse` class should create a response object with the expected data.
- The `litellm_api_base` attribute of the `FakeLiteLLMResponse` object should be set to 'https://proxy.corp.com/v1'.

Confidence: 80%

Tokens: 387 input + 222 output = 609 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	35 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182-183, 186-187, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_api_key_passthrough` 1ms 6

AI ASSESSMENT

Scenario: The test verifies that the LiteLLMProvider passes a static API key to the completion call.

Why Needed: This test prevents regression in passing an API key through the completion call, ensuring consistent behavior with previous tests.

Key Assertions:

- Verify that the captured API key matches the expected value.
- Check if the `litellm_api_key` attribute of the `fake_litellm` object is set to 'static-key-placeholder'.
- Verify that the `response_data` dictionary contains the expected keys ('scenario', 'why_needed', and 'key_assertions').
- Check if the `response_data` dictionary has a key named 'api_key' with value 'static-key-placeholder'.
- Verify that the `FakeLiteLLMResponse` object created by `fake_completion` returns a JSON response with the expected structure.
- Check if the captured API key is not set to an empty string or None.
- Verify that the `litellm_api_key` attribute of the `config` object has been updated correctly.
- Check if the `provider` object created by `LiteLLMProvider` has a valid `litellm_api_key` attribute.

Confidence: 80%

Tokens: 384 input + 260 output = 644 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	35 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-188, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)

src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_auth_error_without_refresher` 1ms 5

AI ASSESSMENT

Scenario: Test that the LiteLLM provider returns an authentication error when no refresher is configured.

Why Needed: This test prevents a bug where the provider does not raise an exception for authentication errors without token refresh.

Key Assertions:

- The function `fake_completion` raises a `FakeAuthError` exception with message '401 Unauthorized'.
- The error message returned by the LiteLLMProvider is 'Authentication failed'.
- The test case passes if the annotation of the provider contains an error message.
- The error message is not None.
- The error message contains the string 'Authentication failed'.

Confidence: 80%

Tokens: 338 input + 141 output = 479 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/litellm_provider.py	36 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 122, 132-133, 170-174, 176-178, 182, 186-187, 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_auth_retry_fails_on_second_attempt` 2.00s 6

AI ASSESSMENT

Scenario: Test that the LiteLLMProvider reports an authentication error when retrying after a second failure.

Why Needed: To prevent the test from passing if the LiteLLMProvider fails to report an authentication error on subsequent retries.

Key Assertions:

- The provider should raise an AuthenticationError with message 'Authentication failed' when the refresh operation fails.
- The provider should set the annotation.error attribute to a string containing 'Authentication failed'.
- The provider's completion function should not be called after a second failure.
- The provider's authentication error should be raised with a message of 'Authentication failed'.
- The provider's auth error should contain the token as an attribute.
- The provider's refresh command should return a non-zero exit code when the auth error is raised.
- The provider's llm_max_retries attribute should not be exceeded after a second failure.
- The test case should fail if the LiteLLMProvider does not raise an AuthenticationError on subsequent retries.

Confidence: 80%

Tokens: 419 input + 218 output = 637 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/litellm_provider.py	51 lines (ranges: 37-38, 41-42, 44-48, 60-61, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 122, 124-127, 129-130, 132-133, 141-142, 170-174, 176-178, 182, 186-188, 190)
src/pytest_llm_report/llm/token_refresh.py	31 lines (ranges: 59-60, 63-66, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156, 160-162)

src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_context_too_long_error` 1ms 6

AI ASSESSMENT

Scenario: The test verifies that the LiteLLMProvider class handles a context too long error correctly.

Why Needed: This test prevents a bug where the provider throws an exception when given an invalid response containing a key assertion with no value.

Key Assertions:

- assert annotation.error is not None
- assert 'Context too long for this model' in str(annotation)
- assert 'scenario' in str(annotation)
- assert 'why_needed' in str(annotation)
- assert 'key_assertions' in str(annotation)
- assert 'error' in str(annotation)
- assert 'json.dumps(...)' in str(annotation)

Confidence: 80%

Tokens: 370 input + 144 output = 514 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	16 lines (ranges: 65-66, 325-326, 329-330, 333-334, 337-339, 342, 344, 346-348)
src/pytest_llm_report/llm/litellm_provider.py	3 lines (ranges: 37-38, 41)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_get_max_context_tokens_dict_format` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py::TestLiteLLMProvider::test_get_max_context_tokens_dict_format

Why Needed: To ensure that the LiteLLM provider correctly handles dict format from get_max_tokens.

Key Assertions:

- {'expected_value': 16384, 'actual_value': 16384}

Confidence: 80%

Tokens: 218 input + 84 output = 302 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/litellm_provider.py	10 lines (ranges: 37-38, 41, 221-222, 224, 227-228, 230-231)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_get_max_context_tokens_fallback_on_error` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: To ensure the LLM provider returns a valid JSON response when an error occurs.

Key Assertions:

- {'name': 'Expected JSON response', 'value': {'scenario': 'tests/test_llm_providers.py', 'why_needed': 'To ensure the LLM provider returns a valid JSON response when an error occurs.'}}

Confidence: 80%

Tokens: 101 input + 98 output = 199 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/litellm_provider.py	10 lines (ranges: 37-38, 41, 221-222, 224, 227, 232-234)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_get_max_context_tokens_success` 1ms 5

AI ASSESSMENT

Scenario:
tests/test_llm_providers.py::TestLiteLLMProvider::test_get_max_context_tokens_success

Why Needed: To test the get_max_context_tokens method of LiteLLMProvider.

Key Assertions:

- {'name': 'expected return value', 'value': 8192, 'description': 'The expected return value of get_max_context_tokens should be 8192.'}

Confidence: 80%

Tokens: 213 input + 99 output = 312 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/litellm_provider.py	9 lines (ranges: 37-38, 41, 221-222, 224, 227-229)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_is_available_with_module` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py::TestLiteLLMProvider::test_is_available_with_module

Why Needed: To ensure the LiteLLM provider can detect installed modules.

Key Assertions:

- {'name': 'fake_litellm', 'expected_type': 'SimpleNamespace'}

Confidence: 80%

Tokens: 160 input + 78 output = 238 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	5 lines (ranges: 65-66, 134, 137-138)
src/pytest_llm_report/llm/litellm_provider.py	6 lines (ranges: 37-38, 41, 242-243, 245)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_token_refresh_integration` 1ms 7

AI ASSESSMENT

Scenario: Test the LiteLLMProvider's token refresh integration.

Why Needed: The test prevents a potential bug where the TokenRefresher is not able to refresh tokens for a long time, causing the LLM provider to fail to authenticate.

Key Assertions:

- Verify that the 'api_key' attribute of the case result matches the expected token value.
- Check if the 'why_needed' key in the test result contains the correct reason for the bug.
- Verify that the 'key_assertions' list in the test result includes the expected assertion.
- Check if the captured data from the subprocess is correctly updated with the fake completion function call.
- Verify that the subprocess returns a CompletedProcess object with returncode 0 and stdout 'dynamic-token-789'.
- Check if the subprocess returns an empty string for stderr.
- Verify that the config object passed to the provider has the correct values.
- Check if the provider is correctly set up to use the fake LitellmResponse object.

Confidence: 80%

Tokens: 442 input + 222 output = 664 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	41 lines (ranges: 37-38, 41-42, 44-48, 60-61, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-188, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/llm/token_refresh.py	25 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156)

src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_transient_error_retry` 1ms 6

AI ASSESSMENT

Scenario: The test verifies that the LiteLLMProvider retries transient errors and passes with the correct number of calls.

Why Needed: This test prevents a regression where the provider fails to retry on transient errors, potentially leading to unexpected behavior or failures in critical applications.

Key Assertions:

- Verify that the provider raises ConnectionError when it encounters a network error.
- Verify that the provider retries at least once before raising an exception.
- Verify that the provider does not raise an exception for non-network errors (e.g., authentication errors).
- Verify that the provider returns a successful response with a specific JSON payload after retrying transient errors.
- Verify that the test passes with the correct number of calls (3) when encountering 2 failures and 1 success.

Confidence: 80%

Tokens: 426 input + 171 output = 597 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	42 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 135, 139, 141-142, 170-174, 176-178, 182, 186-187, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_annotate_fallbacks_on_context_length_error` 1ms 7

AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: To ensure that the LLM provider correctly handles context length errors during annotation.

Key Assertions:

- {'assertion_type': 'Context Length Error', 'expected_result': 'A fallback strategy should be applied to handle context length errors.'}

Confidence: 80%

Tokens: 103 input + 79 output = 182 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	70 lines (ranges: 65-66, 87-89, 97-99, 101, 103, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 243, 245, 264, 266-267, 270-272, 274, 277, 279-280, 283, 286, 290-291, 294-295, 298-299, 305, 307-308, 312, 314, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/ollama.py	27 lines (ranges: 42-43, 49, 52, 55, 58, 60-61, 63-67, 71-72, 83, 85-86, 92, 138, 140, 142-144, 175-176, 178)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/llm/utils.py	28 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 96, 98)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_annotate_handles_call_error` 1ms 5

AI ASSESSMENT

Scenario: Test OllamaProvider::test_annotate_handles_call_error verifies that the annotate method handles call errors by returning an appropriate error message.

Why Needed: This test prevents a regression where the annotation fails with a generic 'Failed after X retries. Last error: ' when a call to Ollama raises an exception.

Key Assertions:

- The annotation should return an error message indicating that the annotation failed due to a call error.
- The error message should include the last error raised by Ollama.
- The error message should indicate that the annotation was unable to handle the call error within the specified retries.
- The error message should not be generic but specific to the call error.
- The annotation should raise an exception when a call to Ollama raises an exception.
- The annotation should log or report the call error in some way.

Confidence: 80%

Tokens: 347 input + 193 output = 540 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/ollama.py	18 lines (ranges: 42-43, 49, 52, 55, 58, 60-61, 63-65, 94, 97-98, 100-101, 103-104)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_annotate_missing_httpx` 1ms 5

AI ASSESSMENT

Scenario: The Ollama provider should report an error when the httpx dependency is missing.

Why Needed: This test prevents a bug where the provider incorrectly reports a missing dependency without providing any useful information.

Key Assertions:

- assert annotation.error == 'httpx not installed. Install with: pip install httpx'
- assert annotation.nodeid == 'tests/test_sample.py::test_case'
- assert annotation.outcome == 'passed'

Confidence: 80%

Tokens: 268 input + 105 output = 373 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	9 lines (ranges: 65-66, 87-89, 97-99, 105)
src/pytest_llm_report/llm/ollama.py	5 lines (ranges: 42-46)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_annotate_runtime_error_immediate_fail` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: To ensure that the LLM provider can correctly annotate runtime errors and immediately fail.

Key Assertions:

- {'name': 'LLM provider should return a JSON response with the expected key assertions', 'value': {'scenario': 'tests/test_llm_providers.py', 'why_needed': 'To ensure that the LLM provider can correctly annotate runtime errors and immediately fail.'}}
- {'name': 'JSON response should contain the correct keys', 'value': {'scenario': 'tests/test_llm_providers.py', 'why_needed': 'To ensure that the JSON response is correctly formatted and contains the expected keys.'}}

Confidence: 80%

Tokens: 99 input + 162 output = 261 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	22 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/ollama.py	13 lines (ranges: 42-43, 49, 52, 55, 58, 60-61, 63-65, 94, 96)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_annotate_success_full_flow` 1ms 6

AI ASSESSMENT

Scenario: Test the Ollama provider's full annotation flow with mocked HTTP responses.

Why Needed: Prevents authentication-related bugs in the Ollama provider.

Key Assertions:

- Verify that the response status code is 200 (OK) and contains a JSON object with the expected scenario, why-need, and key assertions.
- Check if the token is valid by validating its presence and correctness in the response.
- Ensure that the response includes the required 'response' field in the JSON structure.
- Verify that the 'scenario', 'why-need', and 'key_assertions' fields are present in the response as expected.
- Test that the provider correctly raises an exception for a 401 Unauthorized status code when attempting to login with invalid credentials.
- Check if the error message is properly formatted and contains relevant information about the authentication process.

Confidence: 80%

Tokens: 414 input + 190 output = 604 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/ollama.py	34 lines (ranges: 42-43, 49, 52, 55, 58, 60-61, 63-67, 71-72, 83, 92, 190, 192-200, 204-207, 209, 211-212)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_annotate_with_prompt_override` 1ms 6

AI ASSESSMENT

Scenario: Test that LiteLLM provider uses prompt_override when provided.

Why Needed: To ensure the correct behavior of the LiteLLM provider, where it overrides prompts with custom ones.

Key Assertions:

- The annotation should not contain any error messages.
- The custom prompt should be included in the output messages.
- The custom prompt should override the original prompt specified in the config.
- The custom prompt should be present in the LiteLLM response.
- The custom prompt should have a content that is different from the original prompt.
- The custom prompt should not cause any error messages to be generated.

Confidence: 80%

Tokens: 373 input + 141 output = 514 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	37 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/ollama.py	34 lines (ranges: 42-43, 49, 52-53, 58, 60-61, 63-67, 71-72, 83, 92, 190, 192-200, 204-207, 209, 211-212)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_annotate_with_token_usage` 1ms 6

AI ASSESSMENT

Scenario: Tests the `annotate` method of `LiteLLMProvider` with token usage data.

Why Needed: Prevents regression in handling token usage from LiteLLM responses.

Key Assertions:

- The `token_usage` attribute is populated with correct values.
- The `prompt_tokens` value matches the expected number of tokens (100).
- The `completion_tokens` value matches the expected number of tokens (50).
- The `total_tokens` value matches the expected total number of tokens (150).

Confidence: 80%

Tokens: 426 input + 123 output = 549 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/ollama.py	40 lines (ranges: 42-43, 49, 52, 55, 58, 60-61, 63-67, 71, 74-80, 83, 92, 190, 192-200, 204-207, 209, 211-212)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_call_ollama_success` 1ms 5

AI ASSESSMENT

Scenario: Test Ollama provider makes correct API call when calling OLLAMA successfully.

Why Needed: This test prevents regression where OLLAMA fails to make the API call with a valid response.

Key Assertions:

- The 'response' key in the result dictionary should be equal to 'test response'.
- The 'url' key in the captured dictionary should match the URL of the API call.
- The 'json' key in the captured dictionary should contain the expected JSON data.
- The 'model' key in the captured dictionary should be set to the correct model name.
- The 'prompt' key in the captured dictionary should be equal to the provided prompt.
- The 'system' key in the captured dictionary should be equal to the provided system prompt.
- The 'stream' key in the captured dictionary should be False (indicating no stream is being generated).
- The 'timeout' key in the captured dictionary should match the expected timeout value.

Confidence: 80%

Tokens: 470 input + 217 output = 687 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	17 lines (ranges: 190, 192-200, 204-207, 209, 211-212)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_call_ollama_uses_default_model` 1ms 5

AI ASSESSMENT

Scenario: Ollama provider uses default model when not specified.

Why Needed: This test prevents a regression where the Ollama provider defaults to the 'llama3.2' model even if no model is provided in the config.

Key Assertions:

- The captured JSON response contains the expected default model.
- The captured JSON response does not contain any custom model specified by the user.
- The captured JSON response has a 'model' key with value 'llama3.2'.
- The captured JSON response does not have a 'model' key or its value is different from 'llama3.2'.

Confidence: 80%

Tokens: 344 input + 145 output = 489 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	17 lines (ranges: 190, 192-200, 204-207, 209, 211-212)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_check_availability_failure` 1ms 5

AI ASSESSMENT

Scenario: TestOllamaProvider::test_check_availability_failure

Why Needed: The test checks if the Ollama provider correctly returns False when the server is unavailable.

Key Assertions:

- {'name': 'provider._check_availability() should return False', 'expected_value': False, 'actual_value': 'True'}

Confidence: 80%

Tokens: 183 input + 87 output = 270 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	6 lines (ranges: 113-114, 116-117, 119-120)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_check_availability_non_200` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py::TestOllamaProvider::test_check_availability_non_200

Why Needed: The test checks if the Ollama provider returns False for non-200 status codes.

Key Assertions:

- {'name': 'provider._check_availability() is False', 'expected_value': False, 'message': 'Expected provider._check_availability() to return False'}

Confidence: 80%

Tokens: 197 input + 104 output = 301 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	5 lines (ranges: 113-114, 116-118)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_check_availability_success` 1ms 5

AI ASSESSMENT

Scenario: The test verifies that the Ollama provider checks availability successfully by making a GET request to /api/tags.

Why Needed: This test prevents regression in case the API endpoint changes or is down, ensuring the Ollama provider can still function correctly.

Key Assertions:

- The response status code should be 200 (OK).
- The URL '/api/tags' should be present in the request URL.
- A valid HTTP response from the server should be received.

Confidence: 80%

Tokens: 296 input + 113 output = 409 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	5 lines (ranges: 113-114, 116-118)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_get_max_context_tokens_context_length_key` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: To ensure that the `get_max_context_tokens` method returns the correct context length key for a given scenario.

Key Assertions:

- {'name': 'context_length_key', 'expected_value': 'max_context_tokens'}

Confidence: 80%

Tokens: 99 input + 76 output = 175 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	16 lines (ranges: 138, 140, 142-147, 149-150, 156, 165-167, 172-173)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_get_max_context_tokens_fallback_on_error` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: To handle cases where the maximum context tokens are exceeded during training.

Key Assertions:

- {'message': 'The maximum number of context tokens is exceeded. This may be due to various reasons such as insufficient memory, excessive model complexity or incorrect hyperparameter settings.', 'expected_value': 1000}

Confidence: 80%

Tokens: 101 input + 92 output = 193 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	11 lines (ranges: 138, 140, 142-147, 175-176, 178)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_get_max_context_tokens_from_model_info` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: To ensure that the `get_max_context_tokens` method returns the correct number of context tokens for a given model info.

Key Assertions:

- {'name': 'max_context_tokens', 'expected_value': 32, 'actual_value': 0}

Confidence: 80%

Tokens: 99 input + 84 output = 183 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	16 lines (ranges: 138, 140, 142-147, 149-150, 156, 165-167, 172-173)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_get_max_context_tokens_from_parameters` 2ms 5

AI ASSESSMENT

Scenario: Tests for LLM providers

Why Needed: To ensure the correct number of context tokens is returned from the `get_max_context_tokens_from_parameters` method.

Key Assertions:

- {'name': 'max_context_tokens', 'expected_value': 32, 'actual_value': 0}
- {'name': 'context_token_type', 'expected_value': 'text', 'actual_value': 'token'}

Confidence: 80%

Tokens: 97 input + 97 output = 194 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	15 lines (ranges: 138, 140, 142-147, 149-150, 156, 158, 160-162)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_get_max_context_tokens_non_200_status` 1ms 5

AI ASSESSMENT

Scenario:
tests/test_llm_providers.py::TestOllamaProvider::test_get_max_context_tokens_non_200_status

Why Needed: This test is needed because the `get_max_context_tokens` method returns a status of 200 when there are less than 200 context tokens, but it should return an error message instead.

Key Assertions:

- {'name': 'status code', 'expected': 200, 'actual': 'None'}
- {'name': 'error message', 'expected': 'Error: max_context_tokens exceeded', 'actual': ''}

Confidence: 80%

Tokens: 101 input + 140 output = 241 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	10 lines (ranges: 138, 140, 142-147, 149, 178)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_is_local_returns_true` 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py::TestOllamaProvider::test_is_local_returns_true

Why Needed: To ensure the Ollama provider always returns `is_local=True`.

Key Assertions:

- {'name': 'provider is not None', 'expected_result': 'True'}
- {'name': 'provider.is_local() is True', 'expected_result': 'True'}

Confidence: 80%

Tokens: 123 input + 104 output = 227 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	1 lines (ranges: 128)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_invalid_json` 1ms 5

AI ASSESSMENT

Scenario: Ollama provider reports invalid JSON responses

Why Needed: To ensure the Ollama provider correctly handles and reports invalid JSON responses.

Key Assertions:

- {'name': 'annotation.error', 'value': 'Failed to parse LLM response as JSON'}

Confidence: 80%

Tokens: 138 input + 73 output = 211 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	7 lines (ranges: 65-66, 325-326, 329-331)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-52, 55)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_invalid_key_assertions` 1ms 5

AI ASSESSMENT

Scenario: {'description': 'Test case for Ollama provider when invalid key_assertions are provided in the response data.', 'expected_output': {'error': 'Invalid response: key_assertions must be a list'}}

Why Needed: The test is necessary to ensure that the Ollama provider correctly handles invalid key_assertions payloads in its responses.

Key Assertions:

- {'name': 'test_parse_response_invalid_key_assertions', 'description': 'Test case for Ollama provider when invalid key_assertions are provided in the response data.'}

Confidence: 80%

Tokens: 174 input + 209 output = 383 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	16 lines (ranges: 65-66, 325-326, 329-330, 333-334, 337-339, 342, 344, 346-348)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_code_fence` 1ms 5

AI ASSESSMENT

Scenario:
tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_code_fence

Why Needed: To ensure that the Ollama provider correctly extracts JSON from markdown code fences.

Key Assertions:

- {'name': 'JSON is present in response', 'expected': 'The response contains a valid JSON string.'}
- {'name': 'JSON is properly formatted', 'expected': 'The JSON string is properly formatted and consistent with the expected format.'}

Confidence: 80%

Tokens: 127 input + 121 output = 248 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	20 lines (ranges: 65-66, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/schemas.py	6 lines (ranges: 38, 42-44, 46-47)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_plain_fence` 1ms 5

AI ASSESSMENT

Scenario:
tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_plain_fence

Why Needed: to test the parsing of JSON responses from plain markdown fences (no language)

Key Assertions:

- {'name': 'response is a string', 'expected': 'str', 'actual': 'response'}
- {'name': 'response contains an opening brace', 'expected': 'True', 'actual': '
  '}

Confidence: 80%

Tokens: 128 input + 118 output = 246 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	20 lines (ranges: 65-66, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/schemas.py	6 lines (ranges: 38, 42-44, 46-47)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_success` 1ms 5

AI ASSESSMENT

Scenario: Test the Ollama provider's ability to parse valid JSON responses.

Why Needed: Prevents bugs in the LLM providers by ensuring that they correctly identify and extract relevant information from JSON responses.

Key Assertions:

- assert a is not None
- assert b is not None
- assert 'scenario' in annotation.scenario
- assert 'why_needed' in annotation.why_needed
- assert 'key_assertions' in annotation.key_assertions

Confidence: 80%

Tokens: 292 input + 113 output = 405 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	20 lines (ranges: 65-66, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_llm_utils.py

6 tests

PASSED tests/test_llm_utils.py::test_distribute_token_budget_constrained 1ms 3

AI ASSESSMENT

Scenario: Verify water-fill algorithm satisfies smaller files first.

Why Needed: This test prevents regression where the algorithm does not satisfy the constraint of distributing tokens to smaller files first due to insufficient budget.

Key Assertions:

- The total content allocated to `small.py` should be approximately equal to its required content plus overhead.
- The total content allocated to `large.py` should be greater than or equal to its required content plus overhead, considering the remaining budget and available files.
- Both `small.py` and `large.py` should have their expected allocations within a reasonable range of each other.

Confidence: 80%

Tokens: 396 input + 136 output = 532 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/utils.py	32 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 90-91, 93-94, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm_utils.py::test_distribute_token_budget_empty 1ms 3

AI ASSESSMENT

Scenario: tests/test_llm_utils.py::test_distribute_token_budget_empty

Why Needed: This test ensures that the `distribute_token_budget` function behaves correctly when given an empty input or no budget.

Key Assertions:

- {'assertion': {'message': 'Expected distribute_token_budget({}, 100) to return {}'}, 'expected_result': '{}'}
- {'assertion': {'message': "Expected distribute_token_budget({'f1': 'c'}, 0) to return {}"}, 'expected_result': '{}'}

Confidence: 80%

Tokens: 115 input + 130 output = 245 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/utils.py	2 lines (ranges: 42-43)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_utils.py::test_distribute_token_budget_fair_share` 1ms 3

AI ASSESSMENT

Scenario: Verify fair sharing when neither fits.

Why Needed: Prevents regression where either L1 or L2 file gets more than half of the budget, leading to unfair token distribution.

Key Assertions:

- The value of `allocations['l1.py']` should be between 35 and 50 (inclusive).
- The value of `allocations['l2.py']` should also be between 35 and 50 (inclusive).
- The absolute difference between the values of `allocations['l1.py']` and `allocations['l2.py']` should not exceed 1.
- Both `allocations['l1.py']` and `allocations['l2.py']` should be roughly equal.

Confidence: 80%

Tokens: 327 input + 170 output = 497 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/utils.py	30 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 90-91, 93-94, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_utils.py::test_distribute_token_budget_max_files` 1ms 3

AI ASSESSMENT

Scenario: tests/test_llm_utils.py::test_distribute_token_budget_max_files

Why Needed: Verify max_files limit.

Key Assertions:

- {'name': 'assert len(allocations) is equal to 3', 'expected_value': 3, 'actual_value': 0}
- {'name': 'assert allocations contains exactly three files', 'expected_value': 3, 'actual_value': []}

Confidence: 80%

Tokens: 133 input + 110 output = 243 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/utils.py	28 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm_utils.py::test_distribute_token_budget_sufficient 1ms 3

AI ASSESSMENT

Scenario: Verify that the `distribute_token_budget` function allocates tokens to files in a sufficient manner.

Why Needed: This test prevents a potential bug where the token budget is insufficient, leading to incomplete or corrupted file contents.

Key Assertions:

- The number of allocations for each file should be equal to the total needed tokens.
- Each allocation should contain exactly 10 tokens (i.e., ~40 characters).
- All files should have their full content allocated with sufficient token budget.

Confidence: 80%

Tokens: 332 input + 116 output = 448 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/utils.py	28 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_llm_utils.py::test_estimate_tokens` 1ms 3

AI ASSESSMENT

Scenario: Verify the rough token estimation (chars / 4) for an empty string.

Why Needed: Prevents a potential division by zero error when estimating tokens.

Key Assertions:

- assert estimate_tokens('') == 1
- assert estimate_tokens('a') == 1
- assert estimate_tokens('aaaa') == 1
- assert estimate_tokens('aaaa' * 10) == 10

Confidence: 80%

Tokens: 217 input + 103 output = 320 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/utils.py	1 lines (ranges: 20)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_models.py

29 tests

PASSED tests/test_models.py::TestArtifactEntry::test_to_dict 1ms 3

AI ASSESSMENT

Scenario: Test coverage entry serialization.

Why Needed: This test prevents a bug where the `CoverageEntry` object is not properly serialized to JSON.

Key Assertions:

- The 'file_path' key in the dictionary should match the expected value.
- The 'line_ranges' key in the dictionary should match the expected value.
- The 'line_count' key in the dictionary should match the expected value.
- The `to_dict()` method of the `CoverageEntry` object returns a dictionary with the correct keys and values.

Confidence: 80%

Tokens: 255 input + 123 output = 378 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	4 lines (ranges: 263-266)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestCollectionError::test_to_dict 1ms 3

AI ASSESSMENT

Scenario: Test that `CoverageEntry.to_dict()` returns the expected dictionary structure.

Why Needed: This test prevents a potential bug where the `CoverageEntry` object is not properly serialized to JSON.

Key Assertions:

- The `file_path` key should contain the correct file path.
- The `line_ranges` key should contain the correct line ranges in the format 'start-end'.
- The `line_count` key should contain the correct number of lines.
- Each assertion should match the expected values exactly.
- The dictionary structure should be consistent and follow the standard JSON format.

Confidence: 80%

Tokens: 255 input + 136 output = 391 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	3 lines (ranges: 241-243)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestCoverageEntry::test_to_dict` 1ms 3

AI ASSESSMENT

Scenario: Test coverage serialization for CoverageEntry.

Why Needed: This test prevents a potential bug where the coverage entry is not properly serialized to JSON.

Key Assertions:

- The 'file_path' key in the dictionary should be equal to 'src/foo.py'.
- The 'line_ranges' key in the dictionary should be equal to '1-3, 5, 10-15'.
- The 'line_count' key in the dictionary should be equal to 10.
- The 'file_path' value is not a string.
- The 'line_ranges' value is not a list or tuple of strings.
- The 'line_ranges' value contains non-string values (e.g., integers).
- The 'line_count' value is not an integer.

Confidence: 80%

Tokens: 255 input + 177 output = 432 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	4 lines (ranges: 65-68)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestLlmAnnotation::test_empty_annotation 1ms 2

AI ASSESSMENT

Scenario: An empty annotation should be created with default values.

Why Needed: This test prevents a regression where an empty annotation would result in a `NoneType` attribute.

Key Assertions:

- {'name': 'annotation.scenario', 'value': ''}
- {'name': 'annotation.why_needed', 'value': ''}
- {'name': 'annotation.key_assertions', 'value': []}
- {'name': 'annotation.confidence', 'value': 'None'}
- {'name': 'annotation.error', 'value': 'None'}

Confidence: 80%

Tokens: 212 input + 124 output = 336 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestLlmAnnotation::test_to_dict_minimal` 1ms 3

AI ASSESSMENT

Scenario: The test verifies that the `LlmAnnotation` object can be serialized into a dictionary with required fields.

Why Needed: This test prevents regression by ensuring that the minimal annotation is properly serialized without any optional fields.

Key Assertions:

- The 'scenario' field should be present in the dictionary.
- The 'why_needed' field should also be present in the dictionary.
- The 'key_assertions' field should not include the 'confidence' key, as it is an optional field.

Confidence: 80%

Tokens: 230 input + 117 output = 347 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	9 lines (ranges: 130-133, 135, 137, 139, 141, 143)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestLlmAnnotation::test_to_dict_with_all_fields` 1ms 3

AI ASSESSMENT

Scenario: Test to dictionary with all fields

Why Needed: Prevents incorrect data representation in API responses

Key Assertions:

- Assert scenario is set correctly
- Assert confidence value matches expected
- Assert context summary contains required mode and bytes properties
- Assert error is None or empty
- Assert token presence in response body is verified

Confidence: 80%

Tokens: 284 input + 87 output = 371 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	11 lines (ranges: 130-133, 135-137, 139-141, 143)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestReportRoot::test_default_report` 1ms 3

AI ASSESSMENT

Scenario: Test that the default report has a schema version and empty lists.

Why Needed: Prevents regression by ensuring the default report is correctly defined with required fields.

Key Assertions:

- The 'schema_version' field should be set to SCHEMA_VERSION.
- The 'tests' field should be an empty list.
- The 'warnings' field should not be included in the dictionary.
- The 'collection_errors' field should not be included in the dictionary.

Confidence: 80%

Tokens: 231 input + 111 output = 342 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	54 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestReportRoot::test_report_with_collection_errors` 1ms 3

AI ASSESSMENT

Scenario: Test Report Root with collection errors should be verified.

Why Needed: This test prevents a regression where the report does not include all collection errors.

Key Assertions:

- The length of `collection_errors` in the report should be 1.
- The nodeid of the first error in `collection_errors` should be 'test_bad.py'.
- All collection errors in `collection_errors` should have a valid nodeid.

Confidence: 80%

Tokens: 237 input + 104 output = 341 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	58 lines (ranges: 241-243, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526-528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestReportRoot::test_report_with_warnings 1ms 4

AI ASSESSMENT

Scenario: TestReportRoot::test_report_with_warnings

Why Needed: The test is necessary to ensure that the ReportWarning class correctly identifies and includes warnings in the report.

Key Assertions:

- {'assertion_type': 'length', 'expected_value': 1, 'actual_value': 1}
- {'assertion_type': 'code', 'expected_value': 'W001', 'actual_value': 'W001'}

Confidence: 80%

Tokens: 144 input + 112 output = 256 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	55 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestReportRoot::test_tests_sorted_by_nodeid 1ms 3

AI ASSESSMENT

Scenario: Tests should be sorted by nodeid in output.

Why Needed: Because the current implementation does not sort tests by nodeid, which can lead to incorrect test results.

Key Assertions:

- {'assertion': "nodeids == ['a_test.py::test_a', 'm_test.py::test_m', 'z_test.py::test_z']", 'expected_result': ['a_test.py::test_a', 'm_test.py::test_m', 'z_test.py::test_z']}

Confidence: 80%

Tokens: 215 input + 120 output = 335 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	73 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestReportWarning::test_to_dict_with_detail 1ms 3

AI ASSESSMENT

Scenario: The `to_dict()` method of the `ReportWarning` class is used to convert a warning object into a dictionary.

Why Needed: This test is needed because it checks if the detail attribute of the `ReportWarning` object is correctly populated in the returned dictionary.

Key Assertions:

- {'name': 'detail', 'expected': '/path/to/file', 'actual': '/path/to/file'}

Confidence: 80%

Tokens: 131 input + 116 output = 247 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	8 lines (ranges: 70-71, 73-75, 77-79)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestReportWarning::test_to_dict_without_detail` 1ms 3

AI ASSESSMENT

Scenario: Test to dictionary without detail should exclude it.

Why Needed: This test prevents a warning about missing detailed information in the report.

Key Assertions:

- The 'detail' key is expected to be present in the dictionary.
- The value of 'detail' is not provided in this case.
- The presence of 'detail' does not prevent the test from passing.

Confidence: 80%

Tokens: 223 input + 92 output = 315 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestRunMeta::test_aggregation_fields_present 1ms 3

AI ASSESSMENT

Scenario: Test that RunMeta has aggregation fields.

Why Needed: Prevents regression where RunMeta is not aggregated by default.

Key Assertions:

- assert d['run_id'] == 'run-123',
- assert d['run_group_id'] == 'group-456',
- assert d['is_aggregated'] is True,
- assert d['aggregation_policy'] == 'merge',
- assert d['run_count'] == 3,
- assert len(d['source_reports']) == 2

Confidence: 80%

Tokens: 343 input + 127 output = 470 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	39 lines (ranges: 286-288, 290-292, 376-392, 394, 397, 399, 402, 405, 407, 409, 411-417, 419, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestRunMeta::test_llm_fields_excluded_when_disabled` 1ms 3

AI ASSESSMENT

Scenario: Test that LLM fields are excluded when annotations are disabled.

Why Needed: This test prevents regression where LLMs are enabled but annotations are disabled, causing unexpected behavior.

Key Assertions:

- The 'llm_annotations_enabled' key should not be present in the data.
- The 'llm_provider' key should not be present in the data.
- The 'llm_model' key should not be present in the data.

Confidence: 80%

Tokens: 232 input + 105 output = 337 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	29 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestRunMeta::test_llm_traceability_fields` 1ms 3

AI ASSESSMENT

Scenario: Verify that LLM traceability fields are included when enabled.

Why Needed: Prevents regression in LLM model tracing functionality.

Key Assertions:

- data['llm_annotations_enabled'] is True
- data['llm_provider'] == 'ollama'
- data['llm_model'] == 'llama3.2:1b'
- data['llm_context_mode'] == 'complete'
- data['llm_annotations_count'] == 10
- data['llm_annotations_errors'] == 2

Confidence: 80%

Tokens: 327 input + 128 output = 455 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	43 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419-431, 433, 435, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestRunMeta::test_non_aggregated_excludes_source_reports` 1ms 3

AI ASSESSMENT

Scenario: tests/test_models.py::TestRunMeta::test_non_aggregated_excludes_source_reports

Why Needed: It's necessary to ensure that non-aggregated reports do not include source_reports.

Key Assertions:

- {'name': 'source_reports is not in d', 'expected_result': {'source_reports': []}}
- {'name': 'is_aggregated is False', 'expected_result': {'is_aggregated': False}}

Confidence: 80%

Tokens: 130 input + 111 output = 241 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	29 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestRunMeta::test_run_meta_to_dict_full` 1ms 3

AI ASSESSMENT

Scenario: Test RunMeta to dict with all optional fields.

Why Needed: Prevents regression in case of missing or outdated plugin version, as it would lead to incorrect data being populated.

Key Assertions:

- The 'git_sha' field is set to the expected value.
- The 'git_dirty' field is set to True.
- The 'repo_version' field matches the expected value.
- The 'repo_git_sha' field matches the expected value.
- The 'repo_git_dirty' field is False.
- The 'plugin_git_sha' field matches the expected value.
- The 'plugin_git_dirty' field is False.
- The 'config_hash' field matches the expected value.
- The length of the data dictionary is 1 as expected.

Confidence: 80%

Tokens: 483 input + 175 output = 658 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	49 lines (ranges: 286-288, 290-292, 376-392, 394-417, 419, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestRunMeta::test_run_status_fields 1ms 3

AI ASSESSMENT

Scenario: Test the RunMeta class to ensure it includes all necessary run status fields.

Why Needed: This test prevents a potential bug where the RunMeta object is missing certain critical fields that are required for proper functioning.

Key Assertions:

- The 'exit_code' field should be set to 1.
- The 'interrupted' field should be True.
- The 'collect_only' field should be True.
- The 'collected_count' field should be equal to 10.
- The 'selected_count' field should be equal to 8.
- The 'deselected_count' field should be equal to 2.

Confidence: 80%

Tokens: 285 input + 148 output = 433 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	29 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestSchemaVersion::test_schema_version_format 1ms 2

AI ASSESSMENT

Scenario: Test Schema Version Format

Why Needed: To ensure the schema version is in a valid semver format.

Key Assertions:

- {'name': 'The number of parts should be 3', 'expected_value': 3, 'message': "The schema version should have exactly 3 parts (e.g., '1.2.3')."}
- {'name': 'Each part should be a digit', 'expected_value': [0, 1, 2], 'message': 'Each part of the schema version should be a digit (0-9).'}

Confidence: 80%

Tokens: 115 input + 144 output = 259 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestSchemaVersion::test_schema_version_in_report_root` 1ms 3

AI ASSESSMENT

Scenario: tests/test_models.py::TestSchemaVersion::test_schema_version_in_report_root

Why Needed: This test is necessary because the ReportRoot class does not include a schema version by default.

Key Assertions:

- {'name': 'ReportRoot.schema_version', 'expected_value': 'SCHEMA_VERSION'}
- {'name': 'report.to_dict().schema_version', 'expected_value': 'SCHEMA_VERSION'}

Confidence: 80%

Tokens: 119 input + 107 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	54 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestSourceCoverageEntry::test_to_dict 1ms 3

AI ASSESSMENT

Scenario: Test coverage entry serialization.

Why Needed: This test prevents a bug where the `CoverageEntry` class does not properly serialize its internal data to JSON.

Key Assertions:

- The 'file_path' key in the serialized dictionary should match the original file path.
- The 'line_ranges' key in the serialized dictionary should match the expected range string.
- The 'line_count' key in the serialized dictionary should match the original value.
- All keys in the serialized dictionary should have unique values.
- Any missing keys in the serialized dictionary should raise an assertion error.

Confidence: 80%

Tokens: 256 input + 134 output = 390 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	8 lines (ranges: 96-103)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestSourceReport::test_to_dict_minimal 1ms 3

AI ASSESSMENT

Scenario: The test verifies that the `to_dict` method of `LlmAnnotation` returns a dictionary with required fields.

Why Needed: This test prevents a potential bug where the minimal annotation is missing some required fields.

Key Assertions:

- The 'scenario' field should be present in the dictionary.
- The 'why_needed' field should be present in the dictionary.
- The 'key_assertions' field should be present in the dictionary.
- The 'confidence' field should not be present in the dictionary when it is `None`.

Confidence: 80%

Tokens: 229 input + 128 output = 357 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	5 lines (ranges: 286-288, 290, 292)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestSourceReport::test_to_dict_with_run_id 1ms 3

AI ASSESSMENT

Scenario: tests/test_models.py::TestSourceReport::test_to_dict_with_run_id

Why Needed: To ensure that the SourceReport object is correctly serializing its run_id attribute.

Key Assertions:

- {'expected_value': 'run-1', 'actual_value': 'run-1'}

Confidence: 80%

Tokens: 134 input + 79 output = 213 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	6 lines (ranges: 286-288, 290-292)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestSummary::test_to_dict 1ms 3

AI ASSESSMENT

Scenario: Test that `CoverageEntry.to_dict()` correctly serializes the test summary.

Why Needed: This test prevents a potential bug where the serialized test summary is not accurate due to incorrect formatting of line ranges.

Key Assertions:

- The 'file_path' key in the dictionary should match the expected value.
- The 'line_ranges' key in the dictionary should match the expected value.
- The 'line_count' key in the dictionary should match the expected value.
- The 'start' and 'end' values of the line ranges should be correctly formatted (e.g., '1-3', '5, 10-15')
- Any missing or incorrect line ranges should raise an AssertionError
- Any invalid or malformed line ranges should raise an AssertionError

Confidence: 80%

Tokens: 254 input + 174 output = 428 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	11 lines (ranges: 467-475, 477, 479)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestTestCaseResult::test_minimal_result 1ms 3

AI ASSESSMENT

Scenario: Test that a minimal result has the required fields.

Why Needed: This test prevents regression where a minimal result is not provided with all necessary information.

Key Assertions:

- The 'nodeid' field should match the expected node ID.
- The 'outcome' field should be set to 'passed'.
- The 'duration' field should be set to 0.0 (indicating no execution time).
- The 'phase' field should be set to 'call'.

Confidence: 80%

Tokens: 244 input + 117 output = 361 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	19 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestTestCaseResult::test_result_with_coverage 1ms 3

AI ASSESSMENT

Scenario: tests/test_models.py::TestTestCaseResult::test_result_with_coverage verifies that the `result` dictionary contains a single 'coverage' key with a list of coverage entries.

Why Needed: This test prevents regression by ensuring that the `result` dictionary includes a 'coverage' key, which is necessary for calculating and displaying coverage statistics.

Key Assertions:

- The 'coverage' key should be present in the `result` dictionary.
- The 'coverage' key should contain a list of coverage entries.
- Each coverage entry should have a 'file_path' attribute set to the expected file path ('src/foo.py').
- Each coverage entry should have a 'line_ranges' attribute set to a valid range (e.g., '1-5') and a 'line_count' attribute equal to the actual number of lines in the file.
- The list of coverage entries should not be empty.
- All file paths in the 'coverage' list should match the expected file path ('src/foo.py').

Confidence: 80%

Tokens: 256 input + 223 output = 479 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	24 lines (ranges: 65-68, 190, 194-199, 201, 203, 205, 207, 210-212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestTestCaseResult::test_result_with_llm_opt_out` 1ms 3

AI ASSESSMENT

Scenario: tests/test_models.py::TestTestCaseResult::test_result_with_llm_opt_out

Why Needed: To ensure that the LLM opt-out flag is correctly set in the test result.

Key Assertions:

- {'name': 'assert llm_opt_out is True', 'expected_value': True, 'actual_value': 'is True'}

Confidence: 80%

Tokens: 145 input + 90 output = 235 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	20 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214-216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestTestCaseResult::test_result_with_rerun` 1ms 3

AI ASSESSMENT

Scenario: Test case 'test_result_with_rerun' has been executed.

Why Needed: The test result is not being recorded in the database because reruns are disabled.

Key Assertions:

- {'name': 'Rerun count', 'value': 2}
- {'name': 'Final outcome', 'value': 'passed'}

Confidence: 80%

Tokens: 162 input + 93 output = 255 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	21 lines (ranges: 190, 194-199, 201, 203, 205, 207-210, 212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models.py::TestTestCaseResult::test_result_without_rerun_excludes_fields` 1ms 3

AI ASSESSMENT

Scenario: tests/test_models.py::TestTestCaseResult::test_result_without_rerun_excludes_fields

Why Needed: This test ensures that the `result` dictionary excludes fields related to reruns.

Key Assertions:

- {'field_name': 'rerun_count', 'expected_value': 0, 'actual_value': 'Not applicable'}
- {'field_name': 'final_outcome', 'expected_value': 'passed', 'actual_value': 'passed'}

Confidence: 80%

Tokens: 152 input + 119 output = 271 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	19 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_models_coverage.py

15 tests

PASSED `tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_all_optional_fields` 1ms 4

AI ASSESSMENT

Scenario:
tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_all_optional_fields

Why Needed: Prevents bar because llm_opt_out=True prevents the annotation from being generated for optional fields.

Key Assertions:

- assert result['param_id'] == 'a-b-c',
- assert result['param_summary'] == 'a=1, b=2, c=3',
- assert result['captured_stdout'] == 'stdout content',
- assert result['captured_stderr'] == 'stderr content',
- assert result['requirements'] == ['REQ-100'],
- assert result['llm_opt_out'] is True,

Confidence: 80%

Tokens: 454 input + 157 output = 611 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	76 lines (ranges: 96-103, 241-243, 263-266, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526-540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_artifacts` 1ms 3

AI ASSESSMENT

LLM error: Failed to parse LLM response as JSON

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	59 lines (ranges: 263-266, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530-532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_collection_errors` 1ms 3

AI ASSESSMENT

Scenario: Test to_dict includes collection_errors when set.

Why Needed: This test prevents a regression where the to_dict method does not include collection_errors in the report.

Key Assertions:

- The length of collection_errors is 1.
- collection_errors[0].nodeid matches 'broken_test.py'.
- The nodeid value contains the string 'SyntaxError'.

Confidence: 80%

Tokens: 243 input + 92 output = 335 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	58 lines (ranges: 241-243, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526-528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_custom_metadata` 1ms 3

AI ASSESSMENT

Scenario: Test to_dict includes custom_metadata when set.

Why Needed: Prevents regression in cases where custom metadata is required but not properly handled by the default to_dict method.

Key Assertions:

- custom_metadata['project'] == 'myproject'
- custom_metadata['environment'] == 'staging'
- custom_metadata['build_number'] == 123
- custom_metadata does not contain any other metadata keys.
- result['custom_metadata']['project'] is equal to the expected value.
- result['custom_metadata']['environment'] is equal to the expected value.
- result['custom_metadata']['build_number'] is equal to the expected value.
- result does not contain any additional custom metadata keys.

Confidence: 80%

Tokens: 264 input + 162 output = 426 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	55 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534-536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_hmac_signature` 1ms 3

AI ASSESSMENT

Scenario: tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_hmac_signature

Why Needed: to ensure that the `report.to_dict()` method includes an HMAC signature when it is set.

Key Assertions:

- {'expected_value': 'signature123', 'actual_value': 'hmac_signature'}

Confidence: 80%

Tokens: 128 input + 85 output = 213 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/models.py`	`55 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538-540)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

PASSED `tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_sha256` 1ms 3

AI ASSESSMENT

Scenario: tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_sha256

Why Needed: The test is necessary because the `to_dict` method of the ReportRoot class includes a SHA-256 hash when it is set.

Key Assertions:

- {'name': 'expected value', 'value': 'abcdef1234567890'}

Confidence: 80%

Tokens: 131 input + 92 output = 223 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	55 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536-538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_source_coverage` 1ms 3

AI ASSESSMENT

Scenario: Test to_dict includes source_coverage when set.

Why Needed: Prevents a potential bug where the test fails if the 'source_coverage' key is not present in the report dictionary, potentially leading to incorrect coverage analysis.

Key Assertions:

- The 'source_coverage' list contains exactly one entry.
- The first element of the 'source_coverage' list has the correct file path ('src/mod.py').
- All elements in the 'source_coverage' list have the correct keys (file_path, statements, missed, covered, coverage_percent, covered_ranges, missed_ranges).

Confidence: 80%

Tokens: 282 input + 134 output = 416 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	63 lines (ranges: 96-103, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532-534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_warnings` 1ms 4

AI ASSESSMENT

Scenario: tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_warnings

Why Needed: This test is necessary because the `to_dict()` method of ReportRoot includes warnings when set.

Key Assertions:

- {'name': "len(result['warnings']) == 1", 'expected_value': 1, 'message': "Expected len(result['warnings']) to be equal to 1"}
- {'name': "result['warnings'][0]['code'] == 'W001'", 'expected_value': 'W001', 'message': "Expected result['warnings'][0]['code'] to be equal to 'W001'"}

Confidence: 80%

Tokens: 151 input + 157 output = 308 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	55 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestSummaryToDict::test_to_dict_with_coverage_total_percent` 1ms 3

AI ASSESSMENT

Scenario: ...

Why Needed: ...

Key Assertions:

- ...
- {'assertion_name': '...', 'expected_value': '...', 'actual_value': '...', 'message': '...'}

Confidence: 80%

Tokens: 153 input + 125 output = 278 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	12 lines (ranges: 467-475, 477-479)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models_coverage.py::TestSummaryToDict::test_to_dict_without_coverage_total_percent 1ms 3

AI ASSESSMENT

Scenario: ...

Why Needed: ...

Key Assertions:

- ...

Confidence: 80%

Tokens: 131 input + 267 output = 398 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	11 lines (ranges: 467-475, 477, 479)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_all_optional_fields` 1ms 3

AI ASSESSMENT

Scenario: Test to_dict includes all optional fields when set.

Why Needed: This test prevents regression in coverage calculation when llm_opt_out is True.

Key Assertions:

- The 'param_id' field should be present and match 'a-b-c'.
- The 'param_summary' field should contain the correct values for 'a', 'b', and 'c'.
- The 'captured_stdout' field should not be empty.
- The 'captured_stderr' field should not be empty.
- The 'requirements' list should include 'REQ-100'.
- The 'llm_opt_out' field should be True.
- The 'llm_context_override' field should match the scenario name.
- The number of coverage entries should be 1.
- The 'llm_annotation' field should contain the correct scenario name.

Confidence: 80%

Tokens: 454 input + 193 output = 647 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	42 lines (ranges: 65-68, 130-133, 135, 137, 139, 141, 143, 190, 194-199, 201-207, 210-224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_captured_stderr` 1ms 3

AI ASSESSMENT

Scenario: tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_captured_stderr

Why Needed: to include captured_stderr in the JSON response when to_dict is used.

Key Assertions:

- {'assertion_type': 'contains', 'expected_value': 'Error output here'}

Confidence: 80%

Tokens: 149 input + 80 output = 229 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	20 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220-222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_captured_stdout` 1ms 3

AI ASSESSMENT

Scenario: tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_captured_stdout

Why Needed: The `to_dict` method includes captured stdout when set.

Key Assertions:

- {'name': 'captured_stdout', 'expected_value': 'Debug output here'}

Confidence: 80%

Tokens: 149 input + 78 output = 227 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	20 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218-220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_param_summary` 1ms 3

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	21 lines (ranges: 190, 194-199, 201, 203-207, 210, 212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_requirements` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_requirements

Why Needed: The `to_dict` method includes requirements when set. This is necessary because the `requirements` key in the test result dictionary is not a standard JSON key, but rather an attribute of the TestCaseResult class.

Key Assertions:

- The `to_dict` method should include the `requirements` key and its value should be a list of strings.
- The `requirements` key should be present in the `test_result` dictionary.

Confidence: 80%

Tokens: 151 input + 124 output = 275 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	20 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222-224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_options.py

21 tests

PASSED tests/test_options.py::TestConfig::test_default_exclude_globs 1ms 3

AI ASSESSMENT

Scenario: Verify that the default exclude globs are correctly set.

Why Needed: This test prevents a potential bug where the default exclude globs are not correctly set, potentially leading to unexpected behavior or errors.

Key Assertions:

- The function `Config().llm_context_exclude_globs` returns a list of strings containing the specified globs.
- * The glob '*.pyc' is included in the list of default exclude globs.
- * The glob '__pycache__/*' is included in the list of default exclude globs.
- * The glob '*secret*' is included in the list of default exclude globs.
- * The glob '*password*' is included in the list of default exclude globs.

Confidence: 80%

Tokens: 222 input + 161 output = 383 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestConfig::test_default_redact_patterns 1ms 3

AI ASSESSMENT

Scenario: Tests the default redact patterns configuration.

Why Needed: Prevents a potential security vulnerability where sensitive information like passwords and tokens are not properly redacted.

Key Assertions:

- The `--password` pattern is present in the default redact patterns.
- The `--token` pattern is present in the default redact patterns.
- The `--api[_-]?key` pattern is present in the default redact patterns.

Confidence: 80%

Tokens: 228 input + 105 output = 333 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestConfig::test_default_values 1ms 3

AI ASSESSMENT

Scenario: Test that default values are set correctly for the test_default_values scenario.

Why Needed: This test prevents a potential regression where the default values of the Config class are not set properly, potentially leading to unexpected behavior or errors in the application.

Key Assertions:

- cfg.provider == 'none'
- cfg.llm_context_mode == 'minimal'
- cfg.llm_max_tests == 0
- cfg.llm_max_retries == 10
- cfg.llm_context_bytes == 32000
- cfg.llm_context_file_limit == 10
- cfg.llm_requests_per_minute == 5
- cfg.llm_timeout_seconds == 30
- cfg.llm_cache_ttl_seconds == 86400
- cfg.include_phase == 'run'
- cfg.aggregate_policy == 'latest'
- cfg.is_llm_enabled() is False
- cfg.omit_tests_from_coverage is True

Confidence: 80%

Tokens: 318 input + 209 output = 527 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestConfig::test_get_default_config 1ms 3

AI ASSESSMENT

Scenario: tests/test_options.py::TestConfig::test_get_default_config

Why Needed: To test the default configuration of the options.

Key Assertions:

- {'name': 'cfg', 'expected_type': 'Config'}
- {'name': 'cfg.provider', 'expected_value': 'none'}

Confidence: 80%

Tokens: 104 input + 84 output = 188 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 293)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestConfig::test_is_llm_enabled 1ms 3

AI ASSESSMENT

Scenario: Verify the is_llm_enabled check for different providers.

Why Needed: Prevents a potential bug where the test fails when using an unsupported provider.

Key Assertions:

- The function `is_llm_enabled` should return False for a provider without LLM (e.g., 'none')
- The function `is_llm_enabled` should return True for providers with LLMs (e.g., 'ollama', 'litellm', 'gemini')

Confidence: 80%

Tokens: 263 input + 112 output = 375 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options.py::TestConfig::test_validate_invalid_aggregate_policy` 1ms 3

AI ASSESSMENT

Scenario: test_validate_invalid_aggregate_policy

Why Needed: to test the validation of an invalid aggregation policy

Key Assertions:

- {'name': 'length of errors', 'value': 1, 'description': 'The number of errors found during validation'}
- {'name': 'error message', 'value': "Invalid aggregate_policy 'random'", 'description': 'The error message returned by the validation process'}

Confidence: 80%

Tokens: 128 input + 110 output = 238 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-221, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options.py::TestConfig::test_validate_invalid_context_mode` 1ms 3

AI ASSESSMENT

Scenario: tests/test_options.py::TestConfig::test_validate_invalid_context_mode

Why Needed: To ensure that the `validate()` method raises an error when a valid context mode is specified.

Key Assertions:

- {'assertion_type': 'length', 'expected_value': 1, 'actual_value': 0}

Confidence: 80%

Tokens: 131 input + 85 output = 216 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-213, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options.py::TestConfig::test_validate_invalid_include_phase` 1ms 3

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-229, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestConfig::test_validate_invalid_provider 1ms 3

AI ASSESSMENT

Scenario: Tests for configuration options

Why Needed: To ensure that the `Config` class correctly validates an invalid provider.

Key Assertions:

- {'message': 'The provider is invalid.', 'expected_value': "Invalid provider 'invalid_provider'"}

Confidence: 80%

Tokens: 122 input + 68 output = 190 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	28 lines (ranges: 123, 171, 199, 202-205, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestConfig::test_validate_numeric_ranges 1ms 3

AI ASSESSMENT

Scenario: Test validation of numeric constraints for TestConfig.

Why Needed: This test prevents a potential regression where the default values for LLM context bytes, max tests, requests per minute, timeout seconds, and max retries are not validated against their expected minimum or maximum values.

Key Assertions:

- cfg.validate() returns an empty list if all constraints are met
- cfg.validate() returns a list of errors with the specified message
- The 'llm_context_bytes' constraint is at least 1000 bytes
- The 'llm_max_tests' constraint is positive or zero
- The 'llm_requests_per_minute' constraint is at least 1
- The 'llm_timeout_seconds' constraint is at least 1
- The 'llm_max_retries' constraint is positive or zero

Confidence: 80%

Tokens: 329 input + 184 output = 513 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	31 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245-254, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestConfig::test_validate_valid_config 1ms 3

AI ASSESSMENT

Scenario: tests/test_options.py::TestConfig::test_validate_valid_config

Why Needed: To ensure that the `validate` method returns an empty list of errors when a valid configuration is passed.

Key Assertions:

- {'name': 'errors should be empty', 'expected_value': [], 'actual_value': 0}

Confidence: 80%

Tokens: 100 input + 85 output = 185 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	26 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestLoadConfig::test_load_aggregation_options 4ms 3

AI ASSESSMENT

Scenario: Test loads aggregation options with correct directory, policy and run ID.

Why Needed: This test prevents a bug where the aggregate options are not loaded correctly due to incorrect or missing values in the mock configuration.

Key Assertions:

- The expected value of `aggregate_dir` is set to `

Confidence: 80%

Tokens: 295 input + 205 output = 500 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	89 lines (ranges: 123, 171, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599-607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestLoadConfig::test_load_batch_flag_conflict 4ms 3

AI ASSESSMENT

Scenario: tests/test_options.py::TestLoadConfig::test_load_batch_flag_conflict

Why Needed: To test that the disabled batch flag works correctly.

Key Assertions:

- {'name': 'cfg.batch_parametrized_tests is True', 'expected_value': True, 'actual_value': 'assert cfg.batch_parametrized_tests is True'}

Confidence: 80%

Tokens: 138 input + 90 output = 228 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	85 lines (ranges: 123, 171, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options.py::TestLoadConfig::test_load_config_missing_pyproject` 4ms 3

AI ASSESSMENT

Scenario: Test handling when pyproject.toml doesn't exist.

Why Needed: Prevents regression in LLM configuration loading without a pyproject.toml file.

Key Assertions:

- The 'llm_max_retries' attribute is not set to its default value (10).
- The 'llm_report_html', 'llm_report_json', 'llm_report_pdf', 'llm_evidence_bundle', 'llm_dependency_snapshot', 'llm_requests_per_minute', 'llm_aggregate_dir', 'llm_aggregate_policy', 'llm_aggregate_run_id', 'llm_aggregate_group_id' attributes are set to None.
- The 'llm_provider' attribute is not set. This flag is required for LLMs.
- The 'llm_model' attribute is not set. This flag is required for LLMs.
- The 'llm_context_mode' attribute is not set. This flag is required for LLMs.
- The 'llm_prompt_tier', 'llm_batch_parametrized', and 'llm_context_compression' attributes are set to None, which may cause issues with the LLM configuration.
- }

Confidence: 80%

Tokens: 413 input + 260 output = 673 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	85 lines (ranges: 123, 171, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestLoadConfig::test_load_coverage_source 4ms 3

AI ASSESSMENT

Scenario: tests/test_options.py::TestLoadConfig::test_load_coverage_source

Why Needed: To test the coverage source option.

Key Assertions:

- {'name': 'mock_pytest_config.option.llm_coverage_source', 'expected_value': 'cov_dir'}

Confidence: 80%

Tokens: 126 input + 72 output = 198 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	86 lines (ranges: 123, 171, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607-608, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestLoadConfig::test_load_defaults 4ms 3

AI ASSESSMENT

Scenario: tests/test_options.py::TestLoadConfig::test_load_defaults

Why Needed: To test the default configuration when no options are set.

Key Assertions:

- {'name': "cfg.provider == 'none'", 'expected_value': 'None'}
- {'name': 'cfg.report_html is None', 'expected_value': 'None'}

Confidence: 80%

Tokens: 116 input + 93 output = 209 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	85 lines (ranges: 123, 171, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options.py::TestLoadConfig::test_load_from_cli_overrides_pyproject` 4ms 3

AI ASSESSMENT

Scenario: Test that CLI options override pyproject.toml options.

Why Needed: To test the ability of CLI options to override pyproject.toml settings.

Key Assertions:

- {'name': 'pyproject.toml overrides CLI options', 'expected_value': {'key': 'value'}, 'actual_value': {'key': 'override value'}}
- {'name': 'CLI options override pyproject.toml values', 'expected_value': {'key': 'override value'}, 'actual_value': {'key': 'override value'}}

Confidence: 80%

Tokens: 134 input + 131 output = 265 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	132 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482-484, 486, 488, 490, 492-494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options.py::TestLoadConfig::test_load_from_cli_provider_override` 5ms 3

AI ASSESSMENT

Scenario: Test that CLI provider option overrides pyproject.toml.

Why Needed: To ensure that the CLI provider option can override the default configuration in pyproject.toml, which is used to load dependencies.

Key Assertions:

- {'name': 'pyproject.toml', 'expected_content': '...toml'}
- {'name': 'CLI provider option overrides pyproject.toml', 'expected_content': '...overrides pyproject.toml'}

Confidence: 80%

Tokens: 130 input + 114 output = 244 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	133 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460-461, 463-464, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestLoadConfig::test_load_from_cli_retries 4ms 3

AI ASSESSMENT

Scenario: tests/test_options.py::TestLoadConfig::test_load_from_cli_retries

Why Needed: To test the functionality of loading retries from the CLI.

Key Assertions:

- {'name': 'mock_pytest_config.option.llm_max_retries', 'expected_value': 2, 'actual_value': 0}

Confidence: 80%

Tokens: 130 input + 86 output = 216 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	86 lines (ranges: 123, 171, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494-495, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options.py::TestLoadConfig::test_load_from_pyproject 5ms 3

AI ASSESSMENT

Scenario: Create a new directory with a pyproject.toml file

Why Needed: To test the functionality of loading values from a pyproject.toml file in an environment where the file does not exist.

Key Assertions:

- {'name': 'pyproject.toml exists', 'expected': True, 'actual': 'True'}

Confidence: 80%

Tokens: 119 input + 87 output = 206 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	134 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-336, 340, 342, 344, 348, 352, 356, 360-362, 364, 366, 368, 372, 374, 378, 380, 382-384, 386-388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options.py::TestLoadConfig::test_load_token_optimization_options` 4ms 3

AI ASSESSMENT

Scenario: Test loading token optimization options from CLI.

Why Needed: Prevents regression in token optimization configuration.

Key Assertions:

- The `prompt_tier` option should be set to 'minimal'.
- The `batch_parametrized_tests` option should not be enabled.
- The `context_compression` option should be set to 'none'.

Confidence: 80%

Tokens: 264 input + 89 output = 353 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	88 lines (ranges: 123, 171, 308, 311-312, 320-322, 460, 463, 466, 470-474, 476-477, 479, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_options_coverage.py

47 tests

PASSED `tests/test_options_coverage.py::TestCliOverrides::test_cli_dependency_snapshot` 5ms 3

AI ASSESSMENT

Scenario: Verify that the test_cli_dependency_snapshot function correctly sets the dependency snapshot to 'deps.json' when an option is set.

Why Needed: This test prevents a potential regression where the CLI overrides for dependency snapshots are not being properly applied.

Key Assertions:

- The value of `cfg.report_dependency_snapshot` should be set to `'deps.json'` after setting the `llm_dependency_snapshot` option.
- The function `load_config(mock)` correctly loads the mock configuration and returns a valid configuration object.
- The `mock.option.llm_dependency_snapshot` attribute is set to the expected value 'deps.json'.
- The `cfg.report_dependency_snapshot` attribute of the loaded configuration object is updated with the correct value 'deps.json'.
- The test function does not fail when an option is set for dependency snapshot.
- The test function passes without any assertion failures when an option is set for dependency snapshot.

Confidence: 80%

Tokens: 213 input + 203 output = 416 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	92 lines (ranges: 123, 171, 308, 311-312, 320-322, 460-461, 463-464, 466-467, 470-474, 476-477, 479, 482, 484, 486, 488, 490-492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestCliOverrides::test_cli_evidence_bundle` 6ms 3

AI ASSESSMENT

Scenario: Testing the `test_cli_evidence_bundle` function to ensure it correctly sets the `llm_evidence_bundle` option to 'bundle.zip'.

Why Needed: This test prevents a potential bug where the `llm_evidence_bundle` option is not set correctly, potentially leading to incorrect evidence bundle reporting.

Key Assertions:

- The value of `cfg.report_evidence_bundle` should be set to 'bundle.zip' after setting `mock.option.llm_evidence_bundle = 'bundle.zip'`.
- The function `load_config(mock)` should return the correct configuration with the updated option value.
- The assertion `assert cfg.report_evidence_bundle == 'bundle.zip'` should pass if the configuration is correctly loaded and the option is set to 'bundle.zip'.

Confidence: 80%

Tokens: 217 input + 175 output = 392 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	92 lines (ranges: 123, 171, 308, 311-312, 320-322, 460-461, 463-464, 466-467, 470-474, 476-477, 479, 482, 484, 486, 488-490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options_coverage.py::TestCliOverrides::test_cli_report_json 6ms 3

AI ASSESSMENT

Scenario: Verify that the `test_cli_report_json` test verifies that the `report_json` option is set to 'output.json' when CLI override for report JSON is enabled.

Why Needed: This test prevents a bug where the `report_json` option is not correctly overridden in the configuration, potentially leading to incorrect output.

Key Assertions:

- The value of `cfg.report_json` should be set to 'output.json' when `mock.option.llm_report_json = "output.json"`.
- The `load_config(mock)` function should return a valid configuration object with the correct `report_json` option value.
- The `assert cfg.report_json == "output.json"` statement should pass if the above conditions are met.
- If the `mock.option.llm_report_json = "other.json"` is executed, the test should fail and report an error message indicating that the `report_json` option was not overridden correctly.

Confidence: 80%

Tokens: 212 input + 207 output = 419 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	92 lines (ranges: 123, 171, 308, 311-312, 320-322, 460-461, 463-464, 466-467, 470-474, 476-477, 479, 482, 484-486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestCliOverrides::test_cli_report_pdf` 6ms 3

AI ASSESSMENT

Scenario: Tests the CLI option to generate a report in PDF format.

Why Needed: Prevents regression where the test fails due to incorrect report PDF path.

Key Assertions:

- The `llm_report_pdf` option is set to 'output.pdf' when the test runs.
- The `report_pdf` value of the configuration object matches 'output.pdf'.
- The `llm_report_pdf` option has been successfully overridden in the mock configuration.
- The report PDF path can be changed without affecting the test result.
- The `llm_report_pdf` option is correctly set even if the config file does not exist.
- The `report_pdf` value of the configuration object matches 'output.pdf' when the config file exists.
- The mock configuration has been successfully created with the correct report PDF path.

Confidence: 80%

Tokens: 212 input + 184 output = 396 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	92 lines (ranges: 123, 171, 308, 311-312, 320-322, 460-461, 463-464, 466-467, 470-474, 476-477, 479, 482, 484, 486-488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestConfigValidationCoverage::test_validate_invalid_token_output_format` 1ms 3

AI ASSESSMENT

Scenario: test_validate_invalid_token_output_format

Why Needed: To ensure that the token output format is correctly validated and raises an error when it's invalid.

Key Assertions:

- {'assertion': "errors contains 'litellm_token_output_format' key", 'description': 'The validation should return an error message indicating the invalid token output format.'}

Confidence: 80%

Tokens: 130 input + 90 output = 220 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-237, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestConfigValidationCoverage::test_validate_token_refresh_interval_too_short` 1ms 3

AI ASSESSMENT

Scenario: Test validation when token refresh interval is too short

Why Needed: Token refresh intervals should be at least 60 seconds to ensure sufficient time for the token to expire and be refreshed.

Key Assertions:

- {'message': 'litellm_token_refresh_interval must be at least 60', 'expected_value': 60}

Confidence: 80%

Tokens: 146 input + 86 output = 232 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	27 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241-242, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestConfigValidationCoverage::test_validate_valid_litellm_config` 1ms 3

AI ASSESSMENT

Scenario: Test validation of valid LiteLLM configuration.

Why Needed: To ensure that the LiteLLM provider is correctly configured and validated without any errors.

Key Assertions:

- {'assertion': 'The validate method returns an empty list of errors.', 'expected_result': [], 'message': 'Expected validate method to return an empty list of errors.'}

Confidence: 80%

Tokens: 142 input + 91 output = 233 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	26 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_aggregate_include_history` 2ms 3

AI ASSESSMENT

Scenario: test_load_aggregate_include_history

Why Needed: To ensure that the `aggregate_include_history` option is properly loaded and included in the generated code.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'include_history = True', 'actual': 'include_history = False'}

Confidence: 80%

Tokens: 118 input + 85 output = 203 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438-440, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_aggregate_policy_from_pyproject` 2ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_aggregate_policy_from_p

Why Needed: To ensure that the aggregate policy can be loaded from the PyProject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists and is not empty', 'expected_value': 'True'}
- {'name': 'aggregate_policy_path exists in pyproject.toml', 'expected_value': '/path/to/aggregate/policy.py'}

Confidence: 80%

Tokens: 121 input + 121 output = 242 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436-438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_all_config_keys_combined` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_all_config_keys_combined

Why Needed: To ensure that all config keys are loaded when loading the entire pyproject.toml file.

Key Assertions:

- {'name': 'All config keys should be present in the pyproject.toml file', 'expected_value': {'all_keys': ['config', 'keys', 'load', 'pyproject']}, 'actual_value': [False], 'message': 'Expected all config keys to be present, but got {falses}'}

Confidence: 80%

Tokens: 120 input + 136 output = 256 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	150 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-337, 340-346, 348-350, 352-354, 356-357, 360-369, 372-375, 378-392, 396, 400, 402, 404, 408-410, 412-413, 416-422, 426-428, 430-432, 436-440, 444-447, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_cache_dir` 1ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_cache_dir

Why Needed: To ensure that the cache directory is loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists and has a valid cache_dir', 'expected_value': 'path/to/cache/dir'}

Confidence: 80%

Tokens: 113 input + 89 output = 202 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390-392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_cache_ttl_seconds` 2ms 3

AI ASSESSMENT

Scenario: Tests for PyProjectLoadingCoverage

Why Needed: To ensure that the cache TTL seconds are loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists and is not empty', 'value': 'True'}
- {'name': "pyproject.toml has a section named 'cache_ttl_seconds'", 'value': 'True'}

Confidence: 80%

Tokens: 116 input + 103 output = 219 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388-390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_failed_output` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_failed_output

Why Needed: To ensure that the `capture_failed_output` option in `pyproject.toml` is correctly loaded and used for coverage purposes.

Key Assertions:

- {'name': 'pyproject.toml file exists', 'expected': 'The pyproject.toml file was created successfully.'}
- {'name': 'capture_failed_output key exists in pyproject.toml', 'expected': "The 'capture_failed_output' key was found in the pyproject.toml file."}

Confidence: 80%

Tokens: 116 input + 140 output = 256 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418-420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_output_max_chars` 2ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_output_max_cha

Why Needed: To ensure that the `capture_output_max_chars` option is properly loaded from the `pyproject.toml` file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'max_chars = 100', 'actual': 'None'}

Confidence: 80%

Tokens: 119 input + 99 output = 218 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420-422, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_bytes` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_bytes

Why Needed: To ensure that the context bytes are loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'context_bytes_path', 'expected_value': 'path/to/pyproject.toml', 'actual_value': 'pyproject / pyproject.toml'}
- {'name': 'context_bytes_file_mode', 'expected_value': 'r', 'actual_value': 'rb'}

Confidence: 80%

Tokens: 113 input + 127 output = 240 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362-364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_exclude_globs` 2ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_exclude_globs

Why Needed: To ensure that the `context_exclude_globs` setting in `pyproject.toml` is properly excluded from coverage reports.

Key Assertions:

- {'name': 'pyproject.toml file exists and is writable', 'expected': {'status': 'ok', 'message': ''}, 'actual': {'status': 'error', 'message': "PermissionError: 'tmp_path / ' is not writable"}}

Confidence: 80%

Tokens: 119 input + 127 output = 246 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368-369, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_file_limit` 2ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_file_limit

Why Needed: To ensure that the context file limit is correctly loaded from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists', 'expected_value': True, 'message': 'Expected pyproject.toml to exist'}
- {'name': 'pyproject.toml content is correct', 'expected_value': {'context_file_limit': 100}, 'message': 'Expected context_file_limit to be 100'}

Confidence: 80%

Tokens: 116 input + 138 output = 254 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364-366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_include_globs` 2ms 3

AI ASSESSMENT

Scenario: Tests for `tests/test_options_coverage`

Why Needed: To ensure that the `context_include_globs` setting is correctly loaded from the `pyproject.toml` file.

Key Assertions:

- {'name': 'Context include globs are loaded from pyproject.toml', 'description': 'The context include globs should be present in the test environment.', 'expected_value': 'True'}

Confidence: 80%

Tokens: 119 input + 103 output = 222 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366-368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_hmac_key_file` 1ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_hmac_key_file

Why Needed: To ensure that the hmac_key_file is loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists and contains hmac_key_file', 'expected': {'hmac_key_file': 'path/to/hmac_key_file'}, 'actual': {'hmac_key_file': 'path/to/hmac_key_file'}}
- {'name': 'pyproject.toml does not contain hmac_key_file', 'expected': {}, 'actual': {}}

Confidence: 80%

Tokens: 118 input + 146 output = 264 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446-447, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_param_values` 2ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_param_values

Why Needed: To ensure that the `include_param_values` option is correctly loaded from the `pyproject.toml` file.

Key Assertions:

- {'name': 'The `include_param_values` option should be present in the `pyproject.toml` file.', 'expected_value': True}
- {'name': 'The value of `include_param_values` should match the expected value.', 'expected_value': 'True'}

Confidence: 80%

Tokens: 116 input + 131 output = 247 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372-374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_phase` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_phase

Why Needed: To ensure that the `include_phase` is loaded correctly from the PyProject.toml file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': '```toml\n[tool.pyproject]\ninclude_phase = ["phase1", "phase2"]\n``', 'actual': '```toml\n[tool.pyproject]\ninclude_phase = []\n``', 'error_message': "The 'include_phase' key is missing from the PyProject.toml file."}

Confidence: 80%

Tokens: 113 input + 150 output = 263 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412-413, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_pytest_invocation` 2ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_pytest_invocatio

Why Needed: To ensure that the `include_pytest_invocation` option is correctly loaded from the PyProject.toml file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'include_pytest_invocation = True', 'actual': 'True'}

Confidence: 80%

Tokens: 122 input + 101 output = 223 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426-428, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_invocation_redact_patterns` 2ms 3

AI ASSESSMENT

Scenario: Tests for pyproject.toml coverage

Why Needed: To ensure that the invocation_redact_patterns are correctly loaded and used in tests.

Key Assertions:

- {'assertion_type': 'File existence', 'expected_result': 'pyproject.toml exists in the test directory', 'actual_result': 'pyproject.toml does not exist'}

Confidence: 80%

Tokens: 121 input + 91 output = 212 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430-432, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_api_base` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_api_base

Why Needed: To ensure that the litellm_api_base is loaded correctly from pyproject.toml.

Key Assertions:

- {'name': 'litellm_api_base exists in pyproject.toml', 'expected_value': 'True'}
- {'name': "pyproject.toml contains a [tool] section with 'litellm_api_base' as the name", 'expected_value': 'True'}

Confidence: 80%

Tokens: 122 input + 129 output = 251 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340-342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_api_key` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_api_key

Why Needed: To ensure that the litellm API key is correctly loaded from the pyproject.toml file.

Key Assertions:

- {'name': 'litellm_api_key exists in pyproject.toml', 'expected_value': 'litellm_api_key', 'actual_value': 'litellm_api_key'}

Confidence: 80%

Tokens: 122 input + 108 output = 230 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342-344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_token_json_key` 2ms 3

AI ASSESSMENT

Scenario: Loading litellm_token_json_key from pyproject.toml

Why Needed: To ensure that the litellm token JSON key is correctly loaded and used in the application.

Key Assertions:

- {'name': 'pyproject.toml file exists', 'description': 'The pyproject.toml file should exist at the specified path.', 'expected_result': 'True'}
- {'name': 'litellm_token_json_key is present in pyproject.toml', 'description': 'The litellm token JSON key should be present in the pyproject.toml file.', 'expected_result': 'True'}

Confidence: 80%

Tokens: 125 input + 151 output = 276 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342, 344, 348, 352, 356-357, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_token_output_format` 2ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_token_output_for

Why Needed: To ensure that the `litellm` package correctly handles token output formats in PyProject.toml.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': "```toml\noutput_format = 'litellm_token_output_format'\n``", 'actual': 'pyproject.toml contents'}

Confidence: 80%

Tokens: 125 input + 114 output = 239 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	111 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342, 344, 348, 352-354, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_token_refresh_command` 2ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_token_refresh_co

Why Needed: To ensure that the `litellm_token_refresh_command` is loaded correctly from the `pyproject.toml` file.

Key Assertions:

- {'name': 'pyproject.toml exists', 'description': 'The `pyproject.toml` file should exist in the test directory.', 'expected_result': 'True'}
- {'name': 'litellm_token_refresh_command is present in pyproject.toml', 'description': 'The `litellm_token_refresh_command` should be present in the `pyproject.toml` file.', 'expected_result': 'True'}

Confidence: 80%

Tokens: 125 input + 170 output = 295 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	111 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342, 344-346, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_token_refresh_interval` 3ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_token_refresh_int

Why Needed: To ensure that the `litellm_token_refresh_interval` option is properly loaded from the PyProject.toml file, allowing for accurate coverage analysis.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'a string representing the contents of the PyProject.toml file'}

Confidence: 80%

Tokens: 125 input + 107 output = 232 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	111 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342, 344, 348-350, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_malformed_pyproject 2ms 3

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	73 lines (ranges: 123, 171, 308, 311-312, 320-325, 449, 451, 453-456, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_max_concurrency` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_max_concurrency

Why Needed: To ensure that the `max_concurrency` setting in `pyproject.toml` is correctly loaded and used by the project.

Key Assertions:

- {'name': 'The `max_concurrency` setting in `pyproject.toml` should be a non-negative integer.', 'expected_value': 0, 'actual_value': 1}
- {'name': 'The `max_concurrency` setting in `pyproject.toml` should not cause any errors when loaded by the project.', 'expected_error': 'max_concurrency must be a non-negative integer', 'actual_error': ''}

Confidence: 80%

Tokens: 116 input + 169 output = 285 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380-382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_max_tests` 2ms 3

AI ASSESSMENT

Scenario: Testing the ability to load max_tests from pyproject.toml

Why Needed: To ensure that the 'max_tests' setting in the pyproject.toml file can be loaded correctly and used by the tests.

Key Assertions:

- {'name': "The 'max_tests' setting is present in pyproject.toml", 'value': 'True'}
- {'name': "The 'max_tests' setting is a boolean value", 'value': 3}

Confidence: 80%

Tokens: 113 input + 119 output = 232 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378-380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_metadata_file` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_metadata_file

Why Needed: To ensure that the metadata file is loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists', 'expected': 'True', 'actual': 'False'}
- {'name': 'metadata_file exists in pyproject.toml', 'expected': 'True', 'actual': 'False'}

Confidence: 80%

Tokens: 113 input + 118 output = 231 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444-446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_ollama_host` 2ms 3

AI ASSESSMENT

Scenario: test_pyproject_loading_coverage

Why Needed: To ensure that the ollama_host is loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml file exists and is not empty', 'expected_value': 'True'}
- {'name': 'ollama_host is defined in pyproject.toml', 'expected_value': 'True'}

Confidence: 80%

Tokens: 119 input + 104 output = 223 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336-337, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_omit_tests_from_coverage` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_omit_tests_from_coverag

Why Needed: To ensure that the `omit_tests_from_coverage` option is correctly loaded from the `pyproject.toml` file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': "The 'omit_tests_from_coverage' setting in pyproject.toml should be set to a boolean value."}

Confidence: 80%

Tokens: 121 input + 108 output = 229 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408-410, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_param_value_max_chars` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_param_value_max_chars

Why Needed: To ensure that the `param_value_max_chars` option is correctly loaded from the `pyproject.toml` file and that its value is being used in the build process.

Key Assertions:

- {'name': 'The value of `param_value_max_chars` is a string', 'expected_type': 'str'}
- {'name': 'The length of `param_value_max_chars` is less than or equal to 50 characters', 'expected_value': 50}

Confidence: 80%

Tokens: 119 input + 143 output = 262 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374-375, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_report_collect_only` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_report_collect_only

Why Needed: To ensure that the `report_collect_only` option is correctly loaded from the PyProject.toml file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'Collect only: ["..."]', 'actual': 'Collect only: ["..."]'}

Confidence: 80%

Tokens: 116 input + 103 output = 219 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416-418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_timeout_seconds` 2ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_timeout_seconds

Why Needed: To ensure that the timeout seconds are loaded correctly from pyproject.toml.

Key Assertions:

- {'name': 'pyproject.toml exists and is not empty', 'expected_value': 'True'}
- {'name': 'timeout_seconds key exists in pyproject.toml', 'expected_value': 'True'}

Confidence: 80%

Tokens: 113 input + 109 output = 222 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384-386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_batch_max_tests` 4ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_batch_max_tests

Why Needed: To ensure that the `batch_max_tests` option is correctly loaded from the PyProject.toml file and used to optimize Python packages.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': "The 'batch_max_tests' key in the PyProject.toml file should be set to a non-empty list of test names."}
- {'name': 'optimized_packages', 'expected': "The 'optimized_packages' key in the optimized package metadata should contain a list of test names that were batched together."}

Confidence: 80%

Tokens: 117 input + 156 output = 273 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	130 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400-402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_batch_parametrized_tests` 5ms 3

AI ASSESSMENT

Scenario: test_load_batch_parametrized_tests

Why Needed: Optimization of PyProject token in batch parameterized tests

Key Assertions:

- {'name': 'PyProject token is not empty', 'description': 'The PyProject token should be present in the pyproject.toml file.', 'value': 'True'}
- {'name': 'PyProject path exists', 'description': 'The PyProject path should exist and point to a valid directory.', 'value': 'True'}

Confidence: 80%

Tokens: 123 input + 125 output = 248 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	131 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396-398, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_context_compression` 4ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_context_compression

Why Needed: To ensure that the context compression feature in Pytest is properly loaded and used.

Key Assertions:

- {'name': 'Context compression should be enabled by default', 'value': True, 'expected_value': False}
- {'name': 'Context compression should be disabled if --with-optional-removal flag is used', 'value': True, 'expected_value': False}

Confidence: 80%

Tokens: 117 input + 127 output = 244 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	130 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402-404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_context_line_padding` 5ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_context_line_padding

Why Needed: To ensure that the context line padding is correctly loaded and applied in the build process.

Key Assertions:

- {'name': 'Context line padding is correctly loaded from pyproject.toml', 'expected_value': 'context_line_padding', 'actual_value': 'context_line_padding'}
- {'name': 'Context line padding is applied to the build output', 'expected_value': ['context_line_padding'], 'actual_value': []}

Confidence: 80%

Tokens: 117 input + 135 output = 252 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	130 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404-405, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_prompt_tier` 4ms 3

AI ASSESSMENT

Scenario:
tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_prompt_tier

Why Needed: To ensure that the `prompt_tier` is correctly loaded from the `pyproject.toml` file and used to determine the token optimization strategy.

Key Assertions:

- {'name': 'prompt_tier loading', 'expected': "The 'prompt_tier' key should be present in the pyproject.toml file."}
- {'name': 'token_optimization_strategy', 'expected': "The 'token_optimization_strategy' value should be a string that indicates the optimization strategy to use."}

Confidence: 80%

Tokens: 117 input + 148 output = 265 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	130 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392-393, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_batch_max_tests_too_small` 1ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_batch_max_tests_to

Why Needed: Because the `batch_max_tests` configuration option is not being used effectively.

Key Assertions:

- {'description': 'The validation error message should contain a clear indication that `batch_max_tests` must be at least 1.', 'expected_value': 'batch_max_tests must be at least 1', 'actual_value': 'None'}

Confidence: 80%

Tokens: 135 input + 113 output = 248 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	27 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271-273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_context_line_padding_negative 1ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_context_line_paddin

Why Needed: Negative context_line_padding is not allowed.

Key Assertions:

- {'message': 'context_line_padding must be 0 or positive'}

Confidence: 80%

Tokens: 129 input + 68 output = 197 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	27 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273-274, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_invalid_context_compression 1ms 3

AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_invalid_context_com

Why Needed: To ensure that the validation of context compression settings does not fail when an invalid value is provided.

Key Assertions:

- {'name': 'errors', 'type': 'list', 'value': ['Invalid context_compression']}

Confidence: 80%

Tokens: 124 input + 88 output = 212 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-269, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_invalid_prompt_tier` 1ms 3

AI ASSESSMENT

Scenario: test_validate_invalid_prompt_tier

Why Needed: To ensure that the `validate()` method correctly identifies and reports invalid `prompt_tier` values.

Key Assertions:

- {'expected': 'Invalid prompt_tier', 'actual': 'Invalid prompt_tier'}
- {'expected': 'Invalid prompt_tier', 'actual': 'Invalid prompt_tier'}

Confidence: 80%

Tokens: 125 input + 99 output = 224 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-261, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_plugin_integration.py

14 tests

PASSED `tests/test_plugin_integration.py::TestPluginConfigLoading::test_config_defaults` 3ms 3

AI ASSESSMENT

Scenario: tests/test_plugin_integration.py::TestPluginConfigLoading::test_config_defaults

Why Needed: To ensure that the plugin configuration has safe defaults.

Key Assertions:

- {'name': 'cfg is an instance of Config', 'expected_type': 'Config'}

Confidence: 80%

Tokens: 119 input + 72 output = 191 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	124 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-337, 340, 342, 344, 348, 352, 356, 360-362, 364, 366, 368, 372, 374, 378-380, 382, 384-386, 388, 390, 392, 396, 400, 402, 404, 408-410, 412-413, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460-461, 463-464, 466-467, 470, 472-473, 476-477, 482-488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603-605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_plugin_integration.py::TestPluginConfigLoading::test_markers_exist_in_config` 1ms 2

AI ASSESSMENT

Scenario:
tests/test_plugin_integration.py::TestPluginConfigLoading::test_markers_exist_in_config

Why Needed: The test checks if markers exist in the plugin configuration.

Key Assertions:

- {'name': 'pytestconfig is not None', 'expected_value': 'True'}

Confidence: 80%

Tokens: 108 input + 74 output = 182 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_both_json_and_html_outputs` 102ms 8

AI ASSESSMENT

Scenario: Test generates both JSON and HTML reports for a test function.

Why Needed: This test prevents regression in cases where the plugin is used with both JSON and HTML output formats.

Key Assertions:

- The generated report files should exist at `report.json` and `report.html` paths.
- The report data should be correctly formatted as either JSON or HTML depending on the configuration.
- The test function should produce a valid output that can be parsed by both Pytester's reporting tools.
- The plugin's integration with Pytester should not introduce any new bugs or regressions in this scenario.
- The generated report files should have the correct file extensions (JSON and HTML) even when using the `--llm-report` option.
- The test function should not fail to run due to a missing or incorrect report file path.

Confidence: 80%

Tokens: 279 input + 186 output = 465 total

COVERAGE

src/pytest_llm_report/collector.py	39 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	75 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	91 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322,

	460, 463, 466, 470, 472-473, 476-477, 482-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	122 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

PASSED tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_collection_finish_counts_items 66ms 7

AI ASSESSMENT

Scenario: tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_collection_finish_counts_items

Why Needed: pytest_collection_finish counts items (line 378)

Key Assertions:

- {'name': "assert data['run_meta']['collected_count'] == 3", 'expected_value': 3, 'message': 'Expected collected count to be 3'}

Confidence: 80%

Tokens: 198 input + 96 output = 294 total

COVERAGE

src/pytest_llm_report/collector.py	39 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	75 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)

src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	110 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_creates_nested_directory` 63ms 7

AI ASSESSMENT

Scenario: The test verifies that the `test_pass()` function is executed and a new directory 'nested' with 'dir' as its parent directory is created.

Why Needed: This test prevents regression in cases where the plugin integration fails to create output directories.

Key Assertions:

- The 'nested' directory should be created with 'dir' as its parent directory.
- The 'report.json' file within the 'nested' directory should exist.
- The 'test_pass()' function should be executed successfully.
- The 'pytester.path / nested / dir / report.json' path should be an existing directory.
- The 'pytester.runpytest(f--llm-report-json={report_path})' command should execute without raising any errors.

Confidence: 80%

Tokens: 247 input + 167 output = 414 total

COVERAGE

src/pytest_llm_report/collector.py	39 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	11 lines (ranges: 70-71, 73-75, 77, 79, 142-145)
src/pytest_llm_report/models.py	76 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-

	505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	116 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-484, 502, 504, 506-508, 510, 513)

PASSED `tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_fixture_error_captured` 67ms 7

AI ASSESSMENT

Scenario: Test that fixture errors are captured in report.

Why Needed: Fixture failures are not properly reported, leading to incorrect error counts and debugging issues.

Key Assertions:

- The 'summary' key in the report contains an error code of 1.
- The 'error' value under the 'summary' key is set to a non-zero value (in this case, 1).
- The test fixture raises a RuntimeError and its name is included in the error summary.

Confidence: 80%

Tokens: 286 input + 113 output = 399 total

COVERAGE

src/pytest_llm_report/collector.py	50 lines (ranges: 78-79, 90, 93-94, 96, 99-103, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 227-228, 230-236, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	76 lines (ranges: 190, 194-199, 201-203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-

	594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	115 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324, 326, 328, 330, 332, 334-335, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_makereport_captures_all_outcomes` 179ms 7

AI ASSESSMENT

Scenario: Test pytest_runtest_makereport captures outcomes to verify that it correctly identifies all test outcomes.

Why Needed: pytest_runtest_makereport prevents regression by ensuring that the plugin correctly identifies and reports all test outcomes, including skipped tests.

Key Assertions:

- The 'passed' outcome should be present in the report.
- The 'failed' outcome should be present in the report.
- The 'skipped' outcome should be present in the report.
- All test names should be included in the report.
- The plugin should not miss any test outcomes, including skipped tests.
- The plugin should correctly identify and report all test outcomes, regardless of their status (e.g., running, pending).

Confidence: 80%

Tokens: 335 input + 165 output = 500 total

COVERAGE

src/pytest_llm_report/collector.py	59 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 106-107, 109-112, 114-118, 124, 127, 132-133, 140-141, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 227-228, 230-236, 250-251, 261, 264, 268, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	76 lines (ranges: 190, 194-199, 201-203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473,

	476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	114 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-329, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_no_report_when_disabled` 62ms 3

AI ASSESSMENT

Scenario:
tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_no_report_when_disabled

Why Needed: To ensure that the plugin correctly handles cases where no output is specified.

Key Assertions:

- {'name': 'No report generated when run without a file path', 'expected': 'The report.json file should not exist'}

Confidence: 80%

Tokens: 150 input + 88 output = 238 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	89 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	250 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373,

399, 403-404, 558-559, 562-563, 566-568, 579, 583, 602-603, 619-620)

PASSED `tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_pdf_option_enables_plugin` 629ms 8

AI ASSESSMENT

Scenario: Test that the `--llm-pdf` option enables the plugin and triggers its logic.

Why Needed: Prevents regression in plugin integration where --llm-pdf is used without enabling the plugin.

Key Assertions:

- assert True, 'Expected test_pass() to be called with no arguments.'
- assert result.ret == 0, 'Expected pytester.runpytest('--llm-pdf=report.pdf') to exit with code 0 (success) or warning.'

Confidence: 80%

Tokens: 435 input + 113 output = 548 total

COVERAGE

src/pytest_llm_report/collector.py	39 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486-488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138,

	142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	110 lines (ranges: 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226, 230-231, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 408, 417, 419, 421-423, 431-436, 439, 441-442, 455, 460, 462, 465-469, 477-478)

PASSED `tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_session_start_records_time` 65ms 7

AI ASSESSMENT

Scenario: Test that pytest_sessionstart records start time is verified by Pytester.

Why Needed: This test prevents a potential bug where the start time of the session is not recorded correctly.

Key Assertions:

- The 'start_time' key should be present in the run_meta dictionary.
- The value of the 'start_time' key should be a valid timestamp.
- The 'start_time' value should be greater than or equal to 0.
- The start time should not be None.
- The start time should be within the expected range (e.g., between 2023-01-01 00:00:00 and 2024-01-01 23:59:59).
- The 'start_time' value should match the actual start time of the session recorded by pytest_sessionstart.
- The test should fail if the start time is not within the expected range or is None.

Confidence: 80%

Tokens: 276 input + 205 output = 481 total

COVERAGE

src/pytest_llm_report/collector.py	39 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	75 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473,

	476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	110 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_plugin_integration.py::TestPluginIntegration::test_llm_context_marker` 1ms 2

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_plugin_integration.py::TestPluginIntegration::test_llm_opt_out_marker` 1ms 2

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_plugin_integration.py::TestPluginIntegration::test_requirement_marker` 1ms 2

AI ASSESSMENT

Scenario: tests/test_plugin_integration.py

Why Needed: The requirement marker is used to mark requirements as having a plugin. This helps in identifying which requirements require plugins and can be useful for testing purposes.

Key Assertions:

- {'assertion': 'Requirement marker should not cause errors.', 'expected_result': 'True', 'message': ''}

Confidence: 80%

Tokens: 90 input + 88 output = 178 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_plugin_integration.py::TestReportGeneration::test_report_writer_integration` 48ms 6

AI ASSESSMENT

Scenario: Test the integration of report writer with pytest_llm_report.

Why Needed: This test prevents regression when integrating report writer with pytest_llm_report, as it ensures that all tests are properly formatted and include required information for a full report.

Key Assertions:

- The report JSON file exists at `report.json` in the specified temporary directory.
- The total count of passed tests is 1 (test_a.py::test_pass) out of 2 (total)
- All test names are included in the report HTML, specifically including `test_a.py` and `test_b.py`.
- The report HTML file exists at `report.html` in the specified temporary directory.
- The report HTML contains a reference to each test name (`test_a.py` and `test_b.py`).

Confidence: 80%

Tokens: 417 input + 179 output = 596 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	81 lines (ranges: 190, 194-199, 201-203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	136 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303,

	305-307, 319, 321-322, 324-327, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

📄 tests/test_plugin_maximal.py

26 tests

PASSED `tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_disabled` 1ms 2

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	10 lines (ranges: 558-559, 562, 566-568, 579-580, 586-587)

PASSED `tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_enabled` 2ms 2

AI ASSESSMENT

Scenario: TestPluginCollectReport

Why Needed: To test the collectreport functionality when it is enabled.

Key Assertions:

- {'name': 'mock_collector.handle_collection_report was called once with mock_report', 'expected': 1}

Confidence: 80%

Tokens: 204 input + 68 output = 272 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	12 lines (ranges: 558-559, 562, 566-568, 579-580, 586, 590-592)

PASSED `tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_no_session` 1ms 2

AI ASSESSMENT

Scenario: tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_no_session

Why Needed: To ensure that collectreport does not throw an exception when a session is not available.

Key Assertions:

- {'name': 'mock_report.session', 'expected_value': 'None'}

Confidence: 80%

Tokens: 138 input + 81 output = 219 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	8 lines (ranges: 558-559, 562, 566-568, 579, 583)

PASSED `tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_session_none` 1ms 2

AI ASSESSMENT

Scenario: tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_session_none

Why Needed: To ensure that the collectreport plugin behaves correctly when a Pytest session is None.

Key Assertions:

- {'name': 'pytest_collectreport() should not be called with a mock report object that has a None session attribute', 'description': 'The pytest_collectreport function should not be called with a mock report object that has a None session attribute.'}

Confidence: 80%

Tokens: 134 input + 115 output = 249 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	8 lines (ranges: 558-559, 562, 566-568, 579, 583)

PASSED `tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_llm_enabled_warning` 3ms 3

AI ASSESSMENT

Scenario: tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_llm_enabled_warning

Why Needed: LLM enabled warning is raised when pytest is run with the --llm flag.

Key Assertions:

- {'name': 'LLM enabled flag is present in pyproject.toml', 'value': 'True'}
- {'name': 'pyproject.toml does not exist or is empty', 'value': 'None'}

Confidence: 80%

Tokens: 143 input + 115 output = 258 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	136 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-325, 327-328, 332-336, 340, 342, 344, 348, 352, 356, 360-362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	30 lines (ranges: 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362-364, 366-367, 371-373, 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_validation_errors` 3ms 3

AI ASSESSMENT

Scenario:
tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_validation_errors

Why Needed: Validation errors are raised when the pytest configuration is invalid.

Key Assertions:

- {'name': 'pytest_configure raises UsageError', 'description': 'The pytest_configure function should raise a UsageError if the configuration is invalid.'}

Confidence: 80%

Tokens: 134 input + 87 output = 221 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	135 lines (ranges: 123, 171, 199, 202-205, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	25 lines (ranges: 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-358, 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_worker_skip` 1ms 2

AI ASSESSMENT

Scenario: TestPluginConfigure::test_pytest_configure_worker_skip

Why Needed: To ensure that the configure function skips on xdist workers correctly.

Key Assertions:

- {'name': 'mock_config.addinivalue_line.called', 'expected_result': True}

Confidence: 80%

Tokens: 170 input + 72 output = 242 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`17 lines (ranges: 328-330, 332-334, 336-338, 342-343, 558-559, 562, 566-568)`

PASSED `tests/test_plugin_maximal.py::TestPluginConfigureFallback::test_pytest_configure_fallback_load` 3ms 2

AI ASSESSMENT

Scenario: Test that fallback to load_config is triggered when Config.load is missing.

Why Needed: To prevent regression where Config.load is missing, and the plugin falls back to load_config.

Key Assertions:

- Mocking Config.load with None returns a mock object.
- Mocking load_config with no arguments calls validate() on mock_cfg.
- mock_load.assert_called_once() checks that load_config was called once.
- mock_cfg.validate.return_value is an empty list.
- mock_load.return_value is mock_cfg, which has the correct option values.
- Pytest_configure(mock_config) passes mock_config to load_config.
- load_config() does not call any other functions or methods on mock_cfg.
- The test fails if Config.load is missing and load_config is called without arguments.

Confidence: 80%

Tokens: 747 input + 179 output = 926 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	30 lines (ranges: 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362-364, 366-367, 371-373, 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginLoadConfig::test_load_config_cli_overrides_pyproject` 2ms 3

AI ASSESSMENT

Scenario:
tests/test_plugin_maximal.py::TestPluginLoadConfig::test_load_config_cli_overrides_pyproject

Why Needed: To test the plugin's ability to load configuration files with CLI options overriding those in pyproject.toml.

Key Assertions:

- {'name': 'pyproject.toml file creation', 'expected': 'pyproject.toml was created successfully', 'actual': 'pyproject.toml was not created'}
- {'name': 'pyproject.toml content', 'expected': 'pyproject.toml content was created with the correct CLI options', 'actual': 'pyproject.toml content was not created with the correct CLI options'}

Confidence: 80%

Tokens: 140 input + 159 output = 299 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	122 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460-461, 463-464, 466-467, 470, 472-473, 476-477, 482-494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599-607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginLoadConfig::test_load_config_from_pyproject` 120ms 3

AI ASSESSMENT

Scenario:
tests/test_plugin_maximal.py::TestPluginLoadConfig::test_load_config_from_pyproject

Why Needed: To ensure that the plugin can load configuration files from the PyPI repository.

Key Assertions:

- {'name': 'pyproject.toml file exists', 'expected': 'True'}
- {'name': 'pyproject.toml file is not empty', 'expected': 'True'}

Confidence: 80%

Tokens: 136 input + 105 output = 241 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	112 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-336, 340, 342, 344, 348, 352, 356, 360-362, 364, 366, 368, 372, 374, 378, 380, 382-384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginMaximal::test_terminal_summary_disabled` 1ms 2

AI ASSESSMENT

Scenario: Test that terminal summary skips when plugin is disabled.

Why Needed: Prevents a regression where the plugin's terminal summary might be incorrectly reported as enabled even though it's not.

Key Assertions:

- mocked stash.get() was called once with _enabled_key and False argument.
- Mocking stash.get() to return False for enabled is necessary because pytest_terminal_summary() relies on this assertion.
- pytest_terminal_summary() should have checked if the plugin is enabled before reporting its terminal summary.
- The test verifies that the plugin's terminal summary is skipped when it's disabled.
- This test ensures that the plugin's terminal summary is correctly reported as disabled even without worker input.
- Without this test, there might be a false positive report of the plugin being enabled when it's not.

Confidence: 80%

Tokens: 281 input + 179 output = 460 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	9 lines (ranges: 399, 403-404, 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginMaximal::test_terminal_summary_worker_skip` 1ms 2

AI ASSESSMENT

Scenario:
tests/test_plugin_maximal.py::TestPluginMaximal::test_terminal_summary_worker_skip

Why Needed: To test that terminal summary skips on xdist worker.

Key Assertions:

- {'assertion_type': 'is_none', 'expected_result': 'None'}

Confidence: 80%

Tokens: 164 input + 74 output = 238 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	8 lines (ranges: 399-400, 558-559, 562, 566-568)

PASSED tests/test_plugin_maximal.py::TestPluginMaximal::testload_config 5ms 3

AI ASSESSMENT

Scenario: Test config loading from pytest objects (CLI) to ensure the correct value is set for llm_report_html.

Why Needed: This test prevents a potential bug where the correct value for llm_report_html is not being set, potentially leading to incorrect configuration output.

Key Assertions:

- The `llm_report_html` option should be set to 'out.html'.
- The `llm_report_json` option should be set to 'out.json'.
- The `llm_report_pdf` option should be set to None.
- The `llm_evidence_bundle` option should be set to None.
- The `llm_dependency_snapshot` option should be set to None.
- The `llm_requests_per_minute` option should be set to None.
- The `llm_aggregate_dir` option should be set to None.
- The `llm_aggregate_policy` option should be set to None.
- The `llm_aggregate_run_id` option should be set to None.
- The `llm_aggregate_group_id` option should be set to None.
- The `llm_max_retries` option should be set to None.
- The `llm_coverage_source` option should be set to None.
- The `llm_prompt_tier` option should be set to None.
- The `llm_batch_parametrized` option should be set to None.
- The `llm_context_compression` option should be set to None.
- The `llm_context_bytes` option should be set to None.
- The `llm_context_file_limit` option should be set to None.
- The `llm_max_tests` option should be set to None.
- The `llm_max_concurrency` option should be set to None.
- The `llm_timeout_seconds` option should be set to None.
- The `llm_capture_failed` option should be set to None.
- The `llm_ollama_host` option should be set to None.
- The `llm_litellm_api_base` option should be set to None.
- The `llm_litellm_api_key` option should be set to None.
- The `llm_litellm_token_refresh_command` option should be set to None.
- The `llm_litellm_token_refresh_interval` option should be set to None.
- The `llm_litellm_token_output_format` option should be set to None.
- The `llm_litellm_token_json_key` option should be set to None.
- The `llm_cache_dir` option should be set to the value of `tmp_path`.

Confidence: 80%

Tokens: 639 input + 587 output = 1226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	69 lines (ranges: 123, 171, 308, 311-312, 320-322, 460-461, 463-464, 466-467, 470, 472-473, 476-477, 482-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginRuntest::test_runtest_makereport_disabled` 2ms 2

AI ASSESSMENT

Scenario: tests/test_plugin_maximal.py::TestPluginRuntest::test_runtest_makereport_disabled

Why Needed: The test is failing because the makereport hookwrapper is not completing successfully.

Key Assertions:

- {'name': 'gen.send(mock_outcome).exception', 'expected_type': 'StopIteration'}

Confidence: 80%

Tokens: 220 input + 86 output = 306 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	7 lines (ranges: 558-559, 562-563, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginRuntest::test_runtest_makereport_enabled` 2ms 2

AI ASSESSMENT

Scenario: Test that makereport calls collector when enabled.

Why Needed: This test prevents a potential regression where the plugin does not report any errors even if makereport is called.

Key Assertions:

- The `pytest_runtest_makereport` function should be able to find and call the `mock_collector` instance when it is enabled.
- The `mock_collector.handle_runtest_logreport` method should be called with the `mock_report` object as its argument.
- The `mock_collector` instance should have a `handle_runtest_logreport` method that takes two arguments: `mock_report` and `mock_item`.
- The `mock_collector` instance should be able to handle runtest log reports by calling the `handle_runtest_logreport` method.
- The `pytest_runtest_makereport` function should not call any other functions or methods on the `mock_collector` instance when it is enabled.
- The `mock_collector` instance should have a `stash_get` method that returns `True` for `_enabled_key` and `mock_collector` instances, and `None` otherwise.
- The `stash_get` method should return `False` for `_collector_key` and `None` otherwise.
- The `stash_get` method should not raise an exception when called with a key that is neither `_enabled_key` nor `_collector_key`.
-

Confidence: 80%

Tokens: 371 input + 317 output = 688 total

COVERAGE

`src/pytest_llm_report/collector.py`	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
`src/pytest_llm_report/plugin.py`	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_collection_finish_disabled` 1ms 2

AI ASSESSMENT

Scenario: tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_collection_finish_disabled

Why Needed: This test is needed because the pytest_collection_finish function should skip collection finish when disabled.

Key Assertions:

- {'name': 'mock_session.config.stash.get.assert_called_with(_enabled_key, False)', 'description': 'Verify that stash.get was called with _enabled_key and False as arguments.'}

Confidence: 80%

Tokens: 149 input + 103 output = 252 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`8 lines (ranges: 558-559, 562, 566-568, 602-603)`

PASSED `tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_collection_finish_enabled` 2ms 2

AI ASSESSMENT

Scenario: TestPluginSessionHooks

Why Needed: To ensure that the `pytest_collection_finish` function calls the `_collector_key` collector when collection finish is enabled.

Key Assertions:

- {'name': 'Mocking pytest_collection_finish with mock_collector', 'expected_result': 1, 'actual_result': 0}
- {'name': 'Mocking stash_get with _enabled_key and _collector_key', 'expected_result': ['True', 'mock_collector'], 'actual_result': ['True', 'mock_collector']}

Confidence: 80%

Tokens: 219 input + 134 output = 353 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	10 lines (ranges: 558-559, 562, 566-568, 602, 606-608)

PASSED `tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_sessionstart_disabled` 1ms 2

AI ASSESSMENT

Scenario: TestPluginSessionHooks

Why Needed: To ensure that the plugin correctly handles session start when disabled.

Key Assertions:

- {'name': 'mock_session.get', 'expected_calls': [{'_enabled_key': '_enabled_key', 'False': []}]}

Confidence: 80%

Tokens: 157 input + 76 output = 233 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	8 lines (ranges: 558-559, 562, 566-568, 619-620)

PASSED `tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_sessionstart_enabled` 1ms 3

AI ASSESSMENT

Scenario: Test that sessionstart initializes collector when enabled and creates a stash with both get() and [] methods.

Why Needed: This test prevents a potential regression where the collector is not created or does not have access to the stash, potentially leading to incorrect data collection or other issues.

Key Assertions:

- The _collector_key should be present in the mock_stash dictionary.
- The _start_time_key should also be present in the mock_stash dictionary.
- The collector should have been created successfully by pytest_sessionstart.
- _enabled_key should be set to True in stash_dict.
- Config() should have been created with stash_dict.
- pytest_sessionstart() should not raise any exceptions when called with a valid stash.
- The mock_stash should contain both get() and [] methods.
- The mock_stash should contain the _enabled_key and _config_key keys.

Confidence: 80%

Tokens: 335 input + 198 output = 533 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	11 lines (ranges: 558-559, 562, 566-568, 619, 623, 626, 628-629)

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_pytest_addoption` 3ms 2

AI ASSESSMENT

Scenario: Test pytest_addoption adds expected arguments and verifies specific options.

Why Needed: pytest_addoption prevents a potential bug where the plugin does not add all required arguments to the parser.

Key Assertions:

- parser.getgroup.assert_called_with('llm-report', 'LLM-enhanced test reports')
- group.addoption.call_args_list[0][0] == '--llm-report'
- group.addoption.call_args_list[1][0] == '--llm-coverage-source'

Confidence: 80%

Tokens: 293 input + 117 output = 410 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	220 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_pytest_addoption_no_ini` 3ms 2

AI ASSESSMENT

Scenario: tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_pytest_addoption_no_ini

Why Needed: pytest_addoption no longer adds INI options

Key Assertions:

- {'name': 'parser.addini was not called', 'expected_result': 0, 'actual_result': 1}

Confidence: 80%

Tokens: 140 input + 85 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	220 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_terminal_summary_coverage_calculation` 3ms 3

AI ASSESSMENT

Scenario: Test coverage percentage calculation logic for terminal summary.

Why Needed: Prevents regression in coverage reporting when terminal summary is enabled.

Key Assertions:

- The `report_html` option should be set to 'out.html' before calling `pytest_terminal_summary()`.
- The `CoverageMapper` instance should be created with the correct configuration.
- The `Coverage` object should have a report method that returns the coverage percentage correctly.
- The `MockStash` instance should be used as expected in the mock configuration.
- The `coverage.Coverage` class should be instantiated and returned correctly from the mock.
- The `pytest_llm_report.coverage_map.CoverageMapper` class should be patched to return a mock object.
- The `pytest_llm_report.report_writer.ReportWriter` class should be patched to return a mock object with a report method that returns 85.5 as expected.

Confidence: 80%

Tokens: 395 input + 203 output = 598 total

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 277, 285)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	53 lines (ranges: 399, 403, 407, 410, 429-430, 432, 434, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-466, 468, 470-473, 485-486, 491-492, 534-544, 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_terminal_summary_llm_enabled` 3ms 3

AI ASSESSMENT

Scenario: Test terminal summary with LLM enabled runs annotations.

Why Needed: Prevents regression by ensuring that the plugin is correctly configured when LLM is enabled.

Key Assertions:

- Verify that the `pytest_terminal_summary_llm_enabled` test function is executed only once.
- Check if the correct configuration is passed to `pytest_terminal_summary`.
- Verify that the LLM annotator is called with the correct arguments.
- Ensure that the provider is correctly retrieved and used.
- Verify that the coverage map is not modified during testing.
- Confirm that the report writer is properly initialized.
- Check if the LLM model name matches the expected value.

Confidence: 80%

Tokens: 477 input + 152 output = 629 total

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 277, 285)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	66 lines (ranges: 399, 403, 407, 410, 429-430, 432, 434, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485-486, 491-494, 497, 499, 502-504, 512-514, 516, 523-531, 534-544, 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_terminal_summary_no_collector` 2ms 3

AI ASSESSMENT

Scenario: Test terminal summary creates collector if missing.

Why Needed: The test prevents a potential bug where the plugin does not create a collector even when it is supposed to be present in the configuration.

Key Assertions:

- assert mock_config.stash._enabled_key == True
- assert mock_config.stash._config_key == cfg
- assert mock_terminalreporter.call_args_list[0][1] == [0, {}]
- assert mock_mapper.map_coverage.return_value == {}
- assert mock_writer_cls.return_value.report_writer.call_args_list[0][1] == [0, {}]
- assert stash._enabled_key == True and stash._config_key == cfg

Confidence: 80%

Tokens: 391 input + 157 output = 548 total

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 277, 285)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	45 lines (ranges: 399, 403, 407, 410, 429-430, 432, 434, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_terminal_summary_with_aggregation` 3ms 3

AI ASSESSMENT

Scenario: Test terminal summary with aggregation enabled.

Why Needed: This test prevents regression in the case where aggregation is enabled and there are multiple terminals being reported.

Key Assertions:

- The aggregate_dir parameter should be set to '/agg' when aggregation is enabled.
- The stash object should support both get() and [] methods.
- The aggregator function should return a report when aggregate=True.
- The ReportWriter class should write JSON and HTML files correctly when aggregate=True.
- The aggregate method of the Aggregator class should be called once when aggregate=True.
- The aggregation flag should be set to True in the config object.
- The stash object should have an _enabled_key with value True.

Confidence: 80%

Tokens: 441 input + 159 output = 600 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	21 lines (ranges: 399, 403, 407, 410-411, 413-414, 417-418, 420, 422-426, 558-559, 562, 566-568)

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummaryErrors::test_terminal_summary_coverage_error` 5ms 3

AI ASSESSMENT

Scenario: Test coverage calculation error when loading coverage map.

Why Needed: This test prevents regression where the coverage calculation fails due to an OSError during load.

Key Assertions:

- assert mock_cov_cls.return_value is None
- assert mock_cov.load.side_effect == OSError('Disk full')
- assert pytest_terminal_summary(MagicMock(), 0, mock_config).report_html is None
- assert mock_config.stash._enabled_key is True
- assert mock_config.stash._config_key is cfg
- assert _enabled_key in mock_config.stash
- assert _config_key in mock_config.stash

Confidence: 80%

Tokens: 389 input + 145 output = 534 total

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 277, 285)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	52 lines (ranges: 399, 403, 407, 410, 429-430, 432, 434, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-466, 476-479, 485-486, 491-492, 534-544, 558-559, 562, 566-568)

📄 tests/test_prompts.py

7 tests

PASSED `tests/test_prompts.py::TestContextAssembler::test_assemble_balanced_context` 7ms 5

AI ASSESSMENT

Scenario: Tests the ContextAssembler with a balanced context configuration to ensure it correctly includes dependencies and passes coverage tests.

Why Needed: This test prevents regression by ensuring that the ContextAssembler correctly assembles a balanced context, including all necessary dependencies.

Key Assertions:

- The 'utils.py' file is present in the assembled context.
- The 'def util()' function is found in the 'utils.py' file within the assembled context.
- The coverage report includes the 'utils.py' file and the 'def util()' function.

Confidence: 80%

Tokens: 331 input + 122 output = 453 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	63 lines (ranges: 33, 49, 52, 55, 58, 60-61, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 116, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-193, 196-197, 201, 284-285, 287)

PASSED `tests/test_prompts.py::TestContextAssembler::test_assemble_complete_context` 1ms 4

AI ASSESSMENT

Scenario: tests/test_prompts.py::TestContextAssembler::test_assemble_complete_context

Why Needed: To test the ContextAssembler's ability to assemble a complete context for a test file.

Key Assertions:

- {'assertion_type': 'contains', 'expected_value': 'test_1', 'actual_value': 'True'}

Confidence: 80%

Tokens: 176 input + 87 output = 263 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	38 lines (ranges: 33, 49, 52, 55, 58, 60, 63, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 116, 139-140, 268-272)

PASSED `tests/test_prompts.py::TestContextAssembler::test_assemble_minimal_context` 1ms 4

AI ASSESSMENT

Scenario: Test the ContextAssembler with minimal context mode and a test file.

Why Needed: This test prevents regression when using minimal context mode without specifying a repository root.

Key Assertions:

- The 'test_1' function is present in the source code of the test file.
- The context for the test function is empty.
- The test result nodeid matches the expected outcome.

Confidence: 80%

Tokens: 267 input + 94 output = 361 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	30 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 116)

PASSED `tests/test_prompts.py::TestContextAssembler::test_balanced_context_limits` 1ms 5

AI ASSESSMENT

Scenario: Test the ContextAssembler with balanced context limits to ensure it does not truncate long content within a file.

Why Needed: This test prevents bugs that may occur when the ContextAssembler is used with large files, causing the context to be truncated unnecessarily.

Key Assertions:

- The 'f1.py' file in the test result should contain the original long content.
- The 'truncated' message should not appear within the 'f1.py' file in the test result.
- The length of the 'f1.py' file in the test result should be less than or equal to 40 bytes (20 bytes + truncation message).

Confidence: 80%

Tokens: 335 input + 147 output = 482 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	46 lines (ranges: 33, 49, 52, 55, 58, 60-61, 65, 78-79, 82-84, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-194, 196-197, 201, 284-285, 287)

PASSED `tests/test_prompts.py::TestContextAssembler::test_complete_context_limits_override` 1ms 5

AI ASSESSMENT

Scenario: Test that 'complete' mode does not truncate long files despite a small llm_context_bytes limit.

Why Needed: This test prevents a regression where the LLM context size exceeds the file content size, causing truncation of long files in complete mode.

Key Assertions:

- Context is present in the assembled output.
- File content is preserved and not truncated.
- The 'truncated' assertion is not triggered.
- Context does not contain any 'truncated' key.
- Context size matches the file content size.
- LLM context bytes limit is respected for long files.
- Context assembler correctly handles large files in complete mode.

Confidence: 80%

Tokens: 361 input + 150 output = 511 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	50 lines (ranges: 33, 49, 52, 55, 58, 60, 63, 65, 78-79, 82-84, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-193, 196-197, 201, 268-272, 284-285, 287)

PASSED `tests/test_prompts.py::TestContextAssembler::test_get_test_source_edge_cases` 1ms 4

AI ASSESSMENT

Scenario: Verify the correct handling of non-existent files and nested test names with parameters.

Why Needed: This test prevents a potential bug where the ContextAssembler incorrectly handles cases where the test file does not exist or has nested test names with parameters.

Key Assertions:

- The function `_get_test_source` returns an empty string when given a non-existent file path.
- The function `_get_test_source` correctly identifies the nested test name with parameters in the provided source code.
- The function `_get_test_source` handles nested test names with parameters by including the parameter value in the source code.

Confidence: 80%

Tokens: 275 input + 135 output = 410 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	26 lines (ranges: 33, 78-79, 82-84, 86-87, 92, 94-95, 98-101, 103-112, 116)

PASSED `tests/test_prompts.py::TestContextAssembler::test_should_exclude` 2ms 4

AI ASSESSMENT

Scenario: The test verifies that the ContextAssembler should exclude certain Python files and directories from being processed.

Why Needed: This test prevents a potential bug where the ContextAssembler incorrectly includes certain files or directories in its processing, leading to unexpected behavior or errors.

Key Assertions:

- assert assembler._should_exclude('test.pyc') is True
- assert assembler._should_exclude('secret/key.txt') is True
- assert assembler._should_exclude('public/readme.md') is False

Confidence: 80%

Tokens: 227 input + 114 output = 341 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	1 lines (ranges: 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	5 lines (ranges: 33, 284-287)

📄 tests/test_prompts_coverage.py

12 tests

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_assemble_minimal_mode` 1ms 4

AI ASSESSMENT

Scenario: Test assemble minimal mode returns no context files.

Why Needed: To prevent a regression where the assemble function does not generate any context files when run in minimal mode.

Key Assertions:

- context_files == {}
- def test_foo()
- test_source contains 'def test_foo'

Confidence: 80%

Tokens: 298 input + 78 output = 376 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	29 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-109, 111-112, 116)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_assemble_with_context_override` 1ms 5

AI ASSESSMENT

Scenario: Test assemble respects llm_context_override from test.

Why Needed: This test prevents regression by ensuring the ContextAssembler uses the correct mode when overriding LLM context.

Key Assertions:

- ContextAssembler should use balanced mode due to override.
- ContextAssembler should include module.py in context_files.
- ContextAssembler should respect llm_context_override from test.
- Test assemble respects llm_context_override from test.
- ContextAssembler should not modify test file content when overriding LLM context.
- ContextAssembler should preserve original file path and line information when assembling with override mode.
- ContextAssembler should use correct mode for assembly (balanced in this case).
- ContextAssembler should respect the specified llm_context_override value.

Confidence: 80%

Tokens: 362 input + 166 output = 528 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	62 lines (ranges: 33, 49, 52, 55, 58, 60-61, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-109, 111-112, 116, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-193, 196-197, 201, 284-285, 287)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_excludes_patterns` 1ms 4

AI ASSESSMENT

Scenario: Test 'test_balanced_context_excludes_patterns' verifies that a balanced context excludes files matching exclude patterns.

Why Needed: This test prevents regression where the LLM context mode is set to 'balanced', and it includes files in the context that match exclude patterns.

Key Assertions:

- The file 'secret_config.py' should not be included in the balanced context.
- The file 'test_foo.py' should not be included in the balanced context.
- The LLM context mode is set to 'balanced', and it excludes files matching exclude patterns.
- The LLM context include glob pattern '*secret*' is excluded from the balanced context.
- The coverage of the test file 'secret_config.py' under the balanced context should be 0%.
- The coverage of the test file 'test_foo.py' under the balanced context should be 0%.
- The LLM context exclude glob pattern '*secret*' is not included in the output files.
- The LLM context include glob pattern '**/*' is excluded from the balanced context.
- The LLM context include glob pattern '**/*' excludes files that match exclude patterns.

Confidence: 80%

Tokens: 331 input + 254 output = 585 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	1 lines (ranges: 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	20 lines (ranges: 33, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163-164, 201, 284-286)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_file_not_exists` 1ms 4

AI ASSESSMENT

Scenario: tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_file_not_

Why Needed: To ensure that the ContextAssembler correctly handles cases where a balanced context file is not found.

Key Assertions:

- {'assertion': 'context is empty', 'description': 'The ContextAssembler should return an empty dictionary when no balanced context file exists.'}

Confidence: 80%

Tokens: 201 input + 96 output = 297 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	16 lines (ranges: 33, 139, 142-145, 147-148, 152-153, 155-156, 159-161, 201)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_max_bytes_limit` 14ms 5

AI ASSESSMENT

Scenario: Test that balanced context respects max bytes limit.

Why Needed: This test prevents a potential bug where the LLM context exceeds the maximum allowed bytes, causing truncated content.

Key Assertions:

- The content of the source file is not longer than 120 bytes.
- The LLM context is not truncated when the source file exceeds 6000 bytes.
- The LLM context is truncated when the source file exceeds 10000 bytes (1000 + 9000).
- The LLM context does not exceed the maximum allowed bytes even if it contains a large number of lines.
- The LLM context does not contain any 'truncated' messages when the source file exceeds 6000 bytes.
- The LLM context is truncated only after the last line of the source file.

Confidence: 80%

Tokens: 405 input + 180 output = 585 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	34 lines (ranges: 33, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-194, 196-197, 201, 284-285, 287)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_no_coverage` 1ms 4

AI ASSESSMENT

Scenario: tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_no_cove

Why Needed: To ensure that the ContextAssembler can correctly assemble a balanced context with no coverage.

Key Assertions:

- {'assertion_type': 'equals', 'expected_value': '{}', 'actual_value': '{}'}

Confidence: 80%

Tokens: 162 input + 86 output = 248 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	3 lines (ranges: 33, 139-140)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_reaches_max_bytes_before_file` 1ms 5

AI ASSESSMENT

Scenario: Test that loop exits when max bytes is reached before processing file.

Why Needed: Prevents a potential memory leak by ensuring the context assembler does not exceed the maximum allowed bytes before processing files.

Key Assertions:

- context should have only one node (either 'file1.py' or 'file2.py')
- context length should be less than or equal to 1
- context should contain both file paths and their respective lines and line counts

Confidence: 80%

Tokens: 409 input + 110 output = 519 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	35 lines (ranges: 33, 139, 142-145, 147-148, 152-153, 155-157, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-194, 196-197, 201, 284-285, 287)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_complete_context_delegates_to_balanced` 1ms 5

AI ASSESSMENT

Scenario: tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_complete_context_delegat

Why Needed: To ensure that complete context delegates to balanced correctly.

Key Assertions:

- {'assertion_type': 'inclusion', 'expected_values': ['module.py'], 'actual_value': 'module.py'}

Confidence: 80%

Tokens: 211 input + 87 output = 298 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	38 lines (ranges: 33, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-193, 196-197, 201, 268-272, 284-285, 287)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_empty_nodeid` 1ms 4

AI ASSESSMENT

Scenario: Test _get_test_source with empty nodeid returns empty string

Why Needed: To ensure that the ContextAssembler correctly handles an empty node ID in the test source.

Key Assertions:

- {'assertion_name': 'result == ""', 'expected_result': ''}

Confidence: 80%

Tokens: 148 input + 74 output = 222 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	9 lines (ranges: 33, 78-79, 82-83, 86-89)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_extraction_stops_at_next_def` 1ms 4

AI ASSESSMENT

Scenario:
tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_extraction

Why Needed: To ensure that source extraction stops at the next function definition, even if there are multiple definitions in a single file.

Key Assertions:

- {'assertion': 'The test source extraction should stop at the next function definition.', 'expected_result': 'True'}

Confidence: 80%

Tokens: 129 input + 100 output = 229 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	25 lines (ranges: 33, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 114, 116)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_file_not_exists` 1ms 4

AI ASSESSMENT

Scenario: Edge Case: Test Source File Not Exists

Why Needed: The test assembly function `_get_test_source` should handle cases where the test source file does not exist.

Key Assertions:

- {'description': 'Test that the function returns an empty string for a non-existent test source file.', 'expected_result': '', 'actual_result': ''}

Confidence: 80%

Tokens: 142 input + 90 output = 232 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	6 lines (ranges: 33, 78-79, 82-84)

PASSED `tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_with_class` 1ms 4

AI ASSESSMENT

Scenario:
tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_with_class

Why Needed: To ensure that the _get_test_source function correctly extracts functions with proper indentation, even when they are nested within other code blocks.

Key Assertions:

- {'name': 'Expected output is a string', 'expected_value': 'test_example.py', 'actual_value': {'scenario': 'tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_with_c 'why_needed': 'To ensure that the _get_test_source function correctly extracts functions with proper indentation, even when they are nested within other code blocks.', 'key_assertions': ['Expected output is a string'], 'value': 'test_example.py'}}

Confidence: 80%

Tokens: 118 input + 174 output = 292 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	25 lines (ranges: 33, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 114, 116)

📄 tests/test_ranges.py

13 tests

PASSED tests/test_ranges.py::TestCompressRanges::test_consecutive_lines 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_consecutive_lines

Why Needed: To ensure that consecutive lines are compressed into a single range.

Key Assertions:

- {'expected': '1-3', 'actual': '1-3'}

Confidence: 80%

Tokens: 106 input + 72 output = 178 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	12 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65, 67)

PASSED tests/test_ranges.py::TestCompressRanges::test_duplicates 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_duplicates

Why Needed: To test the handling of duplicate ranges in the compress_ranges function.

Key Assertions:

- {'expected': '1-3', 'actual': '1-2'}
- {'expected': '2-4', 'actual': '2-3'}

Confidence: 80%

Tokens: 107 input + 93 output = 200 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	12 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65, 67)

PASSED tests/test_ranges.py::TestCompressRanges::test_empty_list 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_empty_list

Why Needed: Because an empty list is considered a valid input for the `compress_ranges` function.

Key Assertions:

- {'expected_value': '', 'actual_value': ''}

Confidence: 80%

Tokens: 92 input + 70 output = 162 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	2 lines (ranges: 29-30)

PASSED tests/test_ranges.py::TestCompressRanges::test_mixed_ranges 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_mixed_ranges

Why Needed: To test the functionality of compressing mixed ranges in a list.

Key Assertions:

- {'expected': '1-3, 5, 10-12, 15', 'actual': '1-3, 5, 10-12, 15'}

Confidence: 80%

Tokens: 130 input + 95 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	16 lines (ranges: 29, 33, 35-37, 39-40, 42, 45-47, 50, 52, 65-67)

PASSED `tests/test_ranges.py::TestCompressRanges::test_non_consecutive_lines` 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_non_consecutive_lines

Why Needed: To ensure that non-consecutive lines are correctly compressed to a single comma-separated value.

Key Assertions:

- {'expected_value': '1, 3, 5', 'actual_value': '1, 3, 5'}

Confidence: 80%

Tokens: 113 input + 88 output = 201 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	14 lines (ranges: 29, 33, 35-37, 39-40, 45-47, 50, 52, 65-66)

PASSED `tests/test_ranges.py::TestCompressRanges::test_single_line` 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_single_line

Why Needed: The single line should be compressed using the range notation.

Key Assertions:

- {'expected': 5, 'actual': '5'}

Confidence: 80%

Tokens: 96 input + 66 output = 162 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	10 lines (ranges: 29, 33, 35-37, 39, 50, 52, 65-66)

PASSED tests/test_ranges.py::TestCompressRanges::test_two_consecutive 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_two_consecutive

Why Needed: The test is necessary because the current implementation does not handle two consecutive lines correctly.

Key Assertions:

- {'expected': '1-2', 'actual': '1-2'}

Confidence: 80%

Tokens: 103 input + 76 output = 179 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	12 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65, 67)

PASSED `tests/test_ranges.py::TestCompressRanges::test_unsorted_input` 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_unsorted_input

Why Needed: The test is necessary to ensure that the `compress_ranges` function can handle unsorted input correctly.

Key Assertions:

- {'expected': '1-3, 5', 'actual': '1-3, 5'}

Confidence: 80%

Tokens: 110 input + 86 output = 196 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	16 lines (ranges: 29, 33, 35-37, 39-40, 42, 45-47, 50, 52, 65-67)

PASSED `tests/test_ranges.py::TestExpandRanges::test_empty_string` 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_empty_string

Why Needed: The current implementation does not handle empty strings correctly.

Key Assertions:

- {'expected': [], 'actual': []}

Confidence: 80%

Tokens: 90 input + 60 output = 150 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	2 lines (ranges: 81-82)

PASSED tests/test_ranges.py::TestExpandRanges::test_mixed 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_mixed

Why Needed: The test is necessary because it checks for the correct expansion of mixed ranges and singles.

Key Assertions:

- {'expected': [1, 2, 3, 5, 10, 11, 12], 'actual': ['1', '2', '3', '5', '10', '11', '12']}
- {'expected': [], 'actual': []}

Confidence: 80%

Tokens: 121 input + 113 output = 234 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	11 lines (ranges: 81, 84-91, 93, 95)

PASSED tests/test_ranges.py::TestExpandRanges::test_range 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_range

Why Needed: The range function is not correctly expanding the input string.

Key Assertions:

- {'expected': [1, 2, 3], 'actual': ['1', '2', '3']}
- {'expected': "expand_ranges('1-3')", 'actual': "['1', '2', '3']"}

Confidence: 80%

Tokens: 99 input + 99 output = 198 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	10 lines (ranges: 81, 84-91, 95)

PASSED tests/test_ranges.py::TestExpandRanges::test_roundtrip 1ms 3

AI ASSESSMENT

Scenario: compress_ranges and expand_ranges should be inverses.

Why Needed: This test ensures that the `compress_ranges` and `expand_ranges` functions are inverse operations, meaning they can be used to reconstruct the original list from a compressed representation.

Key Assertions:

- {'name': 'original == expanded', 'expected': [1, 2, 3, 5, 10, 11, 12, 15], 'message': 'Original and expanded lists must be equal.'}

Confidence: 80%

Tokens: 134 input + 116 output = 250 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	27 lines (ranges: 29, 33, 35-37, 39-40, 42, 45-47, 50, 52, 65-67, 81, 84-91, 93, 95)

PASSED `tests/test_ranges.py::TestExpandRanges::test_single_number` 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_single_number

Why Needed: To ensure that the `expand_ranges` function correctly handles a single number as input.

Key Assertions:

- {'message': 'The output should be an array containing only the single element: 5', 'expected_value': [5]}

Confidence: 80%

Tokens: 95 input + 85 output = 180 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	7 lines (ranges: 81, 84-87, 93, 95)

📄 tests/test_render.py

9 tests

PASSED tests/test_render.py::TestFormatDuration::test_milliseconds 1ms 3

AI ASSESSMENT

Scenario: tests/test_render.py::TestFormatDuration::test_milliseconds verifies that the function correctly formats durations in milliseconds for times less than 1 second.

Why Needed: This test prevents a potential bug where the function does not format durations as expected for times less than 1 second, potentially leading to incorrect rendering of time-related content.

Key Assertions:

- {'description': "durations under 1s are formatted correctly as '0ms'"}
- {'description': "durations between 1s and 2s are formatted correctly as '100ms'"}
- {'description': "durations exactly equal to 1 second are formatted correctly as '1000ms'"}
- {'description': "durations greater than or equal to 2 seconds are formatted correctly as '2000ms'"}
- {'description': "durations under 0.5s are formatted correctly as '500ms'"}
- {'description': "durations between 0.5s and 1 second are formatted correctly as '1000ms'"}
- {'description': "durations exactly equal to 1 second is formatted correctly as '2000ms'"}
- {'description': "durations greater than or equal to 2 seconds are formatted correctly as '4000ms'}"}

Confidence: 80%

Tokens: 211 input + 274 output = 485 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	2 lines (ranges: 65, 67)

PASSED tests/test_render.py::TestFormatDuration::test_seconds 1ms 3

AI ASSESSMENT

Scenario: tests/test_render.py::TestFormatDuration::test_seconds

Why Needed: To ensure the function `format_duration` correctly formats time durations in seconds.

Key Assertions:

- {'message': "Expected format to be '1.23s' for duration 1.23 seconds"}
- {'message': "Expected format to be '60.00s' for duration 60 seconds"}

Confidence: 80%

Tokens: 116 input + 94 output = 210 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	2 lines (ranges: 65-66)

PASSED tests/test_render.py::TestOutcomeToCssClass::test_all_outcomes 1ms 3

AI ASSESSMENT

Scenario: Test Outcome Mapping to CSS Classes

Why Needed: To ensure that all outcomes are correctly mapped to their corresponding CSS classes.

Key Assertions:

- The `outcome_to_css_class` function should map each outcome to a unique CSS class.
- The function should handle cases where an outcome is not recognized (e.g., 'xfailed').
- The function should preserve the original outcome value when mapping to a CSS class (e.g., 'passed' → 'outcome-passed').

Confidence: 80%

Tokens: 263 input + 116 output = 379 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	8 lines (ranges: 79-85, 87)

PASSED tests/test_render.py::TestOutcomeToCssClass::test_unknown_outcome 1ms 3

AI ASSESSMENT

Scenario: tests/test_render.py::TestOutcomeToCssClass::test_unknown_outcome

Why Needed: The test is necessary because it checks for the default CSS class when an unknown outcome is encountered.

Key Assertions:

- {'assertion': "outcome_to_css_class('unknown') == 'outcome-unknown'", 'expected_result': 'outcome-unknown'}

Confidence: 80%

Tokens: 102 input + 91 output = 193 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	8 lines (ranges: 79-85, 87)

PASSED `tests/test_render.py::TestRenderFallbackHtml::test_renders_basic_report` 1ms 3

AI ASSESSMENT

Scenario: The test verifies that a complete HTML document is rendered with the expected report content.

Why Needed: This test prevents a potential rendering issue where the report might not be displayed correctly due to missing or incorrect HTML elements.

Key Assertions:

- The presence of the '' header in the rendered HTML.
- The inclusion of 'Test Report' in the HTML content.
- The presence of 'test::passed' and 'test::failed' node IDs in the HTML.
- The correct display of 'PASSED' and 'FAILED' text within the report.
- The accurate display of plugin and repository versions in the HTML.

Confidence: 80%

Tokens: 426 input + 149 output = 575 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	57 lines (ranges: 65-67, 79-85, 87, 121-124, 126-127, 131-132, 155-157, 159-167, 172-174, 210-211, 224, 257-264, 267, 269, 271-277, 280-281, 285)

PASSED `tests/test_render.py::TestRenderFallbackHtml::test_renders_coverage` 1ms 3

AI ASSESSMENT

Scenario: Test renders coverage for fallback HTML test.

Why Needed: Prevents regression and ensures accurate coverage reporting.

Key Assertions:

- The 'src/foo.py' file should be included in the rendered HTML.
- The number of lines rendered should match the total number of lines in the file.
- All lines in the file should be present in the rendered HTML.

Confidence: 80%

Tokens: 288 input + 90 output = 378 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	57 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-129, 131-132, 155-156, 159-167, 172-174, 210-211, 224, 257-264, 267, 269, 271-277, 280-281, 285)

PASSED `tests/test_render.py::TestRenderFallbackHtml::test_renders_llm_annotation` 1ms 3

AI ASSESSMENT

Scenario: tests/test_render.py::TestRenderFallbackHtml::test_renders_llm_annotation

Why Needed: This test prevents the rendering of LLM annotations with a low confidence score, which could be misleading and potentially lead to security vulnerabilities.

Key Assertions:

- The report includes "Tests login flow" in its HTML content.
- The report includes "Prevents auth bypass" in its HTML content.
- The report includes the string 'Confidence:' in its HTML content with a confidence score of '85%'.
- The report does not include any LLM annotations without a confidence score.

Confidence: 80%

Tokens: 317 input + 139 output = 456 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	64 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-127, 131-134, 136-137, 140-142, 144, 147, 155-156, 159-167, 172-174, 210-211, 224, 257-264, 267, 269, 271-277, 280-281, 285)

PASSED `tests/test_render.py::TestRenderFallbackHtml::test_renders_source_coverage` 1ms 3

AI ASSESSMENT

Scenario: Test renders source coverage for fallback HTML.

Why Needed: Prevents a regression where the source coverage summary is not displayed correctly when using fallback HTML.

Key Assertions:

- The 'Source Coverage' section should be present in the rendered HTML.
- The 'src/foo.py' file path should be included in the 'Source Coverage' section.
- The percentage of covered code (80.0%) should be displayed correctly in the 'Source Coverage' section.
- The ranges of missed code ('1-4, 6-8') and missed files ('5, 9-10') should be accurately represented in the 'Source Coverage' section.
- The coverage percentage should be calculated correctly based on the actual number of statements (10) and the total number of lines (12).
- The covered code ranges should match the expected values ('1-4', '6-8').
- The missed code ranges should match the expected values ('5', '9-10').
-
- key_assertions[0] = True
- # The 'Source Coverage' section should be present in the rendered HTML.

Confidence: 80%

Tokens: 331 input + 254 output = 585 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	68 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-127, 131-132, 155-156, 159-167, 172-178, 180-186, 191, 206, 210-211, 224, 257-264, 267, 269, 271-277, 280-281, 285)

PASSED `tests/test_render.py::TestRenderFallbackHtml::test_renders_xpass_summary` 1ms 3

AI ASSESSMENT

Scenario: Test renders xpass summary for ReportRoot report.

Why Needed: This test prevents a regression where the 'xfailed/xpassed' summary is not rendered correctly when there are multiple failed and passed tests.

Key Assertions:

- The string 'XFailed' should be present in the HTML output.
- The string 'XPassed' should be present in the HTML output.
- Both 'XFailed' and 'XPassed' strings should be found in the HTML output.

Confidence: 80%

Tokens: 283 input + 113 output = 396 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`
`src/pytest_llm_report/render.py`	`55 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-127, 131-132, 155-156, 159-167, 172-174, 210-211, 224, 257-264, 267, 269, 271-277, 280-281, 285)`

📄 tests/test_report_writer.py

19 tests

PASSED tests/test_report_writer.py::TestComputeSha256::test_different_content 1ms 3

AI ASSESSMENT

Scenario: tests/test_report_writer.py::TestComputeSha256::test_different_content

Why Needed: To ensure that different content produces different hashes.

Key Assertions:

- {'expected': {'hash': 'd41d8cd98f00b804d0a131e86038e95'}, 'actual': {'hash': '6f5dbce7c8694edd9f2d0783ba3f1d32'}}

Confidence: 80%

Tokens: 115 input + 111 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	1 lines (ranges: 55)

PASSED tests/test_report_writer.py::TestComputeSha256::test_empty_bytes 1ms 3

AI ASSESSMENT

Scenario: tests/test_report_writer.py::TestComputeSha256::test_empty_bytes

Why Needed: To ensure that the test suite is robust and can handle empty input data.

Key Assertions:

- {'message': 'Empty bytes should produce consistent hash.', 'expected_result': 'True'}
- {'message': 'Hash1 length should be 64 (SHA256 hex length).', 'expected_result': 64}

Confidence: 80%

Tokens: 129 input + 98 output = 227 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	1 lines (ranges: 55)

PASSED tests/test_report_writer.py::TestReportWriter::test_build_run_meta 10ms 4

AI ASSESSMENT

Scenario: Test 'Run meta should include version info' verifies that the test report writer correctly includes version information in the build run metadata.

Why Needed: This test prevents regression where the test report writer does not include version information in the build run metadata, potentially leading to incorrect reporting or analysis of test results.

Key Assertions:

- The duration of the test should be 60.0 seconds.
- The pytest version should have a value.
- The plugin version should match the current __version__.
- The Python version should match the current __python_version__.

Confidence: 80%

Tokens: 318 input + 131 output = 449 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	72 lines (ranges: 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307)

PASSED `tests/test_report_writer.py::TestReportWriter::test_build_summary_all_outcomes` 1ms 4

AI ASSESSMENT

Scenario: Test verifies that the `build_summary` method counts all outcome types correctly.

Why Needed: This test prevents a regression where the summary does not include all outcome types, potentially leading to incorrect reporting.

Key Assertions:

- The total count of outcomes should be equal to 6 (all outcome types).
- The number of passed outcomes should be 1.
- The number of failed outcomes should be 1.
- The number of skipped outcomes should be 1.
- The number of xfailed outcomes should be 1.
- The number of xpassed outcomes should be 1.
- The number of error outcomes should be 1.

Confidence: 80%

Tokens: 336 input + 152 output = 488 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	19 lines (ranges: 156-158, 319, 321-322, 324-335, 337)

PASSED `tests/test_report_writer.py::TestReportWriter::test_build_summary_counts` 1ms 4

AI ASSESSMENT

Scenario: Test that the `build_summary_counts` method correctly counts outcomes in a test report.

Why Needed: This test prevents regression where the total count of passed, failed and skipped tests is not updated correctly.

Key Assertions:

- The total number of tests should be equal to the sum of passed, failed and skipped tests.
- The number of passed tests should be equal to the sum of passed outcomes.
- The number of failed tests should be equal to the sum of failed outcomes.
- The number of skipped tests should be equal to the sum of skipped outcomes.
- All test results should have a valid `nodeid` and an associated `outcome`.
- The summary should not contain any invalid or missing data.

Confidence: 80%

Tokens: 283 input + 165 output = 448 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	13 lines (ranges: 156-158, 319, 321-322, 324-329, 337)

PASSED tests/test_report_writer.py::TestReportWriter::test_create_writer 1ms 4

AI ASSESSMENT

Scenario: Testing the creation of a ReportWriter instance with a valid configuration.

Why Needed: This test prevents potential bugs where a new ReportWriter instance is created without properly initializing its configuration.

Key Assertions:

- The `config` attribute of the `ReportWriter` instance should be equal to the provided `Config` object.
- The `warnings` list of the `ReportWriter` instance should be empty.
- The `artifacts` list of the `ReportWriter` instance should be empty.

Confidence: 80%

Tokens: 199 input + 117 output = 316 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	3 lines (ranges: 156-158)

PASSED `tests/test_report_writer.py::TestReportWriter::test_write_report_assembles_tests` 10ms 4

AI ASSESSMENT

Scenario: Test writes a report with all assembled tests.

Why Needed: This test prevents regression where the report does not include all tests, potentially leading to incorrect reporting or missing important information.

Key Assertions:

- The length of the report.tests list should be equal to 2 (the number of tests).
- The total value of report.summary.total should be equal to 2 (the number of tests).

Confidence: 80%

Tokens: 255 input + 99 output = 354 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	98 lines (ranges: 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-327, 337)

PASSED `tests/test_report_writer.py::TestReportWriter::test_write_report_includes_coverage_percent` 10ms 4

AI ASSESSMENT

Scenario:
tests/test_report_writer.py::TestReportWriter::test_write_report_includes_coverage_percent

Why Needed: To ensure that the ReportWriter class correctly calculates and returns the total coverage percentage in the report.

Key Assertions:

- {'name': 'assert report.summary.coverage_total_percent == 85.5', 'expected_value': 85.5, 'message': 'Expected coverage total percent to be 85.5'}

Confidence: 80%

Tokens: 132 input + 108 output = 240 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	98 lines (ranges: 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-199, 202-206, 211-218, 222, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321, 337)

PASSED `tests/test_report_writer.py::TestReportWriter::test_write_report_includes_source_coverage` 10ms 4

AI ASSESSMENT

Scenario: Test ReportWriter::test_write_report_includes_source_coverage verifies that the test writes a report with source coverage summary.

Why Needed: This test prevents regression where the report does not include source coverage information, which is crucial for debugging and tracking changes in codebase.

Key Assertions:

- The length of `report.source_coverage` should be 1.
- The file path of the first element in `report.source_coverage` should match 'src/foo.py'.
- All elements in `report.source_coverage` should have a valid `file_path` attribute.
- Each element in `report.source_coverage` should have a corresponding `covered` value between 0 and 100.
- The total coverage percentage of all covered statements should be greater than or equal to the given coverage percent.
- All covered ranges should match one of the provided patterns.
- All missed ranges should match an empty string.
- Each statement in `source_coverage` should have a corresponding `missed` value between 0 and 100.
- The total number of statements in `source_coverage` should be greater than or equal to the given number of statements.
- All covered ranges should cover at least one statement.
- All missed ranges should not contain any statements.

Confidence: 80%

Tokens: 291 input + 274 output = 565 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

src/pytest_llm_report/report_writer.py	97 lines (ranges: 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202-206, 211-218, 222, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321, 337)

PASSED `tests/test_report_writer.py::TestReportWriter::test_write_report_merges_coverage` 10ms 4

AI ASSESSMENT

Scenario: Test ReportWriter::test_write_report_merges_coverage verifies that the test writes a merged coverage report.

Why Needed: This test prevents regression where the coverage is not properly merged into tests, potentially leading to inaccurate reporting.

Key Assertions:

- The report should contain only one coverage entry for the specified test.
- The file path of the coverage entry matches the expected file path.
- All line ranges and counts in the coverage entry match the expected values.

Confidence: 80%

Tokens: 285 input + 110 output = 395 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	99 lines (ranges: 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186-189, 192-193, 197-198, 202, 211-218, 222, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337)

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_atomic_write_fallback` 11ms 6

AI ASSESSMENT

Scenario: Test that the ReportWriterWithFiles class falls back to direct write if atomic write fails and writes warnings.

Why Needed: This test prevents a regression where the ReportWriterWithFiles class does not fall back to direct write when an atomic write operation fails, potentially leading to incorrect report generation or data loss.

Key Assertions:

- The file "report.json" exists at the expected location.
- Any warnings generated by the ReportWriter are marked with code 'W203'.
- The `write_report` method does not raise an exception when writing to a non-existent file.

Confidence: 80%

Tokens: 276 input + 133 output = 409 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	62 lines (ranges: 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	130 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202-206, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513-514, 516-519, 522-523)

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_creates_directory_if_missing` 11ms 6

AI ASSESSMENT

Scenario: Test case 'tests/test_report_writer.py::TestReportWriterWithFiles::test_creates_directory_if_missing'

Why Needed: Because the test writer does not create an output directory if it already exists.

Key Assertions:

- {'assertion': 'The output directory should exist.', 'expected_result': 'True'}
- {'assertion': 'The output file should be created.', 'expected_result': 'False'}

Confidence: 80%

Tokens: 171 input + 109 output = 280 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	81 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	128 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-484, 502, 504, 506-508, 510, 513)

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_ensure_dir_failure` 1ms 4

AI ASSESSMENT

Scenario: Test that a directory creation failure prevents the capture of a warning code 'W201' from ReportWriter._ensure_dir.

Why Needed: This test prevents a regression where the report writer does not capture warnings when creating directories with permission issues.

Key Assertions:

- The function `writer.warnings` is populated with warnings that have a code of 'W201'.
- The directory creation fails and an OSError is raised.
- The `pathlib.Path.mkdir` mock raises an OSError with the message 'Permission denied' when called on the specified path.

Confidence: 80%

Tokens: 278 input + 129 output = 407 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	12 lines (ranges: 156-158, 477-480, 487-491)

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_git_info_failure` 1ms 3

AI ASSESSMENT

Scenario: Test 'test_git_info_failure' verifies that the `get_git_info` function handles git command failures gracefully by returning `None` for both SHA and dirty flag values.

Why Needed: This test prevents a regression where the `get_git_info` function fails to return expected values when it encounters a git command failure.

Key Assertions:

- The `get_git_info()` function should not raise an exception when `git` is not found.
- The `get_git_info()` function should set both `sha` and `dirty` attributes to `None` in such cases.
- The test should be able to run without any issues or errors when the git command fails.
- The expected values of `sha` and `dirty` should match the actual output for a successful git command execution.
- The function should not raise an exception when it encounters a non-existent git repository.
- The function should set both `sha` and `dirty` attributes to `None` even if the git command fails but returns no error message.
- The test should be able to handle cases where the git command fails due to other reasons (e.g., network issues, etc.) without crashing or raising an exception.

Confidence: 80%

Tokens: 231 input + 262 output = 493 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	9 lines (ranges: 67-73, 85-86)

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_write_html_creates_file` 45ms 6

AI ASSESSMENT

Scenario: Test verifies that the report writer creates an HTML file with expected content.

Why Needed: This test prevents a regression where the report writer does not create an HTML file even if there are tests that fail or are skipped.

Key Assertions:

- The 'report.html' file should exist in the temporary directory.
- The 'report.html' file should contain the expected content.
- The 'report.html' file should include the following text: 'test1', 'test2', 'PASSED', 'FAILED', 'Skipped', 'XFailed', and 'XPassed'.
- The 'report.html' file should not be empty.

Confidence: 80%

Tokens: 366 input + 149 output = 515 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	120 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-327, 337, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_write_html_includes_xfail_summary` 46ms 6

AI ASSESSMENT

Scenario: Test verifies that xfail outcomes are included in the HTML summary.

Why Needed: This test prevents regression where xfail results are not included in the report.

Key Assertions:

- The 'XFAILED' keyword is present in the HTML summary.
- The 'XFailed' keyword is present in the HTML summary.
- The 'XPASSED' keyword is present in the HTML summary.
- The 'XPassed' keyword is present in the HTML summary.
- All xfail results are included in the report.
- No xfail results are excluded from the report.
- The report includes a summary of all test outcomes.

Confidence: 80%

Tokens: 308 input + 148 output = 456 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	123 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324, 326, 328, 330-333, 337, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_write_json_creates_file` 11ms 5

AI ASSESSMENT

Scenario: Test verifies that a JSON file is created with the report.

Why Needed: This test prevents regression where the report writer does not create a JSON file.

Key Assertions:

- The `report.json` file should be created in the specified path.
- At least one artifact should be tracked for the report.
- The number of artifacts should be greater than zero.
- The `writer.artifacts` list should contain at least one element.
- The JSON file should have a valid hash.

Confidence: 80%

Tokens: 265 input + 118 output = 383 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	80 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	122 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_write_pdf_creates_file` 49ms 6

AI ASSESSMENT

Scenario: Test verifies that the `write_pdf` method creates a PDF file when Playwright is available.

Why Needed: This test prevents regression where the `report_writer` module does not create a PDF file when Playwright is installed.

Key Assertions:

- The `writer` object should have created a new PDF file at the specified path.
- Any artifacts generated by the test should match the expected path.
- The `writer` object should have returned a list of artifact paths that match the expected path.
- The `writer` object's `artifacts` attribute should contain any artifacts generated by the test.
- The `writer` object's `path` attribute should be set to the expected PDF file path.
- Any errors raised during the execution of the `write_pdf` method should have been caught and reported correctly.

Confidence: 80%

Tokens: 478 input + 186 output = 664 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)

src/pytest_llm_report/report_writer.py	130 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226, 230-231, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 408, 417, 419, 421-430, 441-442, 444-450, 455, 460, 462, 465-469, 477-478)

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_write_pdf_missing_playwright_warns` 11ms 4

AI ASSESSMENT

Scenario: Test should warn when Playwright is missing for PDF output.

Why Needed: To prevent a warning about missing Playwright for PDF output, which may indicate an issue with the test environment.

Key Assertions:

- The `pdf_path` does not exist after writing the report.
- At least one warning has the code `WarningCode.W204_PDF_PLAYWRIGHT_MISSING.value`.

Confidence: 80%

Tokens: 311 input + 95 output = 406 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	103 lines (ranges: 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226, 230-231, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 408-412, 415)

📄 tests/test_report_writer_coverage.py

10 tests

PASSED `tests/test_report_writer_coverage.py::TestGetGitInfo::test_git_info_from_nonexistent_path` 3ms 🛡️ 3

AI ASSESSMENT

Scenario: tests/test_report_writer_coverage.py::TestGetGitInfo::test_git_info_from_nonexistent_path

Why Needed: To test that the report writer does not attempt to write to a non-existent Git directory.

Key Assertions:

- {'name': 'assert sha is None', 'expected_result': {'type': 'NoneType', 'message': 'git info from nonexistent path should return None'}}
- {'name': 'assert dirty is None', 'expected_result': {'type': 'NoneType', 'message': 'git info from nonexistent path should return None'}}

Confidence: 80%

Tokens: 123 input + 139 output = 262 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	9 lines (ranges: 67-73, 85-86)

PASSED `tests/test_report_writer_coverage.py::TestGetGitInfo::test_git_info_from_valid_repo` 10ms 3

AI ASSESSMENT

Scenario: tests/test_report_writer_coverage.py::TestGetGitInfo::test_git_info_from_valid_repo

Why Needed: To ensure that the `get_git_info` function returns a valid git SHA for a valid repository.

Key Assertions:

- {'name': 'Expected result is None or str', 'description': 'The test expects the function to return either None or a string (representing the git SHA).'}

Confidence: 80%

Tokens: 159 input + 103 output = 262 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	16 lines (ranges: 67-74, 76-81, 83-84)

PASSED `tests/test_report_writer_coverage.py::TestGetPluginGitInfo::test_plugin_git_info_fallback` 1ms 3

AI ASSESSMENT

Scenario: Test falls back to git runtime when _git_info import fails.

Why Needed: Prevents a regression where the plugin's Git info cannot be retrieved due to an import failure.

Key Assertions:

- The function `get_plugin_git_info()` returns None or a string when the fallback is necessary.
- The function `get_plugin_git_info()` does not raise any exceptions when the fallback is necessary.
- The function `get_plugin_git_info()` still works via git runtime after the fallback.
- The `_git_info` cache is cleared before the fallback occurs.
- The `sha` variable is None or a string in the case of a fallback.
- The `isinstance(sha, str)` assertion passes when the fallback is necessary.
- The function does not raise an exception when the fallback is necessary.
- The function still works after clearing the `_git_info` cache.

Confidence: 80%

Tokens: 253 input + 199 output = 452 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	3 lines (ranges: 127-128, 130)

PASSED `tests/test_report_writer_coverage.py::TestGetPluginGitInfo::test_plugin_git_info_returns_values` 1ms 3

AI ASSESSMENT

Scenario: test_get_plugin_git_info

Why Needed: to ensure plugin git info returns some values

Key Assertions:

- {'name': 'assert sha is not None or isinstance(sha, str)', 'description': "Test that the function returns a non-None value for sha and a string if it's dirty"}

Confidence: 80%

Tokens: 141 input + 84 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	3 lines (ranges: 127-128, 130)

PASSED `tests/test_report_writer_coverage.py::TestReportWriterAtomicWrite::test_atomic_write_fallback` 12ms 5

AI ASSESSMENT

Scenario: Test atomic write fallback

Why Needed: To ensure that the ReportWriter can handle unexpected errors during an atomic write operation.

Key Assertions:

- {'name': 'Expected report to be written with correct content', 'description': 'The report should contain the expected key assertions.', 'expected_value': {'scenario': 'Test atomic write fallback', 'why_needed': 'To ensure that the ReportWriter can handle unexpected errors during an atomic write operation.'}}

Confidence: 80%

Tokens: 181 input + 112 output = 293 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	80 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	122 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_report_writer_coverage.py::TestReportWriterPDF::test_pdf_playwright_exception` 123ms 6

AI ASSESSMENT

Scenario: Test PDF generation with playwright exception when browser launch fails.

Why Needed: Prevents test from passing if playwright raises an exception during PDF generation.

Key Assertions:

- Mocked playwright context should raise a RuntimeError when browser launch fails.
- Writer should have warnings about PDF failure for the mocked playwright context.
- Any warning code in the writer should match 'W201'.
- The test should fail if no warnings are present in the writer's output.
- The test should pass if all warnings are removed or ignored by the writer.
- The writer should not raise an exception when the playwright context is successfully launched and PDF generation proceeds normally.

Confidence: 80%

Tokens: 356 input + 149 output = 505 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	26 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65-67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	28 lines (ranges: 156-158, 408, 417, 419, 421-423, 431-436, 439, 441-442, 455, 460, 462, 465-469, 477-478)

PASSED `tests/test_report_writer_coverage.py::TestReportWriterPDF::test_pdf_playwright_not_installed` 1ms 4

AI ASSESSMENT

Scenario: Test PDF generation when playwright is not installed.

Why Needed: Prevents a potential bug where the report writer fails to create a PDF file due to playwright being missing.

Key Assertions:

- The 'W204' warning should be present in the warnings list.
- The PDF file 'report.pdf' should not exist.
- The 'ReportWriter' instance does not raise an exception when playwright is not installed.

Confidence: 80%

Tokens: 293 input + 101 output = 394 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	9 lines (ranges: 156-158, 408-412, 415)

PASSED `tests/test_report_writer_coverage.py::TestReportWriterPDF::test_resolve_html_source_creates_temp` 37ms 6

AI ASSESSMENT

Scenario: Verify that _resolve_pdf_html_source creates a temporary file when no HTML source is provided.

Why Needed: Prevents regression where the test fails due to missing or empty HTML source configuration.

Key Assertions:

- The report writer creates a new temporary file with the suffix '.html'.
- The path of the created temporary file exists and matches the expected extension.
- The path of the temporary file does not contain any non-existent files (i.e., no actual HTML content).

Confidence: 80%

Tokens: 265 input + 115 output = 380 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	26 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65-67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	11 lines (ranges: 156-158, 455, 460, 462, 465-469)

PASSED `tests/test_report_writer_coverage.py::TestReportWriterPDF::test_resolve_html_source_missing_html_file` 37ms 6

AI ASSESSMENT

Scenario: Verify that the test resolves an HTML source when a missing HTML file exists.

Why Needed: Prevents regression in case of missing HTML files, ensuring correct PDF generation.

Key Assertions:

- The report writer should be able to resolve the HTML source and generate a PDF.
- The resolved path should exist and not point to an empty directory.
- The test should fail when the file does not exist, indicating a bug in the configuration or reporting logic.
- The temporary file created by the test should be deleted after use.
- The report writer should handle missing HTML files correctly, falling back to a temporary file if necessary.

Confidence: 80%

Tokens: 270 input + 145 output = 415 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	26 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65-67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	13 lines (ranges: 156-158, 455-457, 460, 462, 465-469)

PASSED `tests/test_report_writer_coverage.py::TestReportWriterPDF::test_resolve_html_source_uses_existing` 1ms 4

AI ASSESSMENT

Scenario: The test verifies that the _resolve_pdf_html_source method uses an existing HTML file as its source.

Why Needed: This test prevents a potential bug where the method does not find any existing HTML files and therefore cannot resolve the PDF.

Key Assertions:

- The function writes to the specified HTML path.
- The function returns False indicating that is_temp is True.
- The function writes to an existing HTML file.
- The function checks if the path matches the expected html_path.
- The function does not write to a non-existent HTML file.
- The function returns True indicating that is_temp is False.

Confidence: 80%

Tokens: 266 input + 142 output = 408 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	7 lines (ranges: 156-158, 455-458)

📄 tests/test_schemas.py

2 tests

PASSED tests/test_schemas.py::TestAnnotationSchema::test_from_dict_full 1ms 3

AI ASSESSMENT

Scenario: Test that `AnnotationSchema.from_dict` can create a full annotation from a dictionary with all required fields.

Why Needed: Prevents regression in case of missing required fields, ensuring the validation logic works correctly.

Key Assertions:

- assert schema.scenario == 'Verify login'
- assert schema.why_needed == 'Catch auth bugs'
- assert schema.key_assertions == ['assert 200', 'assert token']
- assert schema.confidence == 0.95

Confidence: 80%

Tokens: 276 input + 115 output = 391 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/schemas.py	5 lines (ranges: 77-81)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_schemas.py::TestAnnotationSchema::test_to_dict_full 1ms 3

AI ASSESSMENT

Scenario: test_to_dict_full verifies that the AnnotationSchema can convert to a dictionary with all required fields.

Why Needed: This test prevents regression bugs in the AnnotationSchema where it may not be able to generate a full dictionary representation of an annotation.

Key Assertions:

- assert data['scenario'] == 'Verify login',
- assert data['why_needed'] == 'Catch auth bugs',
- assert data['key_assertions'] == ['assert 200', 'assert token'],
- assert data['confidence'] == 0.95

Confidence: 80%

Tokens: 273 input + 128 output = 401 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/schemas.py	8 lines (ranges: 90-92, 94-98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_smoke_pytester.py

15 tests

PASSED `tests/test_smoke_pytester.py::TestBasicReportGeneration::test_html_report_created` 100ms 8

AI ASSESSMENT

Scenario: The HTML report is generated correctly.

Why Needed: Prevents a potential issue where the test does not produce an HTML report even if the function `test_simple` passes.

Key Assertions:

- The report path exists and contains the expected content.
- The report path exists and contains the string 'test_simple' in its content.
- The report path is a valid file system path.
- The report path does not contain any other HTML tags than '
- The report path does not contain any other text except for the expected content.
- The test function `test_simple` passes correctly.

Confidence: 80%

Tokens: 264 input + 147 output = 411 total

COVERAGE

src/pytest_llm_report/collector.py	39 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482-484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)

src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	106 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_smoke_pytester.py::TestBasicReportGeneration::test_html_summary_counts_all_statuses` 145ms 8

AI ASSESSMENT

Scenario: test_html_summary_counts_all_statuses verifies that HTML summary counts include all statuses.

Why Needed: This test prevents regression where the HTML summary counts do not include all statuses, such as when there are multiple failed tests or skipped tests.

Key Assertions:

- assert True is called for each label in the labels list
- assert int(match.group(1)) == expected for match in re.findall(card_pattern, html)
- assert int(match.group(1)) == expected for match in re.findall(fallback_pattern, html)
- assert_summary(['Total Tests', 'Total'], 6) is not called
- assert_summary(['Passed'], 1) is called and asserts True
- assert_summary(['Failed'], 1) is called and asserts False
- assert_summary(['Skipped'], 1) is called and asserts True
- assert_summary(['XFailed'], 1) is called and asserts False
- assert_summary(['XPassed'], 1) is called and asserts True
- assert_summary(['Errors', 'Error'], 1) is called and asserts True

Confidence: 80%

Tokens: 621 input + 242 output = 863 total

COVERAGE

src/pytest_llm_report/collector.py	69 lines (ranges: 78-79, 90, 93-94, 96, 99-104, 106-107, 109-112, 114-119, 121-122, 124, 127, 132-133, 140-141, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 212-214, 216, 227-228, 230-236, 250-251, 261, 264, 268, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482-484, 486, 488,

	490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	116 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-335, 337, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_smoke_pytester.py::TestBasicReportGeneration::test_json_report_created` 73ms 7

AI ASSESSMENT

Scenario: The JSON report is created and contains the expected schema version, summary statistics, and test counts.

Why Needed: This test prevents a regression where the report generation process fails to create a valid JSON file with the required metadata.

Key Assertions:

- The `schema_version` key in the report data should be set to '1.0'.
- The `summary` object should contain exactly two keys: `total` and `passed`.
- The `summary` object should have a single key-value pair for `failed`, with a value of 1.
- The number of tests passed should match the total count in the report.
- The number of failed tests should match the failed count in the report.
- The JSON data should be well-formed and contain only valid JSON syntax.

Confidence: 80%

Tokens: 295 input + 180 output = 475 total

COVERAGE

src/pytest_llm_report/collector.py	55 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-118, 124, 127, 132-133, 140-141, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 227-228, 230-236, 261, 264, 268, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	76 lines (ranges: 190, 194-199, 201-203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488,

	490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	112 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-327, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_smoke_pytester.py::TestBasicReportGeneration::test_llm_annotations_in_report` 66ms 14

AI ASSESSMENT

Scenario: Verify that LLM annotations are included in the report when a provider is enabled.

Why Needed: Prevent regressions by ensuring LLM annotations are present in the report.

Key Assertions:

- The function `test_pass()` passes without any errors.
- The `pytester.makepyfile` call returns a valid test file.
- The `pytester.makeconftest` call creates a conftest that patches litellm.completion before it's imported.
- The `pytester.makefile` call creates a pyproject.toml configuration with the [tool.pytest_llm_report] setting.
- The `pytester.makepyfile` call returns a valid test file after the patch is applied.
- The `pytester.makeconftest` call sets up the conftest to use the patched completion function.
- The `pytester.makefile` call creates a pyproject.toml configuration with the [tool.pytest_llm_report] setting.
- The `pytester.makepyfile` call returns a valid test file after the patch is applied and the config is set up.

Confidence: 80%

Tokens: 385 input + 246 output = 631 total

COVERAGE

src/pytest_llm_report/cache.py	20 lines (ranges: 39-41, 53, 55-56, 86, 90, 92, 94, 97-101, 103, 118-119, 121, 153)
src/pytest_llm_report/collector.py	39 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/llm/annotator.py	96 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-89, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188, 199-206, 213-219, 221, 223, 249-252, 254-255, 257-258, 260,

	262, 264, 269-274, 277-279, 281, 283-284, 289-290, 292-295, 298, 303)
src/pytest_llm_report/llm/base.py	55 lines (ranges: 65-66, 87-89, 97, 105, 134, 137-138, 155, 163, 174, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357, 384, 386, 388, 391, 396-397, 399)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/llm/litellm_provider.py	43 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95-96, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-187, 190, 192-193, 196, 204, 213, 221-222, 224, 227-229, 242-243, 245)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/models.py	103 lines (ranges: 130-133, 135-137, 139, 141, 143, 190, 194-199, 201, 203, 205, 207, 210, 212-214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419-437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	136 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-325, 327-328, 332-336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543,

	547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	316 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362-364, 366-367, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-494, 497, 499, 502-506, 509, 512-514, 516-517, 523-531, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/prompts.py	29 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-109, 111-112, 116)
src/pytest_llm_report/report_writer.py	115 lines (ranges: 55, 67-73, 85-86, 98-99, 102, 105-108, 113, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-296, 298-299, 301-302, 304-305, 307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_smoke_pytester.py::TestBasicReportGeneration::test_llm_error_is_reported` 105ms 14

AI ASSESSMENT

Scenario: Test that LLM errors are surfaced in HTML output.

Why Needed: This test prevents a regression where LLM errors might not be reported in the expected format.

Key Assertions:

- The function `test_llm_error_is_reported` should raise an error when executed with the given input.
- The error message should include the string 'boom'.
- The HTML output of the test should contain a table with the error details.
- The table columns should be labeled correctly (e.g., 'Error Type', 'Error Message').
- The error type column should display the correct value ('LLM Error').
- The error message column should display the correct text ('boom').
- The test output should include a descriptive title indicating that an LLM error occurred.
- The test output should contain a link to the LLM error report (if available).

Confidence: 80%

Tokens: 313 input + 198 output = 511 total

COVERAGE

src/pytest_llm_report/cache.py	12 lines (ranges: 39-41, 53, 55-56, 86, 88, 118-119, 121, 153)
src/pytest_llm_report/collector.py	39 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/llm/annotator.py	100 lines (ranges: 47, 50-51, 58-59, 65, 67, 70, 73-74, 76, 84, 86-89, 95-96, 98-99, 106-108, 112-113, 116, 121-122, 132, 134, 137-141, 144-151, 181-182, 184, 186, 188, 199-206, 213-219, 221-223, 249-252, 254-255, 257-258, 260, 262, 264, 269-274, 277-279, 281, 283-284, 289-290, 292-295, 298-301, 303)

src/pytest_llm_report/llm/base.py	37 lines (ranges: 65-66, 87-89, 97, 105, 134, 137-138, 155, 163, 174, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 384, 386, 388, 391, 396-397, 399)
src/pytest_llm_report/llm/batching.py	33 lines (ranges: 34, 39, 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136, 156-157, 160, 162, 181-185, 187-188, 190, 224)
src/pytest_llm_report/llm/litellm_provider.py	44 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95-96, 100-101, 104, 106-107, 112, 114, 116-117, 120, 135, 137, 170-174, 176-178, 182, 186-187, 190, 221-222, 224, 227-229, 242-243, 245)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	136 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-325, 327-328, 332-336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482-484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	316 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309,

	311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362-364, 366-367, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-494, 497, 499, 502-507, 512-514, 516-517, 523-531, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/prompts.py	29 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-109, 111-112, 116)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	111 lines (ranges: 55, 67-73, 85-86, 98-99, 102, 105-108, 113, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-296, 298-299, 301-302, 304-305, 307, 319, 321-322, 324-325, 337, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

PASSED tests/test_smoke_pytester.py::TestMarkers::test_llm_opt_out_marker 63ms 7

AI ASSESSMENT

Scenario: Verify that the LLM opt-out marker is correctly recorded in the test report.

Why Needed: This test prevents regression where a test might not record the LLM opt-out marker due to a missing or incorrect configuration.

Key Assertions:

- The 'llm_opt_out' marker should be present in the test report.
- The 'llm_opt_out' marker should be set to True for this test.
- There should be only one test associated with the 'llm_opt_out' marker in the report.
- The 'llm_opt_out' marker should not be False for this test.

Confidence: 80%

Tokens: 290 input + 140 output = 430 total

COVERAGE

src/pytest_llm_report/collector.py	40 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181-182, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	76 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214-216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-

	594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	110 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED tests/test_smoke_pytester.py::TestMarkers::test_requirement_marker 62ms 7

AI ASSESSMENT

Scenario: Test the requirement marker functionality.

Why Needed: This test prevents a potential bug where the requirement marker is not recorded correctly, potentially leading to incorrect reporting or analysis of tests.

Key Assertions:

- The `pytest.mark.requirement` decorator should be applied to the `test_with_req` function with both 'REQ-001' and 'REQ-002' requirements.
- The `reqs` list in the test data should contain both 'REQ-001' and 'REQ-002'.
- The requirement string 'REQ-001' should be present in the `reqs` list of the first test in the report.

Confidence: 80%

Tokens: 307 input + 145 output = 452 total

COVERAGE

src/pytest_llm_report/collector.py	40 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-200, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	76 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222-224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-

	594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	110 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_smoke_pytester.py::TestOutcomes::test_multiple_xfail_outcomes` 69ms 7

AI ASSESSMENT

Scenario: Test that multiple xfailed tests are recorded in the report.

Why Needed: This test prevents regression by ensuring that all xfailed tests are properly reported and counted.

Key Assertions:

- The number of xfailed tests is correctly reported as 2.
- All xfailed tests are included in the report.
- Each xfailed test has an outcome of 'xfailed'.
- No other outcomes are present in the report.
- The report path contains a file named 'report.json' with the correct format.
- The JSON data is correctly parsed and loaded into memory.

Confidence: 80%

Tokens: 317 input + 136 output = 453 total

COVERAGE

src/pytest_llm_report/collector.py	47 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-116, 119, 121-122, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 212, 216, 250-251, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	76 lines (ranges: 190, 194-199, 201-203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-

	522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	113 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324, 326, 328, 330-331, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED tests/test_smoke_pytester.py::TestOutcomes::test_skip_outcome 65ms 7

AI ASSESSMENT

Scenario: Test that skipping tests prevents the 'skip' marker from being recorded.

Why Needed: This test prevents regression in case of a skipped test, ensuring that the expected number of skipped tests is reported correctly.

Key Assertions:

- The value of `data['summary']['skipped']` should be equal to 1 when the 'skip' marker is applied.
- The 'summary' key should exist in the report data and its value should be an integer.
- The 'skipped' key should contain a single integer value, which represents the number of skipped tests.
- The 'summary' dictionary should have a 'skipped' key with a string value equal to 1.
- The 'report.json' file should contain a 'summary' section with a 'skipped' key and its corresponding value.
- The 'json.loads()' method should successfully parse the report data from the 'report.json' file.

Confidence: 80%

Tokens: 264 input + 206 output = 470 total

COVERAGE

src/pytest_llm_report/collector.py	43 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 106-107, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 250-251, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	76 lines (ranges: 190, 194-199, 201-203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322,

	460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	112 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324, 326, 328-329, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_smoke_pytester.py::TestOutcomes::test_xfail_outcome` 68ms 7

AI ASSESSMENT

Scenario: Test that xfailed tests are recorded and reported correctly.

Why Needed: This test prevents regression in the reporting of failed tests, ensuring accurate tracking of xfailed tests.

Key Assertions:

- The number of xfailed tests is exactly 1 as recorded in the report.json file.
- The value of `xfailed` in the `summary` dictionary matches the actual count of xfailed tests.
- The test data contains a single failed test with the label 'xfailed'.
- The report.json file includes an entry for the xfailed test, indicating its status.
- The number of xfailed tests is not affected by changes to the test code or environment.
- The test does not fail when run without pytester or any other dependencies.
- The test can be run multiple times with different inputs and still produce the same report.json file.

Confidence: 80%

Tokens: 264 input + 192 output = 456 total

COVERAGE

src/pytest_llm_report/collector.py	47 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-116, 119, 121-122, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 212, 216, 250-251, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	76 lines (ranges: 190, 194-199, 201-203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488,

	490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	113 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324, 326, 328, 330-331, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_smoke_pytester.py::TestParametrization::test_parametrized_tests` 66ms 7

AI ASSESSMENT

Scenario: Test Parametrized Tests: Verify that parameterized tests are recorded separately and their results are reported correctly.

Why Needed: This test prevents regression by ensuring that the same test is run multiple times with different inputs, which can lead to false negatives if the test is not properly configured.

Key Assertions:

- The 'pytester.makepyfile' function creates a pytest file with parameterized tests.
- The 'pytester.runpytest' function runs the tests and reports their results in a JSON format.
- The 'json.loads' function parses the report JSON and verifies that it contains the correct information.
- The test asserts that the total number of tests is 3 (since there are three input values) and that all tests pass (since all assertions pass).
- The test also asserts that the passed count matches the expected value (three passes).

Confidence: 80%

Tokens: 290 input + 192 output = 482 total

COVERAGE

src/pytest_llm_report/collector.py	40 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163-164, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	76 lines (ranges: 190, 194-199, 201, 203-205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322,

	460, 463, 466, 470, 472-473, 476-477, 482, 484-486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/report_writer.py	110 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED `tests/test_smoke_pytester.py::TestPluginRegistration::test_help_contains_examples` 55ms 3

AI ASSESSMENT

Scenario: tests/test_smoke_pytester.py::TestPluginRegistration::test_help_contains_examples

Why Needed: This test is necessary to ensure that the CLI help text includes usage examples for the plugin registration feature.

Key Assertions:

- {'assertion_name': 'CLI help text includes usage examples', 'expected_result': 'The CLI help text should include usage examples for the plugin registration feature.', 'actual_result': 'True'}

Confidence: 80%

Tokens: 123 input + 106 output = 229 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	89 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	240 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349,

```
351, 354-355, 362, 371-373,
558-559, 562, 566-568)
```

PASSED `tests/test_smoke_pytester.py::TestPluginRegistration::test_markers_registered` 49ms 3

AI ASSESSMENT

Scenario: TestPluginRegistration

Why Needed: To verify that LLM markers are registered.

Key Assertions:

- {'name': 'Markers are registered', 'description': 'LLM markers should be present in the test suite.', 'expected_result': 'True'}
- {'name': 'Markers match expected lines', 'description': 'The output of `pytest --markers` should contain the expected marker lines.', 'expected_result': 'True'}

Confidence: 80%

Tokens: 142 input + 115 output = 257 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	89 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	240 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349,

```
351, 354-355, 362, 371-373,
558-559, 562, 566-568)
```

PASSED `tests/test_smoke_pytester.py::TestPluginRegistration::test_plugin_registered` 56ms 3

AI ASSESSMENT

Scenario: tests/test_smoke_pytester.py::TestPluginRegistration::test_plugin_registered

Why Needed: To verify that the plugin is registered correctly.

Key Assertions:

- {'assertion_type': 'contains', 'pattern': '--llm-report*', 'expected_value': ''}

Confidence: 80%

Tokens: 118 input + 77 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	89 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	240 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 558-559, 562, 566-568)

PASSED `tests/test_smoke_pytester.py::TestSpecialCharacters::test_special_chars_in_nodeid` 107ms 8

AI ASSESSMENT

Scenario: Verify that special characters in nodeid do not cause Pytest to crash or produce invalid HTML reports.

Why Needed: This test prevents a potential regression where special characters in nodeids might cause Pytest to fail or produce corrupted report files.

Key Assertions:

- The presence of special characters in the nodeid does not cause Pytest to crash.
- The HTML report generated by Pytest is valid and contains the '
- The 'report.html' file exists after running the test.
- The content of the 'report.html' file includes the '
- The presence of special characters in nodeids does not prevent Pytest from generating a report with a valid HTML structure.

Confidence: 80%

Tokens: 288 input + 162 output = 450 total

COVERAGE

src/pytest_llm_report/collector.py	40 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163-164, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482-484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)

src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	106 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

📄 tests/test_time.py

15 tests

PASSED `tests/test_time.py::TestFormatDuration::test_boundary_one_minute` 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_boundary_one_minute

Why Needed: To ensure the `format_duration` function correctly formats durations in one minute.

Key Assertions:

- {'expected_value': '1m 0.0s', 'actual_value': '1m 0.0s'}

Confidence: 80%

Tokens: 106 input + 84 output = 190 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	6 lines (ranges: 39, 41, 43, 46-48)

PASSED `tests/test_time.py::TestFormatDuration::test_microseconds_format` 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_microseconds_format

Why Needed: To ensure that the `format_duration` function correctly formats sub-millisecond durations as microseconds.

Key Assertions:

- {'description': "The result contains 'µs' (microseconds) assertion", 'expected': '500µs'}

Confidence: 80%

Tokens: 121 input + 86 output = 207 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	2 lines (ranges: 39-40)

PASSED tests/test_time.py::TestFormatDuration::test_milliseconds_format 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_milliseconds_format

Why Needed: To ensure that the `format_duration` function correctly formats sub-second durations as milliseconds.

Key Assertions:

- {'name': "result contains 'ms'", 'expected': '500.0ms'}

Confidence: 80%

Tokens: 119 input + 78 output = 197 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	3 lines (ranges: 39, 41-42)

PASSED tests/test_time.py::TestFormatDuration::test_minutes_format 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_minutes_format

Why Needed: To ensure the `format_duration` function correctly formats durations over a minute, including minutes and seconds.

Key Assertions:

- {'description': "The result should contain 'm' (minutes) in its string representation.", 'expected_value': '1m'}
- {'description': 'The result should be equal to the expected value.', 'expected_value': '1m 30.5s'}

Confidence: 80%

Tokens: 124 input + 122 output = 246 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	6 lines (ranges: 39, 41, 43, 46-48)

PASSED tests/test_time.py::TestFormatDuration::test_multiple_minutes 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_multiple_minutes

Why Needed: To ensure the `format_duration` function correctly formats multiple minutes into a human-readable string.

Key Assertions:

- {'name': 'result', 'expected': '3m 5.0s'}

Confidence: 80%

Tokens: 112 input + 78 output = 190 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	6 lines (ranges: 39, 41, 43, 46-48)

PASSED tests/test_time.py::TestFormatDuration::test_one_second 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_one_second

Why Needed: To ensure the `format_duration` function correctly formats a duration of exactly one second as '1.00s'.

Key Assertions:

- {'description': "The result of calling format_duration(1.0) should be equal to '1.00s'.", 'expected_value': '1.00s'}

Confidence: 80%

Tokens: 101 input + 102 output = 203 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	4 lines (ranges: 39, 41, 43-44)

PASSED tests/test_time.py::TestFormatDuration::test_seconds_format 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_seconds_format

Why Needed: To ensure that the `format_duration` function correctly formats seconds under a minute.

Key Assertions:

- {'expected': '5.50s', 'actual': '5.50s'}

Confidence: 80%

Tokens: 110 input + 76 output = 186 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	4 lines (ranges: 39, 41, 43-44)

PASSED tests/test_time.py::TestFormatDuration::test_small_milliseconds 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_small_milliseconds

Why Needed: To ensure the `format_duration` function correctly formats small millisecond durations.

Key Assertions:

- {'name': 'result', 'expected_value': '1.0ms'}

Confidence: 80%

Tokens: 111 input + 74 output = 185 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	3 lines (ranges: 39, 41-42)

PASSED `tests/test_time.py::TestFormatDuration::test_very_small_microseconds` 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_very_small_microseconds

Why Needed: To ensure that the `format_duration` function correctly formats very small durations as microseconds.

Key Assertions:

- {'name': 'Expected result', 'value': '1µs'}

Confidence: 80%

Tokens: 116 input + 77 output = 193 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	2 lines (ranges: 39-40)

PASSED `tests/test_time.py::TestIsoFormat::test_formats_datetime_with_utc` 1ms 3

AI ASSESSMENT

Scenario: Test ISO Format with UTC

Why Needed: To verify the correct formatting of datetime objects with UTC timezone.

Key Assertions:

- {'name': 'Expected output', 'value': '2024-01-15T10:30:45+00:00'}

Confidence: 80%

Tokens: 143 input + 75 output = 218 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	1 lines (ranges: 27)

PASSED `tests/test_time.py::TestIsoFormat::test_formats_naive_datetime` 1ms 3

AI ASSESSMENT

Scenario: Tests for ISO Format

Why Needed: To ensure the naive datetime format is correct without timezone.

Key Assertions:

- {'name': 'Expected result', 'value': '2024-06-20T14:00:00'}

Confidence: 80%

Tokens: 136 input + 69 output = 205 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	1 lines (ranges: 27)

PASSED `tests/test_time.py::TestIsoFormat::test_formats_with_microseconds` 1ms 3

AI ASSESSMENT

Scenario: Test IsoFormat with microseconds

Why Needed: To test the functionality of formatting datetime objects with microseconds.

Key Assertions:

- {'name': 'Expected value in result', 'value': '123456'}

Confidence: 80%

Tokens: 133 input + 62 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	1 lines (ranges: 27)

PASSED tests/test_time.py::TestUtcNow::test_has_utc_timezone 1ms 3

AI ASSESSMENT

Scenario: Test that the current time has a valid UTC timezone.

Why Needed: To ensure that the `datetime` object returned by `utc_now()` has a valid timezone.

Key Assertions:

- {'name': 'assert result.tzinfo is not None', 'description': 'The test asserts that the result of `utc_now()` has a timezone info.', 'expected_result': 'True'}
- {'name': 'assert result.tzinfo == UTC', 'description': 'The test asserts that the result of `utc_now()` has a valid UTC timezone.', 'expected_result': 'UTC'}

Confidence: 80%

Tokens: 109 input + 147 output = 256 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	1 lines (ranges: 15)

PASSED tests/test_time.py::TestUtcNow::test_is_current_time 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestUtcNow::test_is_current_time

Why Needed: To ensure that the `utc_now` function returns a time within a reasonable tolerance of the current UTC time.

Key Assertions:

- {'name': 'before <= result <= after', 'description': 'The before and after times should be within a reasonable tolerance of each other.'}

Confidence: 80%

Tokens: 116 input + 94 output = 210 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	1 lines (ranges: 15)

PASSED tests/test_time.py::TestUtcNow::test_returns_datetime 1ms 3

AI ASSESSMENT

Scenario: tests/test_time.py::TestUtcNow::test_returns_datetime

Why Needed: This test ensures that the function `utc_now()` returns a datetime object.

Key Assertions:

- {'name': 'Type of result', 'expected': 'datetime', 'actual': 'isinstance(result, datetime)'}

Confidence: 80%

Tokens: 94 input + 82 output = 176 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	1 lines (ranges: 15)

tests/test_token_refresh.py

12 tests

PASSED `tests/test_token_refresh.py::TestTokenRefresher::test_command_failure` 1ms 3

AI ASSESSMENT

Scenario: When TokenRefresher raises an error on command failure, then the test verifies that it correctly returns a TokenRefreshError with appropriate error message.

Why Needed: This test prevents potential regression where the TokenRefresher might not raise an error when encountering a command failure, potentially causing unexpected behavior or errors later in the application.

Key Assertions:

- The function `get_token()` of the `TokenRefresher` object should return a `TokenRefreshError` with the message 'Authentication failed'.
- The string 'exit 1' should be present in the error message returned by `get_token()`.
- The string 'Authentication failed' should be present in the error message returned by `get_token()`.

Confidence: 80%

Tokens: 310 input + 161 output = 471 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	20 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101-104, 113, 115)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_token_refresh.py::TestTokenRefresher::test_empty_output 1ms 3

AI ASSESSMENT

Scenario: Verify TokenRefresher raises error on empty output when no token is available.

Why Needed: This test prevents a potential bug where the TokenRefresher does not raise an error when there is no token to refresh.

Key Assertions:

- The `get_token()` method of the `TokenRefresher` class should raise a `TokenRefreshError` exception with the message 'empty output'.
- The `stdout` and `stderr` attributes of the `get_token()` method should be empty strings.
- The `returncode` attribute of the `get_token()` method should be 0 (indicating successful execution).

Confidence: 80%

Tokens: 297 input + 144 output = 441 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	20 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101, 107-109, 113, 115)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_token_refresh.py::TestTokenRefresher::test_force_refresh 1ms 3

AI ASSESSMENT

Scenario: Test that forcing a refresh bypasses the cache and returns a new token.

Why Needed: This test prevents a regression where the TokenRefresher does not return a new token when forced to refresh.

Key Assertions:

- The function `get_token()` of the `TokenRefresher` class returns the expected token value for both `token1` and `token2` after force refresh.
- The number of calls to the `run` method of the subprocess is equal to 2, which matches the expected behavior when forcing a refresh.
- Both `token1` and `token2` have different values than before the force refresh.
- The output of the subprocess contains the token value with the correct index (e.g. 'token-1' for `token1` and 'token-2' for `token2`).
- The error message from the subprocess is empty.
- The return code of the subprocess is 0, indicating successful execution.
- The output format of the subprocess is set to 'text'.

Confidence: 80%

Tokens: 346 input + 230 output = 576 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	25 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh.py::TestTokenRefresher::test_get_token_json_custom_key` 1ms 3

AI ASSESSMENT

Scenario: Test that the `TokenRefresher` uses a custom JSON key for token refresh.

Why Needed: This test prevents a potential issue where the default JSON key is used instead of a custom one.

Key Assertions:

- The `get_token()` method returns the expected custom JSON key value.
- The `access_token` attribute of the returned object matches the custom key value.
- The `json_key` parameter passed to the `TokenRefresher` constructor is used correctly.

Confidence: 80%

Tokens: 303 input + 115 output = 418 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	29 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132-135, 139, 143-144, 148)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh.py::TestTokenRefresher::test_get_token_json_format` 1ms 3

AI ASSESSMENT

Scenario: The test verifies that the `TokenRefresher` extracts a JSON object containing the extracted token from the expected JSON output.

Why Needed: This test prevents a potential bug where the `TokenRefresher` does not extract the token correctly if the output format is set to 'json'.

Key Assertions:

- The output of the `get-token` command should be a JSON object with the following structure: `{'token': 'json-token-value', 'expires_in': 3600}`.
- The extracted token should match the expected value: `

Confidence: 80%

Tokens: 308 input + 131 output = 439 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	29 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132-135, 139, 143-144, 148)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh.py::TestTokenRefresher::test_get_token_text_format` 1ms 3

AI ASSESSMENT

Scenario: The test verifies that the `TokenRefresher` class correctly extracts a token from text output when the `output_format` is set to 'text'.

Why Needed: This test prevents a potential bug where the extracted token is not in the expected format, potentially leading to incorrect usage or unexpected behavior.

Key Assertions:

- The extracted token matches the provided input string 'my-secret-token'.
- The output of `subprocess.run` contains the correct text after processing.
- The error message from `subprocess.run` does not indicate an issue with the token extraction process.

Confidence: 80%

Tokens: 298 input + 135 output = 433 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	25 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_token_refresh.py::TestTokenRefresher::test_invalid_json 1ms 3

AI ASSESSMENT

Scenario: The test verifies that the TokenRefresher raises an error when it encounters invalid JSON.

Why Needed: This test prevents a potential bug where the TokenRefresher incorrectly interprets valid JSON as a token refresh request.

Key Assertions:

- The `get_token()` method of the `TokenRefresher` instance should raise a `TokenRefreshError` exception when it encounters invalid JSON.
- The error message returned by the `get_token()` method should indicate that the input is not valid JSON.
- The test should fail when running the `get_token()` method on an invalid JSON string.
- The test should pass when running the `get_token()` method on a valid JSON string.
- The error message returned by the `get_token()` method should be in lowercase (e.g. 'json' instead of 'JSON'),
- The test should only fail when running the `get_token()` method on an invalid JSON string, not on other types of exceptions.
- The test should pass for all other valid input strings.

Confidence: 80%

Tokens: 299 input + 234 output = 533 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	25 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101, 107-108, 111, 113, 115, 132-134, 149-150)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_token_refresh.py::TestTokenRefresher::test_invalidate 1ms 3

AI ASSESSMENT

Scenario: Test TokenRefresher.invalidate() clears cache and updates the token count correctly.

Why Needed: This test prevents a potential bug where the TokenRefresher does not update the token count after calling invalidate().

Key Assertions:

- The function `invalidate()` of the `TokenRefresher` class should clear the cache by setting the call_count to 1.
- The function `invalidate()` of the `TokenRefresher` class should update the token count by setting it to 2 after calling `get_token()`.
- The function `invalidate()` of the `TokenRefresher` class should not have any side effects on the token count.
- The function `invalidate()` of the `TokenRefresher` class should clear the cache even if an exception is raised during execution.
- The function `invalidate()` of the `TokenRefresher` class should update the token count after calling `get_token()` regardless of whether an exception is raised or not.

Confidence: 80%

Tokens: 340 input + 211 output = 551 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	28 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156, 160-162)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh.py::TestTokenRefresher::test_missing_json_key` 1ms 3

AI ASSESSMENT

Scenario: Test that TokenRefresher raises an error when the JSON key is missing.

Why Needed: To prevent a potential bug where the TokenRefresher fails to refresh tokens due to a missing required JSON key.

Key Assertions:

- The `TokenRefreshError` exception should be raised with a message indicating that the token was not found.
- The error message should include the word 'not found'.
- The error message should be case-insensitive (i.e., it should match 'Not Found' regardless of the original casing).

Confidence: 80%

Tokens: 325 input + 125 output = 450 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	28 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101, 107-108, 111, 113, 115, 132-135, 139-141, 149)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh.py::TestTokenRefresher::test_thread_safety` 52ms 3

AI ASSESSMENT

Scenario: Test TokenRefresher thread safety by starting multiple threads concurrently and verifying that they all retrieve the same token.

Why Needed: This test prevents a potential bug where multiple threads accessing the TokenRefresher instance simultaneously could result in inconsistent or incorrect results due to race conditions.

Key Assertions:

- The function `get_token()` should return the same token for each thread that acquires the lock.
- Each thread should acquire the lock before executing the `get_token()` method and retrieve the same token.
- If multiple threads start retrieving tokens concurrently, they should all return the same token.

Confidence: 80%

Tokens: 427 input + 134 output = 561 total

COVERAGE

`src/pytest_llm_report/collector.py`	`14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)`
`src/pytest_llm_report/llm/token_refresh.py`	`28 lines (ranges: 59-60, 63-66, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156)`
`src/pytest_llm_report/plugin.py`	`6 lines (ranges: 558-559, 562, 566-568)`

PASSED tests/test_token_refresh.py::TestTokenRefresher::test_timeout_handling 1ms 3

AI ASSESSMENT

Scenario: The test verifies that the TokenRefresher handles command timeouts correctly.

Why Needed: This test prevents a potential bug where the TokenRefresher fails to refresh tokens when they timeout.

Key Assertions:

- The `get_token()` method raises a `TokenRefreshError` with the message 'timed out' after a timeout of 30 seconds.
- The `get_token()` method does not raise an exception or log an error when the token refresh times out.

Confidence: 80%

Tokens: 279 input + 111 output = 390 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	16 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 113-114)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh.py::TestTokenRefresher::test_token_caching` 1ms 3

AI ASSESSMENT

Scenario: Test that TokenRefresher caches tokens and doesn't call the command again.

Why Needed: Prevents a potential bug where multiple requests to get a token would result in the same cached token being returned.

Key Assertions:

- The function `get_token()` returns the expected token value.
- The output of `refresher.get_token()` is the same as the input, indicating caching.
- The number of calls to `refresher.get_token()` is correct (1 in this case).
- The cached token has not changed even after multiple requests.
- The command 'get-token' is called only once with a valid token.
- The output format is set to 'text'.

Confidence: 80%

Tokens: 353 input + 159 output = 512 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	28 lines (ranges: 59-60, 63-66, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_token_refresh_coverage.py

9 tests

PASSED `tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_command_failure_no_stderr` 1ms 3

AI ASSESSMENT

Scenario: Test the TokenRefresher edge case when command fails with no stderr output.

Why Needed: To prevent a regression where the TokenRefresher does not raise an exception when the command fails without producing any error output.

Key Assertions:

- The `get_token()` method of the `TokenRefresher` instance raises a `TokenRefreshError` with an exit code of 1.
- The `stderr` attribute of the `get_token()` method is set to an empty string, indicating no error output.
- The `stdout` attribute of the `get_token()` method is also set to an empty string, indicating no error output.
- When the command fails without producing any error output, the `TokenRefresher` instance raises a `TokenRefreshError` with an exit code of 1.
- The `exit_code` attribute of the exception raised by the `get_token()` method is set to 1.
- The `stderr` and `stdout` attributes are not changed in the case where the command fails without producing any error output.
- In this scenario, the `TokenRefresher` instance correctly raises a `TokenRefreshError` with an exit code of 1.

Confidence: 80%

Tokens: 322 input + 261 output = 583 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	20 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101-104, 113, 115)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_empty_command_string` 1ms 3

AI ASSESSMENT

Scenario: Token Refresh Coverage

Why Needed: Test case for handling empty command string.

Key Assertions:

- {'name': 'Expected exception type', 'value': 'TokenRefreshError'}
- {'name': 'Expected error message', 'value': 'empty'}

Confidence: 80%

Tokens: 151 input + 77 output = 228 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	11 lines (ranges: 59-60, 63, 69, 83, 85-86, 90-91, 113, 115)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_invalid_command_string` 1ms 3

AI ASSESSMENT

Scenario: Test the test_invalid_command_string function to verify it handles an invalid command string (shlex parse error).

Why Needed: Prevents a potential bug where the TokenRefresher class incorrectly raises a TokenRefreshError when given an invalid command string.

Key Assertions:

- The `TokenRefresher` instance is created with an invalid command string that contains unescaped quotes, which causes a shlex parse error.
- When calling `get_token()` on the `TokenRefresher` instance with an invalid command string, it raises a `TokenRefreshError` exception containing the message 'Invalid command string'.
- The `TokenRefresher` class correctly handles the case where the input command string contains unescaped quotes, preventing the shlex parse error and ensuring the correct behavior.
- The test verifies that the `TokenRefresher` instance is created with an invalid command string that causes a `TokenRefreshError` exception to be raised when calling `get_token()`.

Confidence: 80%

Tokens: 251 input + 213 output = 464 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	11 lines (ranges: 59-60, 63, 69, 83, 85-88, 113, 115)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_json_not_dict` 1ms 3

AI ASSESSMENT

Scenario: Test that the test_json_not_dict scenario verifies a bug preventing handling of non-dict JSON output.

Why Needed: This test prevents regression by ensuring the TokenRefresher handles non-dict JSON output correctly.

Key Assertions:

- The `get_token` method should raise a `TokenRefreshError` when receiving a non-dict JSON output.
- The error message should contain 'Expected JSON object'.
- The error message should contain 'list'.

Confidence: 80%

Tokens: 328 input + 110 output = 438 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	27 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101, 107-108, 111, 113, 115, 132-137, 149)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_json_token_empty_string` 1ms 3

AI ASSESSMENT

Scenario: Test handling when token value is an empty string.

Why Needed: Prevents TestTokenRefresherEdgeCases::test_json_token_empty_string from failing due to unexpected behavior of the TokenRefresher class.

Key Assertions:

- The 'token' key in the JSON output should be a non-empty string.
- The 'token' key in the JSON output should not contain any whitespace characters.
- The error message should indicate that the token value is empty or not a string.
- The test should raise a TokenRefreshError with an appropriate error message when encountering an empty or non-string token value.

Confidence: 80%

Tokens: 324 input + 140 output = 464 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	30 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101, 107-108, 111, 113, 115, 132-135, 139, 143-146, 149)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_json_token_not_string` 1ms 3

AI ASSESSMENT

Scenario: Test handling when token value is not a string.

Why Needed: Prevents the TokenRefresher from attempting to refresh a non-string token value, which could lead to unexpected behavior or errors.

Key Assertions:

- The `json.dumps` function is called with an integer value instead of a string.
- The `returncode` of the subprocess call is set to 0 (success), but the error message does not indicate that the token was invalid.
- The `stderr` parameter is empty, which may indicate that no error occurred or that the token is valid.
- The `json_key` parameter is set to `

Confidence: 80%

Tokens: 326 input + 147 output = 473 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	30 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101, 107-108, 111, 113, 115, 132-135, 139, 143-146, 149)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_oserror_on_execution` 1ms 3

AI ASSESSMENT

Scenario: Test verifies that a TokenRefresher handles an OSError when executing a command.

Why Needed: This test prevents the regression of TokenRefreshError not being raised when executing commands that are not found.

Key Assertions:

- The function `get_token()` raises a `TokenRefreshError` with the message 'Failed to execute' when executed on a nonexistent command.
- The error message includes the string 'Command not found'.
- The function does not raise an exception or return an error code when executing a nonexistent command.

Confidence: 80%

Tokens: 280 input + 122 output = 402 total

COVERAGE

`src/pytest_llm_report/collector.py`	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
`src/pytest_llm_report/llm/token_refresh.py`	19 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 113, 115-118)
`src/pytest_llm_report/plugin.py`	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_text_only_whitespace_lines` 1ms 3

AI ASSESSMENT

Scenario: Test handling when text output has only whitespace lines after initial strip, specifically when parsing with only blank lines.

Why Needed: Prevents a potential bug where the TokenRefresher fails to handle text output with only whitespace lines after an initial strip.

Key Assertions:

- assert 'No non-empty lines' in str(exc_info.value)
- assert 'Only whitespace lines' in str(exc_info.value)
- assert 'Blank line' in str(exc_info.value)
- assert 'Only whitespace content lines' in str(exc_info.value)
- assert 'Non-empty wrapper but only whitespace content lines' in str(exc_info.value)
- assert 'No non-whitespace output' in str(exc_info.value)
- assert 'Output is empty' in str(exc_info.value)

Confidence: 80%

Tokens: 376 input + 174 output = 550 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	4 lines (ranges: 132, 153-155)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED `tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_whitespace_only_command` 1ms 3

AI ASSESSMENT

Scenario: Test the test_whitespace_only_command to ensure it correctly raises a TokenRefreshError for an empty whitespace-only command string.

Why Needed: Prevents a potential bug where the TokenRefresher is not raised with a meaningful error message when given an empty whitespace-only command string.

Key Assertions:

- The function `get_token()` of the `TokenRefresher` object should raise a `TokenRefreshError` exception.
- The error message returned by `get_token()` should contain the word 'empty'.
- The assertion should fail when the input command is an empty string (i.e., no whitespace characters).

Confidence: 80%

Tokens: 236 input + 141 output = 377 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	11 lines (ranges: 59-60, 63, 69, 83, 85-86, 90-91, 113, 115)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

📄 tests/test_token_usage.py

1 tests

PASSED `tests/test_token_usage.py::test_token_usage_aggregation` 5ms 2

AI ASSESSMENT

Scenario: Test token usage aggregation with mock stash and terminal reporter.

Why Needed: Prevents regression in token usage reporting for different test cases.

Key Assertions:

- The total input tokens should be 30 (10 + 20).
- The total output tokens should be 15 (5 + 10).
- The number of annotations should be 2.
- The total tokens should be 45 (15 + 30).

Confidence: 80%

Tokens: 775 input + 107 output = 882 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	73 lines (ranges: 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485-487, 491-494, 497, 499, 502-506, 509, 512-514, 516-521, 523-531, 534-544, 558-559, 562, 566-568)