

Test Report

Run ID: 21201358217-py3.12 • Generated: 2026-01-21 07:48:14 • Duration: 120.44s

Plugin: v0.2.1 (a03dbe622cdc018f89b74731aed91adf1a582867) [dirty]

Repo: v0.2.1 (af30b58f9617c57b114b26afe3e20619a38888d8) [dirty]

LLM: ollama / llama3.2:1b (minimal context, 620 annotated, 2 errors)

Token Usage: 135663 input, 75900 output (Total: 211563)

93.04%

Total Coverage

623

TOTAL TESTS

623

PASSED

0

FAILED

0

SKIPPED

0

XFAILED

0

XPASSED

0

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/ca
che.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

src/pytest_llm_report/co
llector.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, 239
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

src/pytest_llm_report/co
nTEXT_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, 53, 83-84
81-82, 86, 88-90,
93, 96, 108, 111,
124, 126-127,
129-130, 133, 135

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, 91-92, 230, 284, 274-275, 277, 292, 296 279, 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
-----------------------------------	-----	---	-----	--------	---

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, 158, 207, 211, 139, 156-157, 160, 162, 164- 167, 170-176, 181-185, 187-188, 190, 192-194,
---------------------------------------	----	---	----	--------	--

196-197, 203-206,
209-210, 213-214,
216-218, 222, 224

src/pytest_llm_report/ll m/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
---	-----	---	-----	--------	---

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, 48, 78 77, 79, 81-82, 84, 86-87, 90, 93-94, 96, 98
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

src/pytest_llm_report/options.py	268	57	211	78.73%	<p>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</p>
src/pytest_llm_report/plugin.py	182	24	158	86.81%	<p>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, 13, 15-18, 20-21, 403-404, 407, 23, 29-32, 35, 410-411, 413-414, 319, 377, 481- 417-418, 420, 482, 488, 548- 422-426, 429-430, 549, 571, 595, 432, 434, 437- 611-612</p>

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/pr
ompts.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, 80, 185, 233
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

src/pytest_llm_report/re
nder.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/utils/fs.py	34	1	33	97.06%	11, 13-14, 17, 30, 33, 36, 39, 42, 45, 55-56, 58-60, 63-65, 67, 40 70, 79, 82, 100, 103, 111-113, 116-117, 119-121, 123	
src/pytest_llm_report/utils/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/ut	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/ut	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_complexity_test_high_complexity	1ms	5
--------	--	-----	---

AI ASSESSMENT

Scenario:

tests/test_adaptive_prompts.py::TestComplexityEstimation::test_complexity_test_high_complexity

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

Key Assertions:

- {'name': 'assertion 1', 'description': 'Assertion 1 should be executed before the assertion 2', 'expected_result': True}
- {'name': 'assertion 2', 'description': 'Assertion 2 should be executed after the assertion 3', 'expected_result': False}

Confidence: 80%

Tokens: 118 input + 139 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/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 needed because it checks the behavior of the 'Config' class when given an empty source.**Key Assertions:**

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

Confidence: 80%**Tokens:** 136 input + 158 output = 294 total

COVERAGE

src/pytest_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: To ensure that simple tests have low complexity scores and are not misleading.

Key Assertions:

- {'name': 'complexity_score', 'description': 'The complexity score of the test should be low.'}

Confidence: 80%

Tokens: 115 input + 86 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	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 configuration**Why Needed:** To ensure that the `prompt_tier` field is validated correctly and raises an error when it's not a valid value.**Key Assertions:**

- {'message': "The provided 'prompt_tier' should be one of: 'basic', 'advanced', or 'none'.", 'code': 400}

Confidence: 80%**Tokens:** 126 input + 93 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	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



AI ASSESSMENT

Scenario: Valid prompt tiers

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

Key Assertions:

- {'message': 'Should return an empty list for valid values', 'expected_result': [], 'actual_result': 'errors'}
- {'message': 'Should throw an AssertionError for invalid values', 'expected_result': 'TestError', 'actual_result': 'None'}

Confidence: 80%

Tokens: 142 input + 113 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/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': 'use_standard_prompt_for_complex_tests', 'expected_value': True, 'actual_value': False}

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/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-tiering mechanism is working correctly for simple tests, where minimal prompts are sufficient.

Key Assertions:

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

Confidence: 80%**Tokens:** 155 input + 88 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/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



AI ASSESSMENT

Scenario:

tests/test_adaptive_prompts.py::TestPromptTierSelection::test_minimal_tier_override

Why Needed: To ensure that the minimal prompt is always used for config override tests.

Key Assertions:

- {'name': 'config_override', 'description': 'The config override should be applied to the test provider.'}

Confidence: 80%

Tokens: 122 input + 85 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/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_standar_d_tier_override

1ms



AI ASSESSMENT

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

Why Needed: This test is necessary because it ensures that the config override to standard does not cause any issues with the system prompts.

Key Assertions:

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

Confidence: 80%

Tokens: 148 input + 82 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	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

AI ASSESSMENT

Scenario: Test aggregating all policy for multiple test cases in a single run

Why Needed: Prevents regression when running multiple test cases with the same aggregate policy

Key Assertions:

- The aggregated report contains both tests from each individual report.
- All retained tests are present in the aggregated report.
- No duplicate tests are included in the aggregated report.
- The number of tests in the aggregated report matches the expected value.
- Each test is included only once in the aggregated report.
- Duplicate test names are not included in the aggregated report.
- All retained tests have a unique outcome.
- The aggregate policy 'all' is applied to all test cases.

Confidence: 80%

Tokens: 364 input + 154 output = 518 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: To test that the aggregate function returns None when the aggregation directory does not exist.

Key Assertions:

- {'name': 'assert aggregator.aggregate() is None', 'expected_result': 'None'}

Confidence: 80%

Tokens: 104 input + 80 output = 184 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 latest policy is picked when aggregating reports with different times.**Why Needed:** This test prevents a regression where the latest policy might not be chosen due to inconsistent report times.**Key Assertions:**

- The result from `aggregate()` should contain only one test.
- The outcome of the first test in the result should be 'passed'.
- The second test in the result should have an outcome of 'passed' (latest).
- The aggregated run meta should indicate that both runs were aggregated.
- The summary for the aggregated run should show 1 passed and 0 failed tests.
- The `run_meta` object should contain a `is_aggregated` flag set to True.

Confidence: 80%**Tokens:** 477 input + 162 output = 639 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



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', 'expected_type': 'NoneType', 'message': "Expected agg to be None, but got \"\""}

Confidence: 80%

Tokens: 110 input + 96 output = 206 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)

AI ASSESSMENT

Scenario: Test that aggregate function returns None when no reports exist and no files are found.

Why Needed: Prevents regression where the aggregate function returns an empty list or None when there are no reports or files to aggregate.

Key Assertions:

- aggregator.aggregate() should return None.
- pathlib.Path.exists() should return True for a non-empty directory.
- pathlib.Path.glob() should return [] for a non-existent file or directory.
- The aggregate function should not attempt to aggregate any files or reports.
- No error should be raised when calling the aggregate function with no reports or files.
- The aggregate function should return an empty list or None as expected.

Confidence: 80%

Tokens: 201 input + 154 output = 355 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 2ms 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 proper deserialization of coverage and LLM annotations.

Key Assertions:

- coverage is properly serialized with the correct file paths, line ranges, and line counts.
- LLM annotation is properly serialized with the correct scenario, why needed message, key assertions, confidence level, and token usage information.
- Coverage and LLM annotation can be re-serialized without any issues.

Confidence: 80%

Tokens: 1002 input + 124 output = 1126 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 the aggregation function with source coverage summary.

Why Needed: This test prevents regression in the aggregation function, where it fails to correctly handle source coverage summaries.

Key Assertions:

- The `source_coverage` attribute of each report should contain a single `SourceCoverageEntry` object.
- The `file_path` attribute of the `SourceCoverageEntry` object should match the expected file path.
- All statements in the `coverage_percent` and `covered_ranges` attributes should be integers or floats.
- All statements in the `missed` attribute should be less than or equal to 0.
- The `source_coverage` attribute of each report should contain a list with exactly one element.
- Each `SourceCoverageEntry` object should have all required attributes (file_path, statements, missed, covered, coverage_percent, covered_ranges, missed_ranges).
- All values in the `coverage_percent` and `covered_ranges` attributes should be within valid ranges for source coverage percentages and ranges.

Confidence: 80%

Tokens: 395 input + 221 output = 616 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



AI ASSESSMENT

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

Why Needed: Prevents regression in case the user doesn't configure a source file for LLM coverage.

Key Assertions:

- Verify that calling _load_coverage_from_source() returns None when llm_coverage_source is set to None.
- Verify that calling _load_coverage_from_source() raises a UserWarning with message 'Coverage source not found' when llm_coverage_source is set to '/nonexistent/coverage'.
- Verify that calling _load_coverage_from_source() returns the mock coverage object created by mocking Coverage in pytest_llm_report.coverage_map.CoverageMapper.
- Verify that calling _load_coverage_from_source() calls the mock cov.report() method with a return value of 80.0 when llm_coverage_source is set to '.coverage' and coverage_percentage is 80.0.
- Verify that calling _load_coverage_from_source() returns None when llm_coverage_source is set to a valid file path.
- Verify that the mock cov.report() method was called with the correct arguments (cov, mapper) in all test scenarios.

Confidence: 80%

Tokens: 584 input + 247 output = 831 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)

AI ASSESSMENT

Scenario: Test that the recalculate_summary method updates the latest summary correctly when new test results are added.

Why Needed: This test prevents regression in the aggregation process, where the total count of tests might not be updated correctly if new test results are added after recalculating the summary.

Key Assertions:

- The total count of tests is updated to match the latest summary.
- The passed count remains unchanged.
- The failed count remains unchanged.
- The skipped count remains unchanged.
- The xfailed count remains unchanged.
- The xpassed count remains unchanged.
- The error count remains unchanged.
- The coverage percentage is preserved and updated correctly.
- The total duration of the latest summary is updated to match the new test results.

Confidence: 80%

Tokens: 473 input + 170 output = 643 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



AI ASSESSMENT

Scenario: Test that skipping an invalid JSON report prevents regression.

Why Needed: This test verifies that the aggregation function correctly skips reports with invalid JSON files, preventing potential regressions.

Key Assertions:

- The test verifies that only valid reports are counted in the aggregate result.
- The test verifies that missing fields in an invalid JSON report are not included in the aggregate result.
- The test verifies that a UserWarning is raised when skipping an invalid JSON report, indicating that it's being handled correctly.

Confidence: 80%

Tokens: 352 input + 116 output = 468 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



AI ASSESSMENT

Scenario: The test verifies that the aggregator recalculates the summary correctly when there are multiple tests with different outcomes.

Why Needed: This test prevents regression where a single test's outcome affects the overall coverage calculation.

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 + 113 output = 412 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 functionality of batch optimization message generation.

Key Assertions:

- {'name': 'mock_provider', 'expected_value': 'Mock provider instance'}
- {'name': 'mock_cache', 'expected_value': 'Mock cache instance'}
- {'name': 'mock_assembler', 'expected_value': 'Mock assembler instance'}

Confidence: 80%

Tokens: 112 input + 119 output = 231 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 lost in case of a failure.**Key Assertions:**

- {'name': 'Mocked cache should be initialized with mock provider', 'expected_value': 'mock_provider', 'actual_value': 'mock_cache'}
- {'name': 'Mocked cache should have no pending operations when not in use', 'expected_value': [], 'actual_value': 'mock_cache'}

Confidence: 80%**Tokens:** 101 input + 133 output = 234 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:** To ensure that cached tests are skipped correctly.**Key Assertions:**

- {'name': 'mock_provider', 'value': 'The mock provider is not called during test execution.'}
- {'name': 'mock_cache', 'value': 'The mock cache is not populated with test results.'}
- {'name': 'mock_assembler', 'value': 'The mock assembler does not call the annotated function.'}

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::TestConcurrentAnnotation**Why Needed:** To ensure that annotators can annotate data concurrently without causing performance issues or data corruption.**Key Assertions:**

- {'name': 'Mock provider should not raise an exception when called concurrently', 'expected_value': 'None'}
- {'name': 'Mock cache should be created and populated correctly when called concurrently', 'expected_value': {'key1': 'value1', 'key2': 'value2'}}}

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	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 necessary to ensure that concurrent annotation handles failures correctly.**Key Assertions:**

- {'message': 'Mocked annotator failed to annotate the document.', 'expected_exception': 'annotator.exceptions.AnnotatorException'}
- {'message': 'Mocked annotator encountered an error while annotating the document.', 'expected_exception': 'annotator.exceptions.AnnotatorException'}

Confidence: 80%**Tokens:** 116 input + 126 output = 242 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)

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_progress_reporting**Why Needed:** To ensure that the annotator is reporting progress correctly and accurately.**Key Assertions:**

- {'name': 'Mocked progress bar updates are being reported correctly', 'expected_value': 'True'}
- {'name': 'The progress bar is not being updated when it should be', 'expected_value': 'False'}

Confidence: 80%**Tokens:** 96 input + 109 output = 205 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 annotator correctly displays progress messages during report generation.**Key Assertions:**

- {'name': 'mock_provider', 'expected_value': 'Mock provider instance'}
- {'name': 'mock_cache', 'expected_value': 'Mock cache instance'}
- {'name': 'mock_assembler', 'expected_value': 'Mock assembler instance'}

Confidence: 80%**Tokens:** 101 input + 122 output = 223 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::TestAnnotateTests::test_respects_opt_out_and_limit

Why Needed: The test respects the opt-out and limit settings for annotators.

Key Assertions:

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

Confidence: 80%

Tokens: 104 input + 121 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	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)

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_respects_rate_limit**Why Needed:** To ensure that the annotator respects rate limits and does not exceed them.**Key Assertions:**

- {'name': 'mock_provider.get_rate_limit() returns a valid limit', 'expected_value': 10, 'actual_value': 5}
- {'name': 'mock_provider.get_rate_limit() returns an invalid limit', 'expected_value': 20, 'actual_value': 15}

Confidence: 80%**Tokens:** 112 input + 129 output = 241 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 annotations are applied in the correct order and to avoid any potential issues with concurrent access to the annotator.**Key Assertions:**

- {'name': 'Annotations are applied sequentially', 'description': 'The annotator should apply each annotation in sequence, without skipping any steps.'}
- {'name': 'No annotations are skipped due to cache or provider issues', 'description': 'Even if the cache or provider is experiencing issues, the annotator should still apply all annotations in sequence.'}

Confidence: 80%**Tokens:** 98 input + 130 output = 228 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: Sequential annotation error tracking**Why Needed:** This test is necessary to ensure that the annotator correctly handles errors during sequential annotation.**Key Assertions:**

- {'expected': 'Error message should be logged and returned in the response', 'actual': 'Mocked logging and caching are not being used correctly.'}
- {'expected': 'Error message should be logged and returned in the response with a specific key', 'actual': 'Mocked logging is not being used to log error messages.'}

Confidence: 80%**Tokens:** 105 input + 122 output = 227 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



AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_skips_if_disabled**Why Needed:** To ensure that the annotator does not skip tests when LLM is disabled.**Key Assertions:**

- {'name': 'does_not_call_config', 'description': "The config function should be called with a Config object that has 'llm' set to False'}
- {'name': 'does_not_skip_test', 'description': 'The test should not skip when LLM is disabled'}

Confidence: 80%**Tokens:** 108 input + 127 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/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: Because the annotator should not be skipped when a provider is unavailable.

Key Assertions:

- {'name': 'mock_provider', 'expected': 'MockProvider instance was created'}

Confidence: 80%

Tokens: 101 input + 81 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/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 test that the `extract_json_from_response` function correctly handles malformed JSON responses.**Key Assertions:**

- {'name': 'Expected error message', 'description': "The error message returned by `extract_json_from_response` should be 'Failed to parse LLM response as JSON'."}
- {'name': 'Expected invalid JSON string', 'description': 'The input JSON string should contain invalid characters or syntax.'}

Confidence: 80%**Tokens:** 152 input + 122 output = 274 total

COVERAGE

src/pytest_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: Tests that the `base_parse_response` function handles non-string fields correctly when provided with a JSON object containing integers, lists, and other types.

Why Needed: This test prevents regression in case the `base_parse_response` function is modified to handle non-string fields without proper error handling or validation.

Key Assertions:

- The value of the 'scenario' key should be an integer.
- The value of the 'why_needed' key should contain a list.
- The value of the 'key_assertions' key should contain the string 'a'.

Confidence: 80%**Tokens:** 269 input + 130 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/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 returns a valid instance of `GeminiProvider`.

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



AI ASSESSMENT

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

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

Key Assertions:

- {'name': 'Expected exception message', 'value': '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 that the `get_litellm_provider` function returns a valid instance of `LiteLLMProvider`.

Key Assertions:

- {'name': 'provider' is an instance of 'LiteLLMProvider', 'expected_type': 'LiteLLMProvider'}

Confidence: 80%

Tokens: 109 input + 97 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/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)

AI ASSESSMENT

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

Why Needed: To test the functionality of getting a provider without any configuration.

Key Assertions:

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

Confidence: 80%

Tokens: 104 input + 74 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/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 ensure that the `get_ollama_provider` function returns an instance of `OllamaProvider` as expected.

Key Assertions:

- {'name': 'provider' is an instance of OllamaProvider', 'expected_result': 'OllamaProvider'}

Confidence: 80%

Tokens: 108 input + 96 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	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



AI ASSESSMENT

Scenario: Verify that the test function checks for available caches correctly when a provider implements _check_availability.

Why Needed: This test prevents regression in cases where a provider does not implement _check_availability, causing tests to fail due to missing cache availability.

Key Assertions:

- provider.is_available() is True
- provider.is_available() is True
- checks == 1

Confidence: 80%

Tokens: 280 input + 94 output = 374 total

COVERAGE

src/pytest_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 `get_model_name` method returns the default model name from the configuration when no custom model is provided.

Key Assertions:

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

Confidence: 80%

Tokens: 114 input + 90 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	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



AI ASSESSMENT

Scenario: tests/test_base_maximal.py

Why Needed: This test ensures that the rate limits are set to None by default when creating a ConcreteProvider instance.

Key Assertions:

- {'name': 'provider.get_rate_limits() is None', 'expected_value': 'None'}

Confidence: 80%

Tokens: 108 input + 75 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	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::TestLlmProviderDefaults::test_is_local_defaults_to_false

Why Needed: To test that the default value for `is_local` is indeed `False`.

Key Assertions:

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

Confidence: 80%

Tokens: 105 input + 93 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/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: Test 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 causing unexpected behavior or errors.

Key Assertions:

- Context files should be added to the prompt.
- The prompt should include the source file `src/module.py`.
- The prompt should include the function `def helper()` from the source file `src/module.py`.

Confidence: 80%

Tokens: 261 input + 105 output = 366 total

COVERAGE

src/pytest_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: Test the parametrized batch prompt functionality.**Why Needed:** This test prevents regression when using parameterized batches, ensuring that all variants are included in the prompt.**Key Assertions:**

- The prompt should include 'Test Group: test.py::test_add[*]' and 'Parameterizations (2 variants)'
- The prompt should contain '[1+1=2]' and '[0+0=0]'
- The prompt should mention 'ONE annotation'

Confidence: 80%**Tokens:** 330 input + 110 output = 440 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Verifies that a single test generates the expected normal prompt.

Why Needed: Prevents regression when testing batched requests with multiple tests.

Key Assertions:

- The generated prompt should contain 'Test: test.py::test_foo'.
- '```python' is present in the prompt.
- The source code of the test should be included in the prompt.
- The presence of 'Parameterizations' in the prompt should be avoided.

Confidence: 80%

Tokens: 269 input + 107 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/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)

AI ASSESSMENT

Scenario: Test that the same source code produces the same hash for consistent hashing.

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

Key Assertions:

- The source code should be the same across all executions.
- The length of the resulting hash should remain constant at 32 bytes.
- If the source code changes, the hash should not change.
- `_compute_source_hash(source)` should return the same hash for different source codes.
- The hash should be a valid SHA-256 hash.

Confidence: 80%

Tokens: 220 input + 131 output = 351 total

COVERAGE

src/pytest_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 sources produce different hashes, which is a requirement for batch processing.

Key Assertions:

- {'assertion_type': 'different', 'condition': {'source': 'def test_a(): pass'}, 'expected_result': {'hash': '1234567890abcdef'}}}
- {'assertion_type': 'different', 'condition': {'source': 'def test_b(): pass'}, 'expected_result': {'hash': 'fedcba987654321'}}}

Confidence: 80%

Tokens: 127 input + 143 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/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



AI ASSESSMENT

Scenario: tests/test_batching.py::TestComputeSourceHash::test_empty_source**Why Needed:** The current implementation of compute_source_hash() 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 + 94 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/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 `batch_max_tests` configuration value is required to ensure the correct behavior of batched tests.

Key Assertions:

- {'name': 'config.validate() returns errors', 'description': 'The `validate()` method should return an error if `batch_max_tests` is not set or is less than 1.', 'expected_type': 'list[str]', 'expected_value': ['batch_max_tests']}
- {'name': "any('batch_max_tests' in e for e in errors)", 'description': "The method should return True if 'batch_max_tests' is present in any of the error messages.", 'expected_type': 'bool'}

Confidence: 80%

Tokens: 126 input + 178 output = 304 total

COVERAGE

src/pytest_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 a non-negative integer', 'value': -1}

Confidence: 80%

Tokens: 126 input + 74 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/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



AI ASSESSMENT

Scenario: TestConfigValidation

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

Key Assertions:

- {'name': 'config.validate() returns an error message', 'description': 'The validate method should return a list of errors.', 'expected': ["context_compression must be one of 'none', 'gzip', or 'lz4'"], 'actual': ["context_compression must be one of 'invalid', 'none', 'gzip', 'lz4'"]}

Confidence: 80%

Tokens: 122 input + 121 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/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



AI ASSESSMENT

Scenario: TestConfigValidation

Why Needed: Valid compression modes should pass.

Key Assertions:

- {'message': 'Context compression mode is required for valid context compression.', 'expected_value': 'none'}

Confidence: 80%

Tokens: 133 input + 60 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	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:** To ensure that complex parameters are fully stripped from the base node ID.**Key Assertions:**

- {'name': 'expected result', 'value': 'test.py::test[a-b-c]'}

Confidence: 80%**Tokens:** 109 input + 78 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: 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:** To ensure that the `parametrized_nodeid` function correctly strips parameters from nodeids.**Key Assertions:**

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

Confidence: 80%**Tokens:** 133 input + 96 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/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



AI ASSESSMENT

Scenario: tests/test_batching.py::TestGetBaseNodeid::test_simple_nodeid**Why Needed:** This test checks that a simple nodeid without params returns unchanged.**Key Assertions:**

- {'expected_value': 'tests/test_foo.py::test_bar', 'actual_value': 'tests/test_foo.py::test_bar'}

Confidence: 80%**Tokens:** 123 input + 87 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/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



AI ASSESSMENT

Scenario: Large groups should be split by batch_max_tests to avoid memory issues.

Why Needed: This test prevents a potential memory leak in large group sizes where the tests are not properly split into batches.

Key Assertions:

- len(batches) == 3
- len(batches[0].tests) == 2
- len(batches[1].tests) == 2
- len(batches[2].tests) == 1

Confidence: 80%

Tokens: 364 input + 113 output = 477 total

COVERAGE

src/pytest_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 batching disabled

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

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 that parametrized tests are grouped together by batch.**Why Needed:** This test prevents a regression where parametrized tests are not properly grouped together in batches.**Key Assertions:**

- The number of batches should be equal to 1.
- Each batch should contain exactly 3 tests.
- Each batch should have an `is_parametrized` attribute set to True.
- Each batch's base nodeid should match the path '`test.py::test_add`'.

Confidence: 80%**Tokens:** 346 input + 113 output = 459 total

COVERAGE

src/pytest_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: Test 'test_single_tests_no_grouping' verifies that single tests are handled correctly without grouping.

Why Needed: This test prevents regression in case of batch testing with no groupings, ensuring each test is executed individually.

Key Assertions:

- Each test should be a separate batch.
- Batching should not affect the number or structure of individual tests.
- The expected number of batches (2) and individual tests (1+1=2) should match the actual output.

Confidence: 80%

Tokens: 278 input + 114 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/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 storing and retrieving data from the same source, which produces the same hash.

Key Assertions:

- {'name': 'same_source', 'expected_value': 'True'}

Confidence: 80%

Tokens: 107 input + 78 output = 185 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/test_cache.py::TestHashSource::test_different_source_different_hash

Why Needed: To ensure that the hash function is working correctly and producing different hashes for different source code.

Key Assertions:

- {'description': 'Different source should produce different hash.', 'expected_result': 'different', 'actual_result': 'same'}

Confidence: 80%

Tokens: 108 input + 82 output = 190 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



AI ASSESSMENT

Scenario: tests/test_cache.py::TestHashSource::test_hash_length**Why Needed:** The hash length is not sufficient to uniquely identify the cache key.**Key Assertions:**

- {'name': 'hash', 'expected': "a string of 16 hexadecimal digits (e.g., '1234567890abcdef')", 'actual': ''}

Confidence: 80%**Tokens:** 100 input + 92 output = 192 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



AI ASSESSMENT

Scenario: Verify that the cache is cleared and all entries are removed.

Why Needed: The test prevents a bug where the cache might not be properly cleared when adding multiple entries, potentially leading to incorrect results or data loss.

Key Assertions:

- Clearing the cache should remove all existing entries.
- Adding multiple entries should result in only two remaining entries.
- Cache should return None for both 'test::a' and 'test::b' after clearing.
- The cache should be empty after clearing, with no matching annotations.

Confidence: 80%

Tokens: 283 input + 124 output = 407 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



AI ASSESSMENT

Scenario: tests/test_cache.py**Why Needed:** To ensure that annotations with errors are not cached.**Key Assertions:**

- {'name': 'annotation is set and retrieved correctly', 'expected_result': 'abc123'}
- {'name': 'result is None when annotation is retrieved from cache', 'expected_result': 'None'}

Confidence: 80%**Tokens:** 157 input + 90 output = 247 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 cache returns None for missing entries.**Key Assertions:**

- {'name': 'cache entry should be None', 'expected_result': 'None'}

Confidence: 80%**Tokens:** 128 input + 71 output = 199 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: Test that annotations are stored and retrieved correctly from the cache.

Why Needed: Prevents bypass attacks by ensuring that the cache stores and retrieves annotations in a consistent manner.

Key Assertions:

- Check if the annotation is set correctly for the given key.
- Verify that the annotation's scenario matches the expected value.
- Ensure that the annotation's confidence level matches the expected value.
- Confirm that the retrieved result has the same scenario and confidence as the original annotation.
- Verify that the retrieved result does not contain any null values.
- Check if the cache stores annotations in a consistent manner across different runs.

Confidence: 80%

Tokens: 286 input + 142 output = 428 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



AI ASSESSMENT

Scenario:

tests/test_collector.py::TestCollectorCollectionErrors::test_collection_error_structure

Why Needed: The current test does not verify the correct structure of collection errors.

Key Assertions:

- {'assertion_type': 'assert', 'expected_value': 'nodeid', 'actual_value': 'test_bad.py'}
- {'assertion_type': 'assert', 'expected_value': 'message', 'actual_value': 'SyntaxError'}

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)

PASSED

tests/test_collector.py::TestCollectorCollectionErrors::test_get_collection_errors_initially_empty

1ms



AI ASSESSMENT

Scenario:

tests/test_collector.py::TestCollectorCollectionErrors::test_get_collection_errors_initially_empty

Why Needed: The test is checking if the `get_collection_errors` method returns an empty list when the collection is initially empty.

Key Assertions:

- {'name': 'assert get_collection_errors is a list', 'expected_value': [], 'actual_value': 'collector.get_collection_errors() == []'}

Confidence: 80%

Tokens: 114 input + 102 output = 216 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



AI ASSESSMENT

Scenario: tests/test_collector.py::TestCollectorMarkerExtraction

Why Needed: Default llm_context_override should be None.

Key Assertions:

- {'assertion': 'result.llm_context_override is None', 'expected_value': 'None'}

Confidence: 80%

Tokens: 136 input + 70 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)

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: Default llm_opt_out should be False.

Key Assertions:

- {'name': 'assert result.llm_opt_out is False', 'expected_value': False, 'message': 'Expected llm_opt_out to be False'}

Confidence: 80%

Tokens: 136 input + 91 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)

PASSED

tests/test_collector.py::TestCollectorOutputCapture::test_capture_enabled_by_default

1ms



AI ASSESSMENT

Scenario:

tests/test_collector.py::TestCollectorOutputCapture::test_capture_enabled_by_default

Why Needed: The test captures output by default, which can lead to unexpected behavior if not expected.

Key Assertions:

- {'name': 'config.capture_failed_output', 'expected_value': True}

Confidence: 80%

Tokens: 104 input + 78 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/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 of `capture_output_max_chars` is not sufficient to handle large output files. This test ensures that the default value is set correctly.

Key Assertions:

- {'name': 'assert capture_output_max_chars is equal to 4000', 'expected_value': 4000, 'actual_value': 10000}

Confidence: 80%

Tokens: 108 input + 109 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/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



AI ASSESSMENT

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

Why Needed: To ensure that xfail failures are recorded as xfailed in the test results.

Key Assertions:

- {'name': 'result.outcome', 'expected_value': 'xfailed'}

Confidence: 80%

Tokens: 206 input + 81 output = 287 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_name': 'result.outcome', 'expected_value': 'xpassed'}

Confidence: 80%

Tokens: 205 input + 77 output = 282 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



AI ASSESSMENT

Scenario: The `TestCollector` class initializes correctly with an empty collection.

Why Needed: This test prevents a potential bug where the `TestCollector` instance is not initialized properly, leading to incorrect results or behavior.

Key Assertions:

- collector.results == {}
- collector.collection_errors == []
- collector.collected_count == 0

Confidence: 80%

Tokens: 205 input + 86 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/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

AI ASSESSMENT

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

Why Needed: To ensure that the collector correctly sorts the results by nodeid.

Key Assertions:

- {'message': 'Results should be sorted by nodeid.', 'expected_result': ['a_test.py::test_a', 'z_test.py::test_z']}

Confidence: 80%

Tokens: 227 input + 89 output = 316 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: Verify that the `handle_collection_finish` method correctly tracks collected and deselected counts.

Why Needed: This test prevents a potential bug where the count of collected items is not updated correctly after the collection finish.

Key Assertions:

- The `collected_count` attribute should be set to 3 (the number of collected items).
- The `deselected_count` attribute should be set to 1 (the number of deselected items).

Confidence: 80%

Tokens: 256 input + 109 output = 365 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_report

Why Needed: Capture output via handle_runtest_logreport is disabled for integration test.

Key Assertions:

- {'name': 'expected_result', 'type': 'assertion', 'message': "Expected `collector.results['t'].captured_stdout` to be None"}

Confidence: 80%

Tokens: 211 input + 98 output = 309 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_stderr

1ms



AI ASSESSMENT

Scenario: TestCollectorInternals::test_capture_stderr

Why Needed: To test that the collector correctly captures stderr.

Key Assertions:

- {'name': 'Expected captured stderr to be empty', 'description': 'The collector should not capture any output from stderr.', 'expected_value': '', 'actual_value': 'Some error'}

Confidence: 80%

Tokens: 157 input + 88 output = 245 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



AI ASSESSMENT

Scenario: TestCollectorInternals::test_capture_output_stdout

Why Needed: To verify that the collector correctly captures stdout and stores it in the TestCaseResult.

Key Assertions:

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

Confidence: 80%

Tokens: 157 input + 71 output = 228 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



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 `create_result_with_item_markers` verifies that the collector extracts item markers correctly.

Why Needed: This test prevents a potential bug where the collector does not extract item markers from the item, potentially leading to incorrect report generation.

Key Assertions:

- item.get_closest_marker('llm_opt_out') returns MagicMock()
- item.get_closest_marker('llm_context') returns MagicMock(args=['complete'])
- item.get_closest_marker('requirement') returns MagicMock(args=['REQ-1', 'REQ-2'])
- result.param_id is set to 'param1'
- result.llm_opt_out is True
- result.llm_context_override is set to 'complete'
- result.requirements contains ['REQ-1', 'REQ-2']

Confidence: 80%

Tokens: 382 input + 178 output = 560 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



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: test_collector_maximal

Why Needed: To test the `extract_error` method's ability to return a string representing an error message.

Key Assertions:

- {'name': 'expected output', 'value': 'Some error occurred'}

Confidence: 80%

Tokens: 130 input + 69 output = 199 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 that the `'_extract_skip_reason` method returns `None` when no longrepr is provided.

Key Assertions:

- {'name': 'assert collector._extract_skip_reason(report) is None', 'expected_result': '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': "assert _extract_skip_reason returns 'Just skipped'", 'expected_value': 'Just skipped'}

Confidence: 80%

Tokens: 133 input + 85 output = 218 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 verifies that the `handle_collection_report` method records a collection error and updates the `collection_errors` list with the relevant information.

Why Needed: This test prevents a potential bug where the collector does not record errors in a collection report, potentially leading to missing important information about the failure.

Key Assertions:

- The `handle_collection_report` method should update the `collection_errors` list with the nodeid and message of the error.
- The `collection_errors` list should contain exactly one item with the specified nodeid and message.
- The nodeid in the first collection_error should match the value passed to the `report.nodeid` attribute.
- The message in the first collection_error should match the value passed to the `report.message` attribute.
- The error type (in this case, a 'SyntaxError') should be present in the first collection_error.

Confidence: 80%

Tokens: 273 input + 196 output = 469 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_runttest_rerun 2ms 3

AI ASSESSMENT

Scenario: Test 'handle_runttest_rerun' verifies that the TestCollector handles rerun attribute correctly.

Why Needed: This test prevents a potential regression where the TestCollector does not handle reruns correctly, potentially leading to incorrect results or failures.

Key Assertions:

- res.rerun_count should be equal to 1
- res.final_outcome should be 'failed'
- collector.results['t:r'] should contain a 'rerun' key with value 1
- collector.results['t:r'].final_outcome should be 'failed'

Confidence: 80%

Tokens: 281 input + 132 output = 413 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_runttest_setup_failure

1ms



AI ASSESSMENT

Scenario: Test 'handle_runttest_setup_failure' verifies that a setup error is recorded in the report.

Why Needed: This test prevents regression by ensuring that setup errors are properly reported and handled.

Key Assertions:

- res.outcome should be set to 'error'
- res.phase should be set to 'setup'
- res.error_message should match 'Setup failed'

Confidence: 80%

Tokens: 300 input + 91 output = 391 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_runttest_teardown_failure

1ms

 3

AI ASSESSMENT

Scenario: Test that handle_runttest_teardown_failure verifies that a teardown failure records an error and prevents the collector from proceeding with the next test.

Why Needed: This test prevents regression by ensuring that when a teardown operation fails, it correctly reports an error and stops the collection of results for the current test.

Key Assertions:

- The `teardown` report should contain an error message indicating 'Cleanup failed'.
- The `outcome` of the result for the current test should be set to 'error'.
- The `phase` of the result for the current test should be set to 'teardown'.
- The `error_message` of the result for the current test should contain 'Cleanup failed'.
- If the teardown operation fails, the collector should not proceed with collecting results for the next test.
- If the teardown operation succeeds, the collector should still report an error and stop collecting results for the current test.

Confidence: 80%

Tokens: 391 input + 207 output = 598 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



AI ASSESSMENT

Scenario: Test invalid compression mode

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

Key Assertions:

- {'description': "Context compression should be 'none' or 'gzip'", 'expected_value': 'none|gzip', 'actual_value': 'invalid'}

Confidence: 80%

Tokens: 124 input + 76 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/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 is not allowed in context lines.

Key Assertions:

- {'message': 'context_line_padding should be a non-negative integer.', 'expected_type': 'int', 'actual_type': -1}

Confidence: 80%

Tokens: 121 input + 83 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/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



AI ASSESSMENT

Scenario: TestConfigValidation

Why Needed: To ensure that valid compression modes are validated correctly.

Key Assertions:

- {'name': 'config.validate()' should return an empty list of errors', 'description': 'The validate method should not return any error messages for valid compression modes.'}

Confidence: 80%

Tokens: 135 input + 76 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/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



AI ASSESSMENT

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

Why Needed: Zero padding is a valid configuration for the context line padding.

Key Assertions:

- {'expected': 'context_line_padding should be 0 or less.', 'actual': 0}

Confidence: 80%

Tokens: 122 input + 77 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::TestContextCompression::test_compression_enabled_by_default

1ms

3

AI ASSESSMENT

Scenario:

tests/test_context_compression.py::TestContextCompression::test_compression_enabled_by_default

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: TestContextCompression::test_compression_mode_lines

Why Needed: Lines compression mode is required for this test.

Key Assertions:

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

Confidence: 80%

Tokens: 113 input + 63 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/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 set correctly in the default context.

Key Assertions:

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

Confidence: 80%

Tokens: 106 input + 82 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	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: Test that contiguous covered lines do not have gap indicators when there is no padding.

Why Needed: Prevents regression where contiguous lines without gaps are incorrectly marked as uncovered.

Key Assertions:

- The count of '# ...' should be zero for contiguous lines without gaps.
- # L3:
- # L4:
- # L5:

Confidence: 80%

Tokens: 293 input + 86 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/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: This test is needed because it checks for an empty coverage scenario, which can occur when no lines are covered.

Key Assertions:

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

Confidence: 80%

Tokens: 130 input + 83 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)
src/pytest_llm_report/prompts.py	3 lines (ranges: 33, 216-217)

PASSED

tests/test_context_compression.py::TestExtractCoveredLines::test_extract_multiple_ranges

1ms



AI ASSESSMENT

Scenario: Test Extract Covered Lines: Multiple covered ranges should be extracted with gap indicators.

Why Needed: This test prevents a regression where multiple covered lines are not correctly identified with gap indicators.

Key Assertions:

- The '# ...' line is present in the result.
- The '# L3:' line is present in the result.
- The '# L15:' line is present in the result.
- Gap indicator between ranges is included in the output.
- Multiple covered lines are correctly identified with gap indicators.

Confidence: 80%

Tokens: 274 input + 120 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/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



AI ASSESSMENT

Scenario: Single covered line should be extracted with padding.

Why Needed: This test prevents a bug where the single covered line is not extracted correctly due to incorrect padding.

Key Assertions:

- assert '# L2:' in result
- assert '# L3:' in result
- assert '# L4:' in result

Confidence: 80%

Tokens: 302 input + 81 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)
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_pading_boundary

1ms



4

AI ASSESSMENT

Scenario: Test Extract Covered Lines: Padding should not go beyond file boundaries.**Why Needed:** This test prevents a potential issue where padding exceeds the file boundary, potentially causing incorrect results or errors.**Key Assertions:**

- The extracted covered lines should not have negative line numbers (L0 and L4).
- The extracted covered lines should only contain lines from the first to third lines (L1, L2, and L3).

Confidence: 80%**Tokens:** 288 input + 104 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	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



AI ASSESSMENT

Scenario: tests/test_context_limits.py::test_no_truncation_needed**Why Needed:** This test is needed because the current implementation may truncate context due to performance reasons.**Key Assertions:**

- {'assertion': "The prompt should contain 'short content'", 'expected_result': 'short content'}
- {'assertion': "The prompt should not contain 'truncated'", 'expected_result': 'no-truncation-needed'}

Confidence: 80%**Tokens:** 158 input + 110 output = 268 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_context_limits.py::test_smart_distribution verifies the context limits for F1 and F2 to ensure they are allocated fair share of budget without unnecessary truncation.

Why Needed: This test prevents regression that could cause F1's content to be truncated unnecessarily when its required tokens exceed available budget.

Key Assertions:

- F1 should get full allocation of budget (40 tokens).
- F2 should get extra budget beyond required allocation (180 tokens).
- F2 content is not fully allocated, with excess characters (>480 tokens) that are truncated.

Confidence: 80%

Tokens: 773 input + 131 output = 904 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Verify that the splitting logic correctly identifies files with large contents and truncates strings while maintaining budget limits.

Why Needed: This test prevents a potential regression where the splitting logic fails to truncate strings for files with large contents, leading to incorrect results or unexpected behavior.

Key Assertions:

- The file "f1" contains more than 100 tokens (400 characters).
- The file "f2" contains more than 100 tokens (400 characters).
- The string 'Present' is truncated and appears in the prompt.
- The budget per file is approximately 80 tokens, which is within the allowed limit of 200 tokens for both files.
- The overhead is small, as expected.

Confidence: 80%

Tokens: 317 input + 160 output = 477 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test truncation logic for large context files when limit is exceeded.

Why Needed: This test prevents regression where the context is too long and needs to be truncated due to memory constraints.

Key Assertions:

- The prompt should be truncated to fit within 100 tokens minus system prompt overhead.
- Context should be very small or empty if limit is exceeded.
- Prompt should contain '[... truncated]' or 'Relevant context' if no budget is left.

Confidence: 80%

Tokens: 397 input + 109 output = 506 total

COVERAGE

src/pytest_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



AI ASSESSMENT

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

Why Needed: To test the functionality of collapsing empty lines in a context.

Key Assertions:

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

Confidence: 80%

Tokens: 128 input + 83 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::TestCollapseEmptyLines::test_many_empty_lines

1ms



AI ASSESSMENT

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

Why Needed: To test the functionality of collapsing empty lines in a multi-line source.

Key Assertions:

- {'assertion': "The result is equal to 'line1\n\nline2'.", 'expected_result': 'line1\n\nline2'}

Confidence: 80%

Tokens: 127 input + 90 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	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



AI ASSESSMENT

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

Why Needed: To test if the `collapse_empty_lines` function correctly preserves up to 2 consecutive newlines.

Key Assertions:

- {'description': 'The source string has two or fewer consecutive newlines.', 'expected_result': 'line1\n\nline2'}

Confidence: 80%

Tokens: 125 input + 94 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/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



AI ASSESSMENT

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

Why Needed: Preserve single newlines in collapsed lines

Key Assertions:

- {'expected_result': 'line1\nline2\nline3', 'actual_result': 'line1\nline2\nline3'}

Confidence: 80%

Tokens: 121 input + 82 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/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



AI ASSESSMENT

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

Why Needed: Because empty lines are not being collapsed by the current implementation.

Key Assertions:

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

Confidence: 80%

Tokens: 137 input + 70 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/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 and optimizes the code efficiently.**Key Assertions:**

- {'assertion': 'The optimized source code contains all necessary key assertions.', 'expected_result': 'The optimized source code contains all necessary key assertions.'}
- {'assertion': 'The combined optimization process does not introduce any new errors or warnings.', 'expected_result': 'No new errors or warnings were introduced by the combined optimization process.'}

Confidence: 80%**Tokens:** 96 input + 134 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/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: To ensure that default context stripping only removes docstrings, without affecting comments.

Key Assertions:

- {'name': 'docstring removal', 'expected_result': 'True'}
- {'name': 'comment presence', 'expected_result': 'False'}

Confidence: 80%

Tokens: 100 input + 97 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/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



AI ASSESSMENT

Scenario: tests/test_context_util.py::TestOptimizeContext::test_empty_source**Why Needed:** The function should be able to handle an empty source without raising an error.**Key Assertions:**

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

Confidence: 80%**Tokens:** 95 input + 69 output = 164 total

COVERAGE

src/pytest_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: To handle cases where the source code contains only whitespace characters, such as blank lines or multiple consecutive whitespace characters.

Key Assertions:

- {'name': 'expected result', 'value': '\n\n \n'}

Confidence: 80%

Tokens: 115 input + 87 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/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



AI ASSESSMENT

Scenario: tests/test_context_util.py::TestOptimizeContext::test_strip_both**Why Needed:** To remove unnecessary documentation from the code.**Key Assertions:**

- {'assertion_type': 'strip_comments', 'expected_output': 'docstring', 'actual_output': 'comment'}

Confidence: 80%**Tokens:** 95 input + 77 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/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 comments.

Key Assertions:

- {'assertion_type': 'expected_output', 'expected_output': 'def foo():\n # This is a comment\n pass'}

Confidence: 80%

Tokens: 95 input + 83 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	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)

AI ASSESSMENT

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

Why Needed: The current implementation of `optimize_context` does not strip unnecessary code, which can lead to unexpected behavior if the user explicitly requests it.

Key Assertions:

- {'assertion': 'The function `foo()` is kept in the optimized context.', 'expected_result': 'def foo():', 'actual_result': 'def foo():'}

Confidence: 80%

Tokens: 94 input + 108 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/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



AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripComments::test_comment_after_string_with_hash

Why Needed: This test ensures that the `strip_comments` function correctly removes comments from strings containing hashes.

Key Assertions:

- {'assertion': 'The output of the `strip_comments` function is equal to the expected string.', 'expected_value': '\'url = "http://example.com#anchor"\'', 'actual_value': 'result'}

Confidence: 80%

Tokens: 134 input + 109 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/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)

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:

- {'name': 'basic behavior', 'expected_result': '# comment', 'actual_result': 'The escaped quote handling is simplified, check basic behavior'}

Confidence: 80%

Tokens: 133 input + 88 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/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)

AI ASSESSMENT

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

Why Needed: To strip quotes from a string containing both single and double quotes.

Key Assertions:

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

Confidence: 80%

Tokens: 101 input + 83 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	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



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 exposed.**Key Assertions:**

- {'assertion_type': 'source_code', 'expected_result': 'def foo():\n# This line will be\nstripped\nreturn 1'}

Confidence: 80%**Tokens:** 91 input + 90 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	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.

Key Assertions:

- {'assertion': 'result = \'url = "http://example.com#anchor"\'.split(\')'}
- 'expected_result': '

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 the `strip_comments` function preserves # inside single-quoted strings, which is essential for maintaining the integrity of the original source code.

Key Assertions:

- {'assertion': 'result == "url = \'http://example.com#anchor\'"', 'expected_result': '"url = \'http://example.com#anchor\'"'}

Confidence: 80%

Tokens: 135 input + 116 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/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



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_result': 'x = 1', 'actual_result': 'x = 1'}

Confidence: 80%

Tokens: 119 input + 76 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/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



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:

- {'assertion_type': 'strip_comments', 'expected_result': 'The line is no longer a comment'}

Confidence: 80%

Tokens: 99 input + 77 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/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 checking if the function `strip_docstrings` handles syntax errors correctly by returning the original source code when an unclosed parenthesis is found.

Key Assertions:

- {'expected_value': 'def foo(unclosed paren', 'actual_value': 'def foo(parentheses ', 'message': 'The expected value does not match the actual value.'}

Confidence: 80%

Tokens: 119 input + 115 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/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:** The function `strip_docstrings` is used to strip unnecessary docstrings from Python code.**Key Assertions:**

- {'assertion': 'The function should remove all docstrings, not just the first one.', 'expected_result': 'All docstrings in the module should be removed.'}
- {'assertion': 'The function should only strip the first occurrence of a docstring.', 'expected_result': 'Only the first docstring should be stripped.'}

Confidence: 80%**Tokens:** 95 input + 135 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/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



AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripDocstrings::test_preserves_multiline_data_strings

Why Needed: Preserve multiline data strings in docstrings when stripping context.

Key Assertions:

- {'name': 'strip_context preserves triple-quoted strings', 'expected_value': 'foo()', 'actual_value': 'def foo():\n """\n def foo():\n ``python\n}'}

Confidence: 80%

Tokens: 103 input + 106 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/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:

- {'name': 'Regular string is preserved', 'expected_value': '\'x = "hello world"\'', 'actual_value': '\'x = "hello world"\''}

Confidence: 80%

Tokens: 102 input + 88 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/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



AI ASSESSMENT

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

Why Needed: Preserve strings in structures is a critical test case for context utilities.

Key Assertions:

- {'description': 'Ensure the function correctly preserves strings in lists/dicts.', 'expected_value': 'Should preserve strings in lists/dicts.'}
- {'description': 'Verify the function handles nested quotes correctly.', 'expected_value': 'Should handle nested quotes correctly'}

Confidence: 80%

Tokens: 146 input + 119 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/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: Because of the presence of a multiline docstring, it is causing issues with test coverage and readability.

Key Assertions:

- {'assertion_type': 'remove', 'expected_result': 'striped'}

Confidence: 80%

Tokens: 97 input + 83 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/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 remove triple double-quoted docstrings from the test source code.

Key Assertions:

- {'assertion': "source.strip('')"}

Confidence: 80%

Tokens: 106 input + 99 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/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



AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripDocstrings::test_strip_triple_single_quoted_docstring

Why Needed: To remove triple single-quoted docstrings from the test source.

Key Assertions:

- {'assertion': 'source should be modified to not include triple single-quoted docstrings', 'expected_result': "source = \"def foo():\\n # triple single-quoted docstring\\n pass\""}'}

Confidence: 80%

Tokens: 106 input + 108 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/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: Test the Gemini model parsing edge cases for coverage boosters.

Why Needed: This test prevents regression in case a new model is added to the preferred list without updating the existing tests.

Key Assertions:

- The function provider._parse_preferred_models() should return ['m1', 'm2'] when the config.model is set to 'm1' or 'm2'.
- The function provider._parse_preferred_models() should return an empty list when the config.model is None.
- The function provider._parse_preferred_models() should return ['All'] when the config.model is set to 'All'.

Confidence: 80%

Tokens: 273 input + 144 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/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



AI ASSESSMENT

Scenario: Verify that the rate limiter prevents over and under token limits for edge cases.**Why Needed:** This test prevents a potential regression in the rate limiter's behavior when dealing with edge cases where there are more tokens available than requested.**Key Assertions:**

- The next_available_in method should return 0 when both limits are exceeded ($50+60 > 100$).
- The next_available_in method should not be able to return a positive value when only one limit is exceeded ($10 > 0$).

Confidence: 80%**Tokens:** 273 input + 120 output = 393 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Verify that the `to_dict()` method of `SourceCoverageEntry` returns the correct coverage percentage.

Why Needed: This test prevents a potential bug where the coverage percentage is not correctly calculated for certain types of source code.

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 + 128 output = 446 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_coverage_map.py::TestCoverageMapper::test_create_mapper**Why Needed:** Mapper initialization should be tested.**Key Assertions:**

- {'name': 'mapper.config is equal to config', 'expected_value': 'config', 'actual_value': 'mapper.config'}
- {'name': 'mapper.warnings are empty', 'expected_value': [], 'actual_value': []}

Confidence: 80%**Tokens:** 109 input + 103 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/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



AI ASSESSMENT

Scenario: tests/test_coverage_map.py::TestCoverageMapper::test_get_warnings**Why Needed:** To ensure the get_warnings method returns a valid list of warnings.**Key Assertions:**

- {'description': 'The function should return a list of warnings.', 'expected_type': 'list'}

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/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 verifies that the `map_coverage` method returns an empty dictionary when no coverage file is present.

Why Needed: Prevents a potential bug where the test fails with an incorrect result (empty dict) when there's no coverage file.

Key Assertions:

- The `map_coverage()` method should return an empty dictionary when no coverage file exists.
- The `map_coverage()` method should have at least one warning in this scenario.
- The `map_coverage()` method should not throw any exceptions when no coverage file is present.

Confidence: 80%

Tokens: 277 input + 124 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/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



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 regression where the coverage map might not include all phases if the `include_phase` parameter is not set to 'all'.

Key Assertions:

- The node ID extracted for each phase should be the same as the original node ID.
- The node IDs for different phases should be in the same order.
- Any additional phase information (e.g., method name) should be ignored when extracting node IDs.

Confidence: 80%

Tokens: 279 input + 138 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	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



AI ASSESSMENT

Scenario:

tests/test_coverage_map.py::TestCoverageMapperContextExtraction::test_extract_nodeid_empty_context

Why Needed: To handle cases where the context is empty or None, allowing for proper coverage extraction.

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 + 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/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: Test coverage mapping

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

Key Assertions:

- {'assertion_type': 'Expected value to be None', 'expected_value': 'None'}

Confidence: 80%

Tokens: 139 input + 63 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/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_run_phase

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

Key Assertions:

- {'expected_value': 'test.py::test_foo', 'actual_value': 'test.py::test_foo'}

Confidence: 80%

Tokens: 145 input + 86 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/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 that the `contexts_by_lineno` method raises an exception when called with a mocked context that fails.

Why Needed: Prevents regression in handling unexpected exceptions during context extraction.

Key Assertions:

- mock_data.contexts_by_lineno.side_effect is set to contexts_side_effect
- call_count[0] should be incremented by 1 before raising the exception
- the exception raised is an instance of `Exception`
- the error message is 'Error accessing contexts'
- the context with ID 1 has a key-value pair with value ['test::test_foo|run']
- the number of calls to `contexts_by_lineno` is 2

Confidence: 80%

Tokens: 332 input + 155 output = 487 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario:

tests/test_coverage_map_coverage.py::TestExtractContexts::test_no_measured_files

Why Needed: To test that the function returns an empty dictionary when there are no measured files.

Key Assertions:

- {'name': 'result is a dictionary', 'expected_value': {}, 'message': 'The result should be an empty dictionary'}

Confidence: 80%

Tokens: 136 input + 90 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/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: Test Extract Contexts

Why Needed: Skip non-Python files

Key Assertions:

- {'assertion_type': 'Value comparison', 'expected_value': {}, 'actual_value': {}}

Confidence: 80%

Tokens: 154 input + 60 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	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



AI ASSESSMENT

Scenario: TestLoadCoverageData

Why Needed: The test is necessary because it checks for coverage when `coverage.py` is not installed.

Key Assertions:

- {'name': 'mapper is created successfully', 'expected_result': 'True'}

Confidence: 80%

Tokens: 166 input + 68 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/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



AI ASSESSMENT

Scenario: TestLoadCoverageData

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

Key Assertions:

- {'type': 'assertion', 'name': 'result is None', 'description': 'The function _load_coverage_data() should return None when no .coverage file exists.'}
- {'type': 'assertion', 'name': "any('W001' in w.code for w in mapper.warnings)", 'description': 'The function _load_coverage_data() should have a warning 'W001' in the warnings list.'}

Confidence: 80%

Tokens: 153 input + 139 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/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 the exception handling of analysis2 when it raises an exception.**Why Needed:** To prevent regression and ensure that warnings are properly generated when analysis2 raises an exception.**Key Assertions:**

- The test verifies that no coverage is added to the report when analysis2 raises an exception.
- The test verifies that a warning is generated with the message 'COVERAGE_ANALYSIS_FAILED' for each measured file.
- The test verifies that all warnings are properly handled and do not prevent further analysis or reporting.

Confidence: 80%**Tokens:** 286 input + 118 output = 404 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_coverage_map_coverage

Why Needed: To test the coverage map when there are no statements in a file.

Key Assertions:

- {'description': 'The function should return an empty list of lines covered by the code.', 'expected_result': []}

Confidence: 80%

Tokens: 178 input + 72 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/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: Prevents regression in coverage reporting when omitting tests from the coverage report.

Key Assertions:

- The `mapped_data` attribute of the `result` object should contain a single entry with 'covered' set to 2 and 'missed' set to 1.
- The `mapped_data` attribute of the `result` object should not be empty.
- The `covered` value in the first test data entry is correct (i.e., it has been measured by the coverage tool).
- The `missed` value in the first test data entry is correct (i.e., it does not have any unmeasured files).

Confidence: 80%

Tokens: 322 input + 168 output = 490 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test Map Source Coverage

Why Needed: Skip non-Python files

Key Assertions:

- {'message': 'Expected to skip non-Python files', 'description': 'The test should skip non-Python files'}

Confidence: 80%

Tokens: 154 input + 65 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/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



AI ASSESSMENT

Scenario: {'id': 'test_skip_test_files_when_configured', 'description': 'Test that test files are skipped when omit_tests_from_coverage is True.', 'key_assertions': [{'name': 'expected_result', 'type': 'list', 'value': []}]}

Why Needed: {'id': 'test_skip_test_files_when_configured', 'description': 'Test that test files are skipped when omit_tests_from_coverage is True.', 'key_assertions': [{'name': 'expected_result', 'type': 'list', 'value': []}]}

Confidence: 80%

Tokens: 182 input + 295 output = 477 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test that all phases are accepted when configured.

Why Needed: Prevents regression where some phases might be missed due to incorrect configuration.

Key Assertions:

- The mapper should return the nodeid for any phase in the config.
- The mapper should return the same nodeid for different phases (e.g., 'setup', 'run', 'teardown')
- All three assertions should pass with the same expected result

Confidence: 80%

Tokens: 305 input + 103 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	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



AI ASSESSMENT

Scenario:

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_empty_string

Why Needed: The test is necessary to ensure that the `extract_nodeid` function handles empty strings correctly.

Key Assertions:

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

Confidence: 80%

Tokens: 115 input + 90 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/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



AI ASSESSMENT

Scenario:

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_none

Why Needed: None input returns None.

Key Assertions:

- {'expected_value': 'None'}

Confidence: 80%

Tokens: 114 input + 58 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/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



AI ASSESSMENT

Scenario: Test that run phase is the default filter.**Why Needed:** Prevents unexpected behavior when run phases are not specified.**Key Assertions:**

- When the 'phase' parameter is missing, it should return the nodeid of the current file.
- When the 'phase' parameter does not match 'run', it should return None for that phase.
- When the 'phase' parameter does not match 'setup' or 'teardown', it should return None for those phases.

Confidence: 80%**Tokens:** 297 input + 114 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/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



AI ASSESSMENT

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

Why Needed: This test prevents a bug where the setup phase is incorrectly filtered, leading to false positives in coverage reports.

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



AI ASSESSMENT

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

Why Needed: This test prevents a regression where the teardown phase is not properly filtered, potentially leading to false positives in coverage reports.

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 + 127 output = 423 total

COVERAGE

src/pytest_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: To ensure that the node id is extracted correctly when there are no phase delimiters in the code.

Key Assertions:

- {'expected_value': 'test_foo.py::test_bar', 'actual_value': 'test_foo.py::test_bar'}

Confidence: 80%

Tokens: 136 input + 94 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	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: Test extracts all contexts for full logic coverage.**Why Needed:** Prevents regression in coverage analysis when `_extract_contexts` is called with a file that has multiple logical contexts.**Key Assertions:**

- `{'file_path': 'test_app.py::test_one', 'line_count': 2}`
- `{'file_path': 'test_app.py::test_two', 'line_count': 3}`

Confidence: 80%**Tokens:** 413 input + 94 output = 507 total

COVERAGE

src/pytest_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_contexts

Why Needed: To test the behavior of the `_extract_contexts` method when there are no test contexts.

Key Assertions:

- `{'name': 'assert result is empty', 'expected_value': {}, 'message': 'Expected result to be an empty dictionary, but got {}'}`

Confidence: 80%

Tokens: 174 input + 100 output = 274 total

COVERAGE

src/pytest_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: Test extracting node IDs for maximal coverage in different phases.**Why Needed:** This test prevents regression by ensuring that the `CoverageMapper` correctly extracts node IDs from code paths regardless of phase.**Key Assertions:**

- The function `'_extract_nodeid'` is called with a valid path and returns the expected node ID.
- The function `'_extract_nodeid'` is called with an invalid path (e.g., without a phase) and returns `None` as expected.
- The function `'_extract_nodeid'` is called with a context that does not contain any code paths (e.g., `test.py::test_no_phase`) and returns the expected node ID.

Confidence: 80%**Tokens:** 323 input + 150 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/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 test_load_coverage_data_no_files function correctly handles the case when no coverage files exist.

Why Needed: This test prevents a potential regression where the test might fail due to missing coverage data.

Key Assertions:

- The function should return None for _load_coverage_data() without any .coverage files.
- The number of warnings should be 1.
- The first warning code should be 'W001'.
- The current working directory should remain unchanged after the test.
- No coverage file should have been loaded into the mapper.
- The mapper's warnings list should contain only one item.

Confidence: 80%

Tokens: 276 input + 142 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	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 ensures that the CoverageMapper can handle errors reading coverage files correctly.

Why Needed: This test prevents a regression where the CoverageMapper fails to report coverage data when an error occurs while reading it.

Key Assertions:

- The function `_load_coverage_data()` should return `None` and set warnings to include 'Failed to read coverage data' when an exception is raised by the mock `CoverageData`.
- The function `_load_coverage_data()` should not raise any exceptions itself, but instead propagate them up the call stack.
- The function `_load_coverage_data()` should log the error message in the warnings list.
- The function `_load_coverage_data()` should set the 'error' level warning to include 'Failed to read coverage data'.
- The function `_load_coverage_data()` should not return any value (i.e., it should be `None`).

Confidence: 80%

Tokens: 343 input + 189 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/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 internal data structures.

Why Needed: This test prevents regression in handling parallel coverage files, ensuring that the CoverageMapper correctly updates its internal data structures when loading coverage data with multiple parallel files.

Key Assertions:

- The mock instances of `CoverageData` are returned by the `load_coverage_data()` method and updated accordingly.
- The `update()` method is called on the `CoverageData` instance for each mock instance.
- At least two calls to `update()` are made on the `CoverageData` instance.
- The `update()` method is called on the first mock instance (`mock_main_data`), and then on the second mock instance (`mock_parallel_data1`).
- The `update()` method is called on the third mock instance (`mock_parallel_data2`) but no call is made to it.
- No calls are made to the `update()` method on any other instances of `CoverageData`.
- The `update()` method is only called for the first two mock instances, and not for the third one.

Confidence: 80%

Tokens: 378 input + 248 output = 626 total

COVERAGE

src/pytest_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 test fails if there is no coverage data loaded.

Key Assertions:

- The `load_coverage_data` method of the `CoverageMapper` instance should return `None` when called with an empty dictionary.
- The `map_coverage` method of the `CoverageMapper` instance should return an empty dictionary when passed a non-empty dictionary.
- The test should not fail if there is no coverage data loaded, i.e., `load_coverage_data` returns None.

Confidence: 80%

Tokens: 228 input + 142 output = 370 total

COVERAGE

src/pytest_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 should handle errors during coverage analysis.

Why Needed: Prevents regression in case of unexpected errors during analysis, ensuring test coverage is not affected by unhandled exceptions.

Key Assertions:

- Mocked `analysis2` raises an exception when called with error message.
- Mocked `get_data` returns mock data but does not raise an exception.
- Mocked `map_source_coverage` skips files with errors.
- Expected no entries in the list of coverage results.
- Asserts that the length of the entries is 0, indicating no skipped files due to error.

Confidence: 80%

Tokens: 274 input + 136 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/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: Tests coverage of map_source_coverage method with comprehensive test data.**Why Needed:** Prevents regression in coverage analysis when testing all paths in map_source_coverage.**Key Assertions:**

- The method should return exactly one entry for the given mock data.
- The file path of the returned entry should be 'app.py'.
- The number of statements in the returned entry should be 3.
- The coverage percentage of the returned entry should be 66.67%.
- All covered lines should have a count greater than or equal to 1.
- No uncovered lines should have a count less than 2.
- All missing lines should have a count greater than 0.
- The coverage percentage of each line should add up to 100%.

Confidence: 80%**Tokens:** 345 input + 174 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/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)

PASSED

tests/test_errors.py::test_make_warning

1ms



AI ASSESSMENT

Scenario: Test the `make_warning` factory function with a valid warning code and message.

Why Needed: Prevents a potential bug where an unknown or invalid warning code is passed to the `make_warning` function, causing it to raise a `ValueError`.

Key Assertions:

- The returned object has the correct `code` attribute set to `WarningCode.W001_NO_COVERAGE`.
- The message of the returned object contains the expected string 'No .coverage file found'.
- The detail of the returned object matches the provided value 'test-detail'.

Confidence: 80%

Tokens: 236 input + 132 output = 368 total

COVERAGE

src/pytest_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 incorrect value is assigned to a warning code, potentially causing unexpected behavior or errors in downstream processing.**Key Assertions:**

- {'message': 'Assertion failed: WarningCode.W001_NO_COVERAGE.value == "W001"', 'expected': 'WarningCode.W001_NO_COVERAGE', 'actual': 'WarningCode.W001'}
- {'message': 'Assertion failed: WarningCode.W101_LLM_ENABLED.value == "W101"', 'expected': 'WarningCode.W101_LLM_ENABLED', 'actual': 'WarningCode.W101'}
- {'message': 'Assertion failed: WarningCode.W201_OUTPUT_PATH_INVALID.value == "W201"', 'expected': 'WarningCode.W201_OUTPUT_PATH_INVALID', 'actual': 'WarningCode.W201'}
- {'message': 'Assertion failed: WarningCode.W301_INVALID_CONFIG.value == "W301"', 'expected': 'WarningCode.W301_INVALID_CONFIG', 'actual': 'WarningCode.W301'}
- {'message': 'Assertion failed: WarningCode.W401_AGGREGATE_DIR_MISSING.value == "W401"', 'expected': 'WarningCode.W401_AGGREGATE_DIR_MISSING', 'actual': 'WarningCode.W401'}

Confidence: 80%**Tokens:** 240 input + 290 output = 530 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test ReportWarning.to_dict() method.**Why Needed:** Prevents a warning about the ReportWarning.to_dict() method being used with warnings that have no detail.**Key Assertions:**

- The 'code' key should be present in the dictionary with value 'W001'.
- The 'message' key should be present in the dictionary with value 'No coverage'.
- The 'detail' key should be present in the dictionary with value 'some/path'.

Confidence: 80%**Tokens:** 276 input + 112 output = 388 total

COVERAGE

src/pytest_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_know_code

1ms

3

AI ASSESSMENT

Scenario: Test creates warning with standard message for known code.

Why Needed: Prevents regression where a known code might not create a warning.

Key Assertions:

- w.code == WarningCode.W101_LLM_ENABLED
- w.message == WARNING_MESSAGES[WarningCode.W101_LLM_ENABLED]
- w.detail is None

Confidence: 80%

Tokens: 222 input + 82 output = 304 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: test_make_warning_unknown_code

Why Needed: To handle unknown code in the typed function.

Key Assertions:

- {'name': 'expected', 'type': 'assertion', 'message': "The message for missing warning code should be 'Unknown warning.'"}
 -

Confidence: 80%

Tokens: 202 input + 75 output = 277 total

COVERAGE

src/pytest_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



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_value': 'WarningCode.W301_INVALID_CONFIG'}
- {'name': "w.detail == 'Bad value'", 'expected_value': 'Bad value'}

Confidence: 80%

Tokens: 127 input + 107 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/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: Error Codes

Why Needed: The `test_codes_are_strings` test is checking that the WarningCode enum values are strings.

Key Assertions:

- {'name': 'assert isinstance(code.value, str)', 'description': 'Ensure that each code value is a string.'}
- {'name': "assert code.value.startswith('W')", 'description': "Ensure that each code starts with 'W', which indicates a warning code."}

Confidence: 80%

Tokens: 109 input + 112 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/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/test_errors_maximal.py::TestWarningDataClass::test_warning_to_dict_no_detail

Why Needed: To ensure that the warning data is correctly serialized without any additional details.

Key Assertions:

- {'assertion_type': 'equals', 'expected_value': {'code': 'W001', 'message': 'No coverage'}, 'actual_value': {'code': 'W001_NO_COVERAGE', 'message': 'No coverage'}}}

Confidence: 80%

Tokens: 147 input + 120 output = 267 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_fs.py::TestIsPythonFile::test_non_python_file**Why Needed:** Because the test is checking for non-.py files, which are not Python files.**Key Assertions:**

- {'message': 'Should return False', 'value': False}
- {'message': 'for non-.py files', 'value': 'The file does not have a .py extension.'}

Confidence: 80%**Tokens:** 115 input + 95 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/fs.py	1 lines (ranges: 79)

PASSED

tests/test_fs.py::TestIsPythonFile::test_python_file

1ms



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:**

- {'description': 'The function is called with a valid .py file path.', 'condition': "assert is_python_file('foo/bar.py') == True"}
- {'description': 'The function raises an error when given a non-.py file path.', 'condition': "is_python_file('non_py_file.txt')"}}

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/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



AI ASSESSMENT

Scenario: tests/test_fs.py::TestMakeRelative::test_makes_path_relative**Why Needed:** To ensure that the `make_relative` function correctly handles path relative paths.**Key Assertions:**

- {'assertion': 'The file path is absolute after making it relative.', 'expected_result': '/subdir/file.py'}

Confidence: 80%**Tokens:** 144 input + 84 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/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:

- {'name': 'result', 'expected_value': 'foo/bar'}

Confidence: 80%

Tokens: 107 input + 78 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	7 lines (ranges: 30, 33, 36, 39, 42, 55-56)

PASSED

tests/test_fs.py::TestNormalizePath::test_already_normalized

1ms



AI ASSESSMENT

Scenario: tests/test_fs.py::TestNormalizePath::test_already_normalized**Why Needed:** This test is needed because it checks if the `normalize_path` function correctly handles already-normalized paths.**Key Assertions:**

- {'name': 'assert normalize_path returns original path for already-normalized paths', 'expected_value': 'foo/bar', 'actual_value': "normalize_path('foo/bar') == 'foo/bar'"}}

Confidence: 80%**Tokens:** 96 input + 104 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/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:**

- {'path': 'foo\\bar', 'expected': 'foo/bar'}

Confidence: 80%**Tokens:** 100 input + 74 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/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': '/foo/bar/', 'actual': 'foo/bar/'}

Confidence: 80%**Tokens:** 102 input + 77 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/fs.py	5 lines (ranges: 30, 33, 36, 39, 42)

PASSED

tests/test_fs.py::TestShouldSkipPath::test_custom_exclude_patterns

1ms



AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_custom_exclude_patterns**Why Needed:** to test the custom exclusion of paths based on patterns**Key Assertions:**

- {'description': 'should skip a path matching a custom pattern', 'expected_result': True, 'actual_result': 'True'}
- {'description': 'should not skip a path not matching any custom pattern', 'expected_result': False, 'actual_result': 'False'}

Confidence: 80%**Tokens:** 126 input + 117 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/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:** This test ensures that the `should_skip_path` function does not incorrectly skip normal file system paths.**Key Assertions:**

- assert should_skip_path('src/module.py') is False

Confidence: 80%**Tokens:** 96 input + 67 output = 163 total

COVERAGE

src/pytest_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 test should be skipped when the path contains a .git directory.**Key Assertions:**

- {'expected': {'path': '.git/objects/foo'}, 'actual': {'should_be': True, 'is_true': False}}

Confidence: 80%**Tokens:** 99 input + 85 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/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:** The test should be skipped because it's trying to access a __pycache__ directory.**Key Assertions:**

- {'path': 'foo/__pycache__/bar.pyc', 'expected_result': True}

Confidence: 80%**Tokens:** 109 input + 83 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/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:** The test checks if the `should_skip_path` function correctly identifies venv directories as being to be skipped.**Key Assertions:**

- {'message': 'Assertion failed', 'expected': True, 'actual': False}

Confidence: 80%**Tokens:** 121 input + 80 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/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED

tests/test_fs_coverage.py::TestIsPythonFile::test_is_python_file_false

1ms



3

AI ASSESSMENT

Scenario: Non-.py files should not be considered Python files.**Why Needed:** This test prevents a potential bug where the function `is_python_file` incorrectly identifies non-.py files as Python files, leading to incorrect results or unexpected behavior in downstream code.**Key Assertions:**

- assert is_python_file('module.txt') is False
- assert is_python_file('module.pyc') is False
- assert is_python_file('module') is False

Confidence: 80%**Tokens:** 210 input + 108 output = 318 total

COVERAGE

src/pytest_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: The test verifies that a module file (.py) returns True.

Why Needed: This test prevents regression where the function `is_python_file` incorrectly identifies non-Python files.

Key Assertions:

- assert is_python_file('module.py') is True
- assert is_python_file('path/to/module.py') is True
- assert is_python_file(Path('module.py')) is True

Confidence: 80%

Tokens: 212 input + 97 output = 309 total

COVERAGE

src/pytest_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 sure `make_relative` returns the correct absolute path when the input path is not relative to the base.

Why Needed: Prevents a potential bug where `make_relative` would incorrectly return an absolute path for paths that are not under the base.

Key Assertions:

- The function should only include paths that are directly under the base in its result.
- The function should exclude any file extensions from being included in the result.
- The function should handle cases where the input paths have different levels of nesting (e.g., `project1/subdir/file.py` vs. `/path/to/project2/subdir/file.py`).
- The function should not include any redundant or unnecessary information in its output (e.g., file extensions, directory names).

Confidence: 80%

Tokens: 301 input + 172 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/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



AI ASSESSMENT

Scenario: Test Make Relative

Why Needed: To test the functionality of making relative paths in the file system.

Key Assertions:

- {'expected_output': 'subdir/file.py', 'actual_output': 'subdir/file.py'}

Confidence: 80%

Tokens: 147 input + 66 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/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



AI ASSESSMENT

Scenario: test_make_relative_with_none_base

Why Needed: To test the functionality of making a relative path with a None base.

Key Assertions:

- {'expected_value': 'path/to/file.py', 'actual_value': 'normalized_path'}

Confidence: 80%

Tokens: 116 input + 68 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/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:

- {'assertion': 'The normalized path contains a single forward slash.', 'expected_result': '/path/to/file.py'}

Confidence: 80%

Tokens: 114 input + 84 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	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: Normalization of a Path object is necessary to ensure consistency and correctness in file system operations.

Key Assertions:

- {'path': '/path/to/file.py', 'expected_result': 'path/to/file.py'}

Confidence: 80%

Tokens: 110 input + 81 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/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 the function correctly handles paths with trailing slashes.

Key Assertions:

- {'expected': '/path/to/dir/', 'actual': 'path/to/dir'}

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/fs.py	5 lines (ranges: 30, 33, 36, 39, 42)

PASSED

tests/test_fs_coverage.py::TestShouldSkipPath::test_should_not_skip_regular_path

1ms



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:

- {'name': 'should skip regular path', 'expected_result': {'scenario': 'False', 'why_needed': ''}, 'actual_result': {'scenario': 'True', 'why_needed': ''}}

Confidence: 80%

Tokens: 120 input + 96 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/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)

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_git

Why Needed: Because the test case checks for a specific path (\.git/hooks/pre-commit) and it's not present in the test directory.

Key Assertions:

- should be True

Confidence: 80%

Tokens: 102 input + 67 output = 169 total

COVERAGE

src/pytest_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 skips paths starting with a skip directory name.

Key Assertions:

- {'message': 'should be True', 'description': 'The function should return True for paths starting with a skip directory name'}
- {'message': '.venv', 'description': 'The function should return True for the path .venv'}

Confidence: 80%

Tokens: 124 input + 110 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/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 `__pycache__` directory contains a cached Python file that should be skipped.

Key Assertions:

- assert `should_skip_path('src/__pycache__/module.cpython-312.pyc')` is True

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/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



AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_site_packages

Why Needed: The test is checking if site-packages directories are skipped by the fs coverage tool.

Key Assertions:

- {'type': 'assertion', 'name': 'should_skip_path', 'value': '/usr/lib/python3.12/site-packages/pkg/mod.py'}

Confidence: 80%

Tokens: 111 input + 94 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)

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_venv

Why Needed: The test case checks 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



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 based on their content.

Key Assertions:

- {'name': 'should_skip_path', 'expected_result': True}
- {'name': 'assert_path', 'expected_result': False, 'message': "Expected 'src/module.py' to be included in the path"}

Confidence: 80%

Tokens: 132 input + 112 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/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: The test verifies that the GeminiProvider class's `_annotate_internal` method throws an error when it hits the daily limit for a model.

Why Needed: This test prevents regression where the provider does not raise an error when exceeding the daily limit for a model.

Key Assertions:

- assert 'Gemini requests-per-day limit reached' in res.error
- assert 'Daily limit exceeded' in res.error
- assert 'requests per day' in res.error
- assert 'limit hit' in res.error
- assert 'daily_limit_hit' in provider._rate_limiters['m1']
- mock_limiter.next_available_in.return_value == None
- provider._models == ['m1']

Confidence: 80%

Tokens: 367 input + 164 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/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` RPD is raised when rate limit is exceeded

Why Needed: To prevent the test from passing when rate limit is exceeded, we need to add a check for this scenario.

Key Assertions:

- Mocking `_GeminiRateLimiter` with `'requests_per_minute=100'`
- Setting `'_call_gemini'` to raise `'_GeminiRateLimitExceeded'` on the mock call
- `_GeminiRateLimitExceeded` should be raised when rate limit is exceeded
- The error message should contain 'requests-per-day' or 'rate limits reached'
- The model should be marked as exhausted after exceeding the rate limit
- No candidate models should remain in the pipeline after exhausting the model
- `_GeminiRateLimitExceeded` RPD should break the loop and mark model exhaustion

Confidence: 80%

Tokens: 730 input + 190 output = 920 total

COVERAGE

src/pytest_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 209ms 5

AI ASSESSMENT

Scenario: Prevents regression in coverage gaps by ensuring proper rate limiting and annotation logic.

Why Needed: This test prevents a bug where the GeminiProvider does not properly handle rate limiting and annotation logic, leading to coverage gaps.

Key Assertions:

- 1. Mock imports to avoid errors
- _rate_limiters['m1'] == _GeminiRateLimiter(requests_per_minute=100)
- _annotate_internal('TestCaseResult(nodeid=
- src
- custom)
- Context too long: Context too long error
- _parse_rate_limits([
- requests_per_day
- value=100])
- _ensure_models_and_limits('token') == ['fallback']
- models, limits = provider._fetch_available_models('token')
- assert limits['gemini-custom'] == 12345

Confidence: 80%

Tokens: 821 input + 185 output = 1006 total

COVERAGE

src/pytest_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

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/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_d
aily_requests

1ms



3

AI ASSESSMENT

Scenario: TestGeminiProvider tests**Why Needed:** To ensure the Gemini provider is correctly pruning daily requests that are older than 24 hours.**Key Assertions:**

- {'name': 'Limiter._daily_requests should be an empty list after prune', 'expected_result': [], 'actual_result': {'scenario': 'TestGeminiProvider tests', 'why_needed': 'To ensure the Gemini provider is correctly pruning daily requests that are older than 24 hours.', 'key_assertions': [{"name": "Limiter._daily_requests should be an empty list after prune", "expected_result": [], "actual_result": {}}, {"name": "assert len(limiter._daily_requests) == 0", "expected_result": 0, "actual_result": 0}], 'message': 'AssertionError: assert len(limiter._daily_requests) == 0 failed for test tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_prune_daily_requests'}}

Confidence: 80%**Tokens:** 157 input + 219 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/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



AI ASSESSMENT

Scenario: Verify that the test_tpm_available_fallback function waits for 30 seconds after using all tokens before returning.

Why Needed: This test prevents a regression where the Gemini provider does not wait for 30 seconds after using all tokens, potentially leading to unexpected behavior or errors.

Key Assertions:

- The `remaining` variable decreases by 50 each iteration of the loop.
- The loop finishes without returning if `remaining + request_tokens <= limit`.
- If `request_tokens` is massive (> 100), it should return 0.0 at line 106/108.

Confidence: 80%

Tokens: 524 input + 136 output = 660 total

COVERAGE

src/pytest_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 a 'google-generativeai' import error is reported when the module is not installed.

Why Needed: This test prevents a potential bug where the GeminiProvider does not report an import error for modules without the required library.

Key Assertions:

- The annotation contains the string 'google-generativeai not installed'.
- The annotation includes the key 'error' with the value containing the string 'google-generativeai not installed'.
- The annotation includes the key 'message' with the value containing the string 'google-generativeai not installed'.

Confidence: 80%

Tokens: 259 input + 132 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	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:** To prevent a test failure due to an unannotated `GEMINI_API_TOKEN` setting.**Key Assertions:**

- The error message should contain 'GEMINI_API_TOKEN is not set'.
- The annotation should report that the token is missing.
- The annotation should include the key 'GEMINI_API_TOKEN' in its error message.
- The annotation should indicate that the token was not found or not set.
- The annotation should provide a clear and concise error message.
- The annotation should be able to identify the specific environment variable being tested (in this case, 'GEMINI_API_TOKEN').
- The annotation should report any other relevant information about the failure (e.g., the test result itself).

Confidence: 80%**Tokens:** 313 input + 186 output = 499 total

COVERAGE

src/pytest_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 case of rate limit retries, ensuring that the provider can handle such scenarios without crashing or returning unexpected results.**Key Assertions:**

- The annotation returned by the `'_annotate_internal'` method matches the expected scenario.
- Mock `mock_post.call_count` is equal to 2 (first call with status code 429 and second call with status code 200).
- The `scenario` attribute of the annotation is set to 'Recovered Scenario'.
- No other critical checks are performed in this test. Only the scenario, retry status, and post call count assertions are made.
- Mock `mock_get.return_value.json.return_value` does not contain any unexpected data or errors.
- The `scenario` attribute of the annotation is set to 'Recovered Scenario' even if the provider returns an error.
- No other critical checks are performed in this test. Only the scenario, retry status, and post call count assertions are made.
- Mock `mock_get.return_value.status_code` is 429 (expected) or 200 (expected).
- The `scenario` attribute of the annotation is set to 'Recovered Scenario' even if the provider returns an error.

Confidence: 80%**Tokens:** 636 input + 279 output = 915 total

COVERAGE

src/pytest_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: Test that _annotate_success returns the correct scenario when successful**Why Needed:** Prevents regression in GeminiProvider where it incorrectly assumes a specific response format from _call_gemini.**Key Assertions:**

- The 'scenario' attribute of the annotation returned by _annotate_internal is set to 'Success Scenario'.
- The error attribute of the annotation returned by _annotate_internal is None.
- The 'error' attribute of the annotation returned by _parse_response is None.
- The 'scenario' attribute of the parsed response from _call_gemini matches the expected scenario.
- The 'tokens' attribute of the parsed response from _call_gemini is empty.
- The 'totalTokenCount' attribute of the parsed response from _call_gemini is 100.
- The 'candidates' attribute of the parsed response from _call_gemini contains a single successful candidate with no error message.
- The 'types' attribute of the parsed response from _call_gemini matches the expected types for successful scenarios.

Confidence: 80%**Tokens:** 649 input + 224 output = 873 total

COVERAGE

src/pytest_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 by setting environment variables and asserting the result.

Why Needed: This test prevents a potential bug where the provider does not detect when it has no available data.

Key Assertions:

- The `provider._check_availability()` method should return False when the GEMINI_API_TOKEN is set but the provider's configuration is 'gemini'.
- The `provider._check_availability()` method should return True when the GEMINI_API_TOKEN is not set and the provider's configuration is 'gemini'.
- The provider's `_check_availability()` method should raise an exception when the environment variable GEMINI_API_TOKEN is not set.
- When setting the GEMINI_API_TOKEN to a valid token, the provider's `_check_availability()` method should return True.
- When setting the GEMINI_API_TOKEN to an invalid token or empty string, the provider's `_check_availability()` method should raise an exception.

Confidence: 80%

Tokens: 235 input + 219 output = 454 total

COVERAGE

src/pytest_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_annotation_retry_exceptions 60.00s 4

AI ASSESSMENT

Scenario: Test that the GeminiProvider class correctly annotates retry exceptions when using a rate limiter with a daily limit and then again after cleanup.

Why Needed: This test prevents regression in cases where the provider is not able to handle retry exceptions properly due to rate limiting or other issues.

Key Assertions:

- Verify that the model exhaustion at node 't' for 'm1' is correctly annotated with a ResourceExhausted exception.
- Verify that the cooldowns for 'm1' are correctly set after a daily limit has been exceeded.
- Verify that the cooldowns for 'm1' can be set to a value greater than the current time after retrying after cleanup.
- Verify that the model exhaustion at node 't' for 'm1' is not retried after the cooldown period has expired.
- Verify that the provider correctly logs the error and sets the cooldowns for 'm1'.
- Verify that the provider correctly updates the _model_exhausted_at dictionary with the correct information.
- Verify that the provider correctly returns a valid result even when there are no more retries left.
-

Confidence: 80%

Tokens: 651 input + 249 output = 900 total

COVERAGE

src/pytest_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)

PASSED

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_annotate_retry_loop_coverage

3ms



AI ASSESSMENT

Scenario: Test that the GeminiProvider's _annotate_internal function clears _model_exhausted_at when models check pass.

Why Needed: This test prevents a regression where the GeminiProvider's _annotate_internal function fails to clear _model_exhausted_at after successful model checks.

Key Assertions:

- The value of _model_exhausted_at for the given model is not set before calling _annotate_internal.
- _model_exhausted_at is cleared when models check pass and their IDs are in provider._models.
- The value of _model_exhausted_at is not set after calling _annotate_internal.
- The value of _model_exhausted_at for the given model is set to a time in the past (24h ago) after calling _annotate_internal.
- _model_exhausted_at is cleared when models check pass and their IDs are in provider._models.
- The value of _model_exhausted_at is not set after calling _annotate_internal.
- The value of _model_exhausted_at for the given model is set to a time in the past (24h ago) after calling _annotate_internal.
-

Confidence: 80%

Tokens: 482 input + 256 output = 738 total

COVERAGE

src/pytest_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: Test GeminiProvider Detailed

Why Needed: To test the error handling of rate limiting in GeminiProvider

Key Assertions:

- {'assertion_type': 'Equal', 'expected_value': 10, 'actual_value': 0}

Confidence: 80%

Tokens: 156 input + 69 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	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: Test fetch available models with network error**Why Needed:** To test the error handling of GeminiProvider when it encounters a network error.**Key Assertions:**

- {'name': 'models are empty', 'description': 'The expected number of models is 0, but the actual result is an empty list.', 'expected_result': [], 'actual_result': []}
- {'name': 'limit_map is empty', 'description': 'The expected limit map is {}, but the actual result is an empty dictionary.', 'expected_result': {}, 'actual_result': {}}

Confidence: 80%**Tokens:** 132 input + 140 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/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

1ms



5

AI ASSESSMENT

Scenario: Verify that the GeminiProvider fetches available models with invalid JSON data.**Why Needed:** This test prevents a potential bug where the GeminiProvider incorrectly assumes valid JSON when fetching available models from an external API.**Key Assertions:**

- The 'm1' model is not included in the list of available models.
- The 'm2' model is not included in the list of available models.
- The 'm3' model is included in the list of available models.
- The 'inputTokenLimit' value for the 'm3' model does not match its supported generation methods.
- The 'limit_map' dictionary does not contain the expected key-value pairs for the 'm3' model.
- The GeminiProvider incorrectly assumes valid JSON when fetching available models from an external API.

Confidence: 80%**Tokens:** 340 input + 179 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/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_ensure` function when necessary.

Key Assertions:

- {'name': 'Mocked get_max_context_tokens call', 'expected_calls': [1], 'message': 'The `get_max_context_tokens` method should be called once.'}

Confidence: 80%

Tokens: 144 input + 115 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	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



AI ASSESSMENT

Scenario:

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_parse_rate_limits_types

Why Needed: To test the parsing of rate limits types and ensure they are correctly converted to Gemini provider configuration.

Key Assertions:

- {'name': 'config.requests_per_minute is None', 'expected_value': 'None'}
- {'name': 'config.tokens_per_minute == 100', 'expected_value': 100}

Confidence: 80%

Tokens: 156 input + 110 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	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



AI ASSESSMENT

Scenario: Test the _prune method of the _GeminiRateLimiter class to ensure it prunes requests within a certain time frame.

Why Needed: This test prevents a potential issue where requests are not pruned after a certain amount of time, potentially leading to excessive token usage or other problems.

Key Assertions:

- The length of _request_times should be equal to 1 after calling _prune with the current timestamp.
- The length of _token_usage should be equal to 1 after calling _prune with the current timestamp.
- The value in _request_times[0] should be equal to the current timestamp minus 10 seconds.

Confidence: 80%

Tokens: 323 input + 150 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/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_to_kens_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 does not handle invalid token counts correctly.

Key Assertions:

- {'name': 'assert len(limiter._token_usage) == 0', 'expected_result': 0, 'actual_result': 0}

Confidence: 80%

Tokens: 127 input + 96 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/llm/gemini.py	6 lines (ranges: 39-42, 66-67)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

AI ASSESSMENT

Scenario: Test the rate limiting feature

Why Needed: To ensure that the Gemini API does not exceed a certain number of requests per day without being throttled.

Key Assertions:

- {'name': 'Limiter should record the request and return None after 100 attempts', 'description': 'The limiter should record the request and return None after 100 attempts, indicating that it has reached its limit.'}

Confidence: 80%

Tokens: 129 input + 101 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/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)

AI ASSESSMENT

Scenario: Verify that the rate limiter does not block requests for a short period after the first two.

Why Needed: This test prevents a potential issue where the rate limiter blocks subsequent requests immediately after the initial two requests.

Key Assertions:

- limiter.next_available_in(100) == 0.0
- limiter.record_request()
- limiter.next_available_in(100) == 0.0
- limiter.record_request()

Confidence: 80%

Tokens: 280 input + 111 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/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



AI ASSESSMENT

Scenario: Verify that the rate limiter waits for tokens to expire before returning a 'TPM available' status when no tokens are requested or more than limit are used.

Why Needed: This test prevents a potential regression where the rate limiter incorrectly returns 'TPM available' when no tokens are requested, but usage exceeds the limit.

Key Assertions:

- The time until TPM availability is greater than 0 seconds when no tokens are requested or more than limit are used.
- The time until TPM availability does not exceed 60.0 seconds when more than limit are used.
- Tokens do not expire before reaching the limit of 100.
- Tokens do not expire after exceeding the limit by more than 10%.
- Tokens do not expire immediately when usage is within the limit but exceeds it.
- The rate limiter waits for tokens to expire before returning 'TPM available' when no tokens are requested or more than limit are used.

Confidence: 80%

Tokens: 377 input + 210 output = 587 total

COVERAGE

src/pytest_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 a `GeminiRateLimitExceeded` exception when the daily limit is exceeded.

Why Needed: This test prevents a potential regression where the rate limiter does not raise an exception when the daily limit is exceeded, potentially causing unexpected behavior or errors in downstream applications.

Key Assertions:

- The `wait_for_slot` method raises a `GeminiRateLimitExceeded` exception with the correct `limit_type` (in this case, 'requests_per_day')
- The error message indicates that the limit was exceeded
- The test verifies that the exception is raised only when the daily limit is exceeded

Confidence: 80%

Tokens: 263 input + 151 output = 414 total

COVERAGE

src/pytest_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 `wait_for_slot` sleeps for the expected amount of time when waiting for a slot to become available.

Why Needed: This test prevents regression in case the rate limiter is not properly implemented or if there are other factors affecting the sleep duration.

Key Assertions:

- The `next_available_in` method returns a value within the expected range (10.0s) when called with an argument equal to 1.
- The `wait_for_slot` method sleeps for at least 10.0 seconds before returning.
- The `mock_sleep.assert_called_once_with(10.0)` assertion ensures that the sleep function was called once with a value of 10.0.

Confidence: 80%

Tokens: 325 input + 157 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/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

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestComputeConfigHash::test_different_config

Why Needed: Different configs should produce different hashes.

Key Assertions:

- {'expected': 'different hashes', 'actual': 'same hashes'}

Confidence: 80%

Tokens: 119 input + 65 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	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 can be stored in a database or used as a unique identifier.

Key Assertions:

- {'name': 'assert len(h) == 16', 'expected_result': 16, 'message': 'Computed hash length should be 16 characters.'}

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/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 hash consistency with bytes

Why Needed: Test that the file hash matches the content hash for consistent results.

Key Assertions:

- {'expected_value': "b'test content'", 'actual_value': "compute_file_sha256(path).decode('utf-8') == compute_sha256(content)"}

Confidence: 80%

Tokens: 144 input + 81 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/hashing.py	6 lines (ranges: 32, 44-48)

PASSED

tests/test_hashing.py::TestComputeFileSha256::test_hashes_file

1ms



AI ASSESSMENT

Scenario: tests/test_hashing.py::TestComputeFileSha256::test_hashes_file

Why Needed: To test the correctness of file hashing.

Key Assertions:

- {'description': 'The hash should be a 64-byte string.', 'expected_value': 0}

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/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:

- {'message': 'Different keys should produce different signatures.', 'expected_result': 'different signature'}

Confidence: 80%

Tokens: 125 input + 72 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/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 that the HMAC computation is correct and produces the expected signature.

Key Assertions:

- {'expected_length': 64, 'actual_length': 0}

Confidence: 80%

Tokens: 108 input + 73 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/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:**

- {'message': 'Same content should produce same hash.', 'expected_result': 'The hash of two identical strings should be equal.'}

Confidence: 80%**Tokens:** 115 input + 88 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/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 hash length is correct and consistent across different inputs.**Key Assertions:**

- {'expected': 64, 'actual': 8, 'description': 'The actual length of the hash should be 8 bytes (64 hex chars)'}

Confidence: 80%**Tokens:** 103 input + 89 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/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 the 'pytest' package is included in the dependency snapshot.

Key Assertions:

- {'assertion_type': 'contains', 'value': 'pytest'}

Confidence: 80%

Tokens: 102 input + 74 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/hashing.py	8 lines (ranges: 113-114, 116-121)

PASSED

tests/test_hashing.py::TestGetDependencySnapshot::test_returns_dict 83ms 3

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestGetDependencySnapshot::test_returns_dict

Why Needed: To ensure the function correctly returns a dictionary representation of the dependency snapshot.

Key Assertions:

- {'name': 'snapshot is a dict', 'expected': 'True'}

Confidence: 80%

Tokens: 98 input + 73 output = 171 total

COVERAGE

src/pytest_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 the functionality of loading a HMAC key from a file.**Key Assertions:**

- {'expected': {'hmac_key_file': "tmp_path / 'hmac.key'"}, 'actual': {'hmac_key_file': 'str(key_file)'}}

Confidence: 80%**Tokens:** 145 input + 93 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	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



AI ASSESSMENT

Scenario: Test Load Hmac Key: Missing Key File**Why Needed:** The test should fail when a key file does not exist.**Key Assertions:**

- {'expected_type': 'NoneType', 'message': 'Expected the HMAC key to be None, but got '}

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:** To test that the function correctly returns None when no key file is configured.**Key Assertions:**

- {'name': 'assert key is None', 'description': 'The load_hmac_key function should return None if no key file is specified.'}

Confidence: 80%**Tokens:** 110 input + 90 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/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: Test the default aggregation configuration.

Why Needed: Prevents a regression where aggregation defaults are not set correctly.

Key Assertions:

- config.aggregate_dir is None (default aggregation directory should be empty)
- config.aggregate_policy == 'latest' (default aggregation policy is 'latest')
- config.aggregate_include_history is False (default aggregation include history is False)

Confidence: 80%

Tokens: 201 input + 90 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/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': 'config', 'type': 'assertion', 'value': 'get_default_config() is not None'}

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/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 that the context mode is set to 'minimal' by default.

Key Assertions:

- {'name': 'config.llm_context_mode', '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



AI ASSESSMENT

Scenario: tests/test_integration_gate.py::TestConfigDefaults::test_llm_not_enabled_by_default

Why Needed: The LLM is not enabled by default.

Key Assertions:

- {'name': 'is_llm_enabled', 'expected_result': False, 'actual_result': 'get_default_config().is_llm_enabled() == False'}

Confidence: 80%

Tokens: 109 input + 89 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	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



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', 'expected_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_de
fault_none

1ms



AI ASSESSMENT

Scenario: tests/test_integration_gate.py::TestConfigDefaults::test_provider_default_none

Why Needed: To ensure the provider is set to None by default.

Key Assertions:

- {'name': 'Provider should be None', 'expected_value': 'none'}

Confidence: 80%

Tokens: 101 input + 71 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/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



AI ASSESSMENT

Scenario: tests/test_integration_gate.py::TestConfigDefaults::test_secret_exclude_globs

Why Needed: This test is needed because the default configuration includes secret files and environment variables by default.

Key Assertions:

- {'name': 'Excluding secret files and environment variables', 'description': "The function should exclude 'secret' and '.env' files from the list of excluded globs.", 'expected_result': ['exclude_globs']}

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	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 output of a deterministic pipeline is sorted by nodeid.

Why Needed: This test prevents regression where the order of nodeids in the report changes unexpectedly.

Key Assertions:

- nodeids should be sorted alphabetically and numerically
- nodeids should not contain duplicates
- nodeids should only include unique values from the input data
- nodeid 'z_test.py::test_z' should always come first in the list
- nodeid 'a_test.py::test_a' should come second in the list
- nodeid 'm_test.py::test_m' should come third in the list

Confidence: 80%

Tokens: 313 input + 149 output = 462 total

COVERAGE

src/pytest_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_suite 10ms 6

AI ASSESSMENT

Scenario: Test that an empty test suite produces a valid report.**Why Needed:** This test prevents a regression where the test suite is empty and still produces a valid report.**Key Assertions:**

- The total count of tests in the report should be zero.
- The 'summary' section of the report should have a 'total' key with a value of zero.
- There should be no failed or skipped tests in the report.
- All test names and descriptions should be present in the report.
- The report format should be valid JSON as expected by the framework.
- The test suite should not contain any invalid or missing data.

Confidence: 80%**Tokens:** 240 input + 146 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/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_ 44ms 6
generation

AI ASSESSMENT

Scenario: The test verifies that the full pipeline generates an HTML report.**Why Needed:** This test prevents regression where the HTML report is not generated correctly due to a change in configuration settings.**Key Assertions:**

- The path to the generated HTML report exists and can be accessed.
- The content of the HTML report contains the string '
- The string 'test_pass' is present in the content of the HTML report as expected.

Confidence: 80%**Tokens:** 270 input + 107 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/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 65ms 7

AI ASSESSMENT

Scenario: The test verifies that a full pipeline generates a valid JSON report with the correct schema version, summary statistics, and number of tests.

Why Needed: This test prevents regression where the JSON report generation is not producing a valid output or contains incorrect information.

Key Assertions:

- data['schema_version'] == SCHEMA_VERSION
- data['summary']['total'] == 3
- data['summary']['passed'] == 1
- data['summary']['failed'] == 1
- data['summary']['skipped'] == 1

Confidence: 80%

Tokens: 419 input + 134 output = 553 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: Test that the ReportRoot class has required fields when created.

Why Needed: This test prevents a potential bug where the report root is missing required fields.

Key Assertions:

- The 'schema_version' field should be present in the data.
- The 'run_meta' field should be present in the data.
- The 'summary' field should be present in the data.
- The 'tests' field should be present in the data.

Confidence: 80%

Tokens: 250 input + 108 output = 358 total

COVERAGE

src/pytest_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



COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test "RunMeta has run status fields" verifies that the `to_dict()` method returns a dictionary with required status fields.

Why Needed: This test prevents regression where `RunMeta` does not have a `to_dict()` method or its output 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 + 151 output = 388 total

COVERAGE

src/pytest_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, which is necessary for compatibility with older versions of the gate.

Key Assertions:

- {'name': 'SCHEMA_VERSION', 'type': 'string'}
- {'name': '!', 'type': 'boolean'}

Confidence: 80%

Tokens: 103 input + 93 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)

PASSED

tests/test_integration_gate.py::TestSchemaCompatibility::test_test_case_has_required_fields 1ms 3

AI ASSESSMENT

Scenario: The `TestSchemaCompatibility` class verifies that the `TestCaseResult` object has all required fields.

Why Needed: This test prevents a potential bug where the `TestCaseResult` object is missing one of its required fields, which could lead to incorrect analysis or reporting.

Key Assertions:

- The 'nodeid' key should be present in the data dictionary.
- The 'outcome' key should be present in the data dictionary.
- The 'duration' key should be present in the data dictionary.

Confidence: 80%

Tokens: 223 input + 119 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	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



AI ASSESSMENT

Scenario: Test that all retries are exhausted when API call fails.**Why Needed:** Prevents regression where LiteLLMProvider returns an annotation with no error even when all retries are exhausted.**Key Assertions:**

- The `result` variable is not None.
- The `result.error` attribute is not None.
- The `provider.annotate()` method raises an exception with the message 'API error'.
- The `mock_completion` function is called with a side effect that raises an exception.
- The `test_source` is set to a function that does not raise any exceptions.
- The `context_files` dictionary is empty.

Confidence: 80%**Tokens:** 346 input + 147 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/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 test verifies that the annotation returns an annotation with an error status code when a non-401 error is encountered.
- The test verifies that the annotation does not force token refresh when a non-401 error is encountered.
- The test verifies that the annotation correctly handles an internal server error (500) instead of 401.
- The test verifies that the annotation raises an exception with the 'Internal server error' message when a non-401 error is encountered.
- The test verifies that the annotation does not raise an exception when a non-401 error is encountered, allowing for token refresh to continue.
- The test verifies that the annotation correctly returns None in case of no error.
- The test verifies that the annotation correctly sets the 'error' attribute on the result object to None.

Confidence: 80%

Tokens: 367 input + 215 output = 582 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test that retry succeeds after transient error and ensures the test fails with a meaningful error message.

Why Needed: To prevent the test from succeeding immediately after a transient error, allowing for retries to be attempted.

Key Assertions:

- The `result.error` attribute is `None` when the test should fail due to a transient error.
- The `test.scenario` attribute is set to "test scenario" even though an error occurred.
- The `test.context_files` dictionary remains empty after the API call fails twice, then succeeds.
- The `mock_completion` function raises an exception when the API call fails for the second time.
- The `result` object contains a meaningful message indicating that the test failed due to a transient error.

Confidence: 80%

Tokens: 433 input + 170 output = 603 total

COVERAGE

src/pytest_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

6.37s



7

AI ASSESSMENT

Scenario: Test that 401 error triggers token refresh (test_token_refresh_on_401)**Why Needed:** To ensure the LLM token refresh mechanism works correctly for 401 errors.**Key Assertions:**

- The test verifies that the LLM provider calls the `token_refresh_command` with a new token when an API call fails with a 401 status code.
- The test checks that the `litellm_token_output_format` is set to 'text' for the output of the `token_refresh_command`.
- The test verifies that the `mock_completion` function is called twice, once with a 401 error and once without it.
- The test ensures that the `result` variable is not `None` after the token refresh attempt.
- The test checks that the `call_count[0] >= 2` condition is met to verify that the LLM provider has retried after token refresh.
- The test verifies that the `test_source` and `context_files` are empty for the annotated result.
- The test ensures that the `error` variable is raised with a 401 status code when the API call fails.

Confidence: 80%**Tokens:** 473 input + 252 output = 725 total

COVERAGE

src/pytest_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: To ensure that the GeminiProvider is correctly instantiated when using the 'gemini' provider.

Key Assertions:

- {'name': 'provider.__class__.__name__', 'expected_value': 'GeminiProvider'}

Confidence: 80%

Tokens: 131 input + 83 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	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



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 ('litellm') is used.**Key Assertions:**

- {'name': 'provider.__class__.__name__', 'expected': 'LiteLLMProvider'}

Confidence: 80%**Tokens:** 140 input + 90 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	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:** To ensure that the GetProvider function returns a NoopProvider when the provider is set to 'none'.**Key Assertions:**

- {'name': 'provider is None', 'expected': 'NoopProvider', 'actual': 'None'}

Confidence: 80%**Tokens:** 115 input + 90 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/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 class is correctly instantiated when a specific provider ('ollama') is used.

Key Assertions:

- {'name': 'provider', 'expected_value': 'OllamaProvider'}

Confidence: 80%

Tokens: 154 input + 85 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/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: Tests the implementation of NoopProvider.

Why Needed: Prevents a regression where NoopProvider does not implement required interface methods.

Key Assertions:

- The 'annotate' method should be present in the provider.
- The 'is_available' method should be present in the provider.
- The 'get_model_name' method should be present in the provider.
- The 'config' attribute should be present in the provider.

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/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



AI ASSESSMENT

Scenario: The test verifies that the `annotate` method of a NoopProvider returns an empty annotation when no annotation is provided.

Why Needed: This test prevents regression in case the `annotate` method does not return an annotation if none is provided.

Key Assertions:

- annotation is always an instance of 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 + 110 output = 359 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_llm.py::TestNoopProvider::test_get_model_name_empty**Why Needed:** The test is failing because the `get_model_name` method of the `NoopProvider` class does not handle an empty input correctly.**Key Assertions:**

- {'name': 'Expected value', 'value': '', 'expected_type': ''}

Confidence: 80%**Tokens:** 114 input + 91 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/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 because the NoopProvider is not properly initialized.**Key Assertions:**

- {'name': 'provider.is_available() should return True', 'expected_value': True, 'actual_value': 'assert provider.is_available() is True'}

Confidence: 80%**Tokens:** 108 input + 92 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	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



AI ASSESSMENT

Scenario: Test that the schema requires 'scenario' and 'why_needed'.

Why Needed: The schema requires these fields because they are specified in the ANNOTATION_JSON_SCHEMA.

Key Assertions:

- {'message': "The field 'scenario' is required.", 'description': "The field 'scenario' must be present in the annotation."}
- {'message': "The field 'why_needed' is required.", 'description': "The field 'why_needed' must be present in the annotation."}

Confidence: 80%

Tokens: 115 input + 124 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/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 data corruption or incorrect parsing of user input.

Key Assertions:

- The 'scenario' field is set to the expected value.
- The 'why_needed' field is set to the expected value.
- The 'key_assertions' list contains the correct number of elements.
- The confidence level is set to the expected value (0.95).
-

Confidence: 80%

Tokens: 274 input + 113 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/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: Tested that the AnnotationSchema class correctly returns an empty string when given an empty dictionary.

Why Needed: The test is necessary because the AnnotationSchema class should handle empty input and return a valid object.

Key Assertions:

- {'name': 'schema.scenario', 'value': '', 'expected_type': 'str'}
- {'name': 'schema.why_needed', 'value': '', 'expected_type': ''}

Confidence: 80%

Tokens: 109 input + 111 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/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 needed to ensure that the AnnotationSchema handles partial input correctly.

Key Assertions:

- {'assertion': "schema.scenario should be 'Partial only'", 'expected_value': 'Partial only'}
- {'assertion': 'schema.why_needed should be an empty string', 'expected_value': ''}

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/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` function is called with a JSON schema that contains required fields.

Why Needed: This test prevents a potential bug where the `schema_has_required_fields` function is not called with a valid JSON schema.

Key Assertions:

- The `schema_has_required_fields` function should be called with a JSON schema that has 'scenario', 'why_needed', and 'key_assertions' properties.
- The `schema_has_required_fields` function should validate that the required fields are present in the JSON schema.
- The `schema_has_required_fields` function should check for the presence of 'scenario', 'why_needed', and 'key_assertions' keys in the JSON schema.
- The `schema_has_required_fields` function should perform checks to ensure these required fields are included in the JSON schema.
- The `schema_has_required_fields` function should validate that the required fields meet their respective validation rules.
- The `schema_has_required_fields` function should report an error if any of the required fields are missing or invalid.
- The `schema_has_required_fields` function should provide detailed information about why a field is required (if applicable).

Confidence: 80%

Tokens: 215 input + 259 output = 474 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: This test verifies that the `AnnotationSchema` class correctly serializes to a dictionary.

Why Needed: This test prevents regression by ensuring that the `AnnotationSchema` class handles scenario and why needed information properly.

Key Assertions:

- assertion 1: The 'scenario' key in the serialized data matches the expected value.
- assertion 2: The 'why_needed' key in the serialized data matches the expected value.
- assertion 3: The 'key_assertions' list in the serialized data contains all expected assertions.

Confidence: 80%

Tokens: 247 input + 127 output = 374 total

COVERAGE

src/pytest_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)

AI ASSESSMENT

Scenario: tests/test_llm_contract.py::TestNoopProvider::test_noop_from_factory

Why Needed: The test is necessary because the NoopProvider class does not implement the required interface.

Key Assertions:

- {'name': 'provider', 'expected_value': 'noop', 'actual_value': 'None'}

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/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: NoopProvider returns empty annotation when no function is annotated with 'noop'

Why Needed: To prevent a test from failing due to an empty annotation in the NoopProvider.

Key Assertions:

- result.scenario is an empty string
- result.why_needed is an empty string
- 1

Confidence: 80%

Tokens: 253 input + 76 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/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



AI ASSESSMENT

Scenario: The test verifies that the annotate method returns a TestCaseResult object with the correct attributes.

Why Needed: This test prevents regression in the LlmAnnotation-like object annotation process, ensuring it correctly handles scenarios where annotations are not present or have incorrect attributes.

Key Assertions:

- The result has an attribute named 'scenario' that contains a string describing the scenario.
- The result has an attribute named 'why_needed' that contains a string explaining why this test is necessary.
- The result has an attribute named 'key_assertions' that stores the critical checks performed during the test.

Confidence: 80%**Tokens:** 263 input + 134 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/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



AI ASSESSMENT

Scenario:

tests/test_llm_contract.py::TestProviderContract::test_provider_handles_empty_code

Why Needed: To ensure that the provider handles empty code gracefully and returns a valid result.

Key Assertions:

- {'assertion_type': 'is not None', 'expected_result': 'None'}

Confidence: 80%

Tokens: 145 input + 78 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/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



AI ASSESSMENT

Scenario: Provider handles None context gracefully**Why Needed:** The contract should be able to handle a None context without raising an error.**Key Assertions:**

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

Confidence: 80%**Tokens:** 148 input + 67 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/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:

- {'expected_type': 'function', 'actual_type': 'callable', 'message': 'Expected annotate to be a callable. Got {0}.}'}

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/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

Why Needed: This test is necessary because the `annotate_handles_context` method of the `GeminiProvider` class has a time complexity of $O(n)$, where n is the number of handles. This can cause performance issues when dealing with large numbers of handles.

Key Assertions:

- {'name': 'Test that annotate_handles_context does not raise an exception for small inputs', 'expected_result': 'No exception is raised'}
- {'name': 'Test that annotate_handles_context returns the expected result for a small input', 'expected_result': {'scenario': '...', 'why_needed': '...', 'key_assertions': ['...']}}}

Confidence: 80%**Tokens:** 98 input + 161 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	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)

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_missing_dependency 1ms ⚡ 5

AI ASSESSMENT

Scenario: Test that the LiteLLM provider correctly reports a missing dependency when an import error occurs.

Why Needed: This test prevents a potential bug where the provider does not report a missing dependency when an import error occurs, potentially masking a dependency issue.

Key Assertions:

- The annotation message includes the correct dependency name and installation instructions.
- The annotation message is not empty or null.
- The annotation message contains the expected error message.
- The annotation message does not contain any other relevant information.
- The annotation message is not too long (less than 50 characters).
- The annotation message includes the correct dependency name and installation instructions.
- The annotation message is a string, not a list or tuple.
- The annotation message contains only one line of text.

Confidence: 80%

Tokens: 270 input + 175 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/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 the GeminiProvider raises an error when the API token is missing.

Why Needed: This test prevents a potential bug where the 'annotate' method fails to raise an error when the API token is missing, potentially leading to unexpected behavior or errors in downstream code.

Key Assertions:

- The 'error' attribute of the annotation object should be set to 'GEMINI_API_TOKEN is not set'.
- The 'error' attribute of the annotation object should be set to 'GEMINI_API_TOKEN is not set'.
- The 'error' attribute of the annotation object should be set to 'GEMINI_API_TOKEN is not set'.

Confidence: 80%

Tokens: 440 input + 157 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	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 bug in the `GeminiProvider` class where token usage is not properly recorded.

Key Assertions:

- The `annotate_records_tokens` test verifies that the provider records token usage correctly.
- The provider's `_rate_limiters` dictionary contains an entry for 'gemini-1.5-pro' with a single entry under '_token_usage'.
- The value of '_token_usage' is 123, indicating that at least one token was recorded.
- The test also verifies that the rate limits logic runs without error.

Confidence: 80%

Tokens: 783 input + 155 output = 938 total

COVERAGE

src/pytest_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 retry annotating tasks when rate limiting occurs.

Key Assertions:

- {'name': 'Llama model is retried', 'description': 'The Llama model should be retried after a certain number of attempts.'}
- {'name': 'Error message is displayed', 'description': 'An error message should be displayed when the rate limit is exceeded.'}

Confidence: 80%

Tokens: 98 input + 116 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	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 LLM model is rotated on a daily basis and that the annotation process does not interfere with this rotation.**Key Assertions:**

- {'name': 'model_rotation', 'description': 'The model rotation should be applied every day'}
- {'name': 'annotation_process', 'description': 'The annotation process should not interfere with the model rotation'}

Confidence: 80%**Tokens:** 100 input + 125 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	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**Why Needed:** skips on daily limit because it's a performance bottleneck and LLMs are not designed to handle large amounts of data in a single session.**Key Assertions:**

- Llama model is not handling large amount of data in a single session efficiently

Confidence: 80%**Tokens:** 98 input + 74 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/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 valid response with the correct information.**Why Needed:** Prevents regression by ensuring the provider correctly annotates responses from LiteLLM.**Key Assertions:**

- status ok
- redirect

Confidence: 80%**Tokens:** 474 input + 64 output = 538 total

COVERAGE

src/pytest_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_mode_l_recovers_after_24h

1ms



6

AI ASSESSMENT

Scenario: tests/test_llm_providers.py**Why Needed:** To ensure that the LLM provider can recover from an exhausted model after 24 hours.**Key Assertions:**

- The recovered model's performance is within a reasonable range.
- No additional warnings or errors are raised when using the recovered model.

Confidence: 80%**Tokens:** 104 input + 74 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/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 handle the case where there are no available models.

Key Assertions:

- {'name': 'Expected error message', 'value': 'No available models found.'}
- {'name': 'Expected error type', 'value': 'LlamaProviderError'}

Confidence: 80%

Tokens: 92 input + 88 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/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: The `model_list_refresh` method of the `GeminiProvider` class is called after an interval.

Why Needed: To ensure that the model list is updated correctly and consistently across different runs.

Key Assertions:

- {'name': 'Model list should be refreshed after a certain interval', 'expected_value': 'The model list should be updated with new data after a specific time period.'}

Confidence: 80%**Tokens:** 96 input + 99 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	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 the retry mechanism for LiteLLM provider on a 401 error after token refresh.**Why Needed:** The test prevents a regression where the LLM provider does not retry when it encounters a 401 Unauthorized error after refreshing its token.**Key Assertions:**

- Verify that the LLM provider retries after a 401 Unauthorized error and captures the refreshed token.
- Verify that the LLM provider fails with a 401 Unauthorized error on the first attempt to refresh its token.
- Verify that the LLM provider succeeds with a new token after refreshing its token.
- Verify that the captured keys match the expected tokens for both attempts.
- Verify that the retry count is incremented correctly for each attempt.
- Verify that the error message is correct and includes the expected reason (401 Unauthorized).
- Verify that the LLM provider uses the new token in subsequent requests without any issues.

Confidence: 80%**Tokens:** 580 input + 197 output = 777 total

COVERAGE

src/pytest_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: Test that the LiteLLMProvider annotates completion errors correctly.

Why Needed: This test prevents regression when a completion error occurs during annotation.

Key Assertions:

- assert 'boom' in annotation.error
- assert annotation.error is not None

Confidence: 80%

Tokens: 307 input + 64 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/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_invaid_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 the LiteLLMProvider.**Key Assertions:**

- The 'response_data' dictionary must contain a list of key_assertion payloads.
- The 'response_data' dictionary must not be empty.
- The 'response_data' dictionary should have at least one key_assertion payload.
- The 'response_data' dictionary should not be None.
- The 'response_data' dictionary should have a valid JSON string.

Confidence: 80%**Tokens:** 346 input + 128 output = 474 total

COVERAGE

src/pytest_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: Test that a LiteLLMProvider annotates missing dependencies correctly.

Why Needed: This test prevents a bug where the provider does not report missing dependencies and instead silently installs them.

Key Assertions:

- The annotation error message is set to 'litellm not installed. Install with: pip install litellm' as expected.
- The provider correctly reports an import error for the missing dependency.
- The test passes even when the mock_import_error function returns a different error message.

Confidence: 80%

Tokens: 271 input + 115 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	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 LiteLLMProvider annotates a successful response with the correct information.

Why Needed: Prevents regressions by ensuring the provider correctly handles successful responses.

Key Assertions:

- status ok
- redirect
- model gpt-4o
- role system
- tests/test_auth.py::test_login

Confidence: 80%

Tokens: 475 input + 85 output = 560 total

COVERAGE

src/pytest_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:** This test prevents a bug where the provider does not override the prompt for custom prompts.**Key Assertions:**

- The function `provider._annotate_internal` is called with the correct argument `prompt_override = 'CUSTOM PROMPT'`.
- The captured messages from the fake completion are added to the list of expected messages. The message should contain a key named 'content'.
- The value of the 'content' key in the captured message matches the expected value 'CUSTOM PROMPT'.

Confidence: 80%**Tokens:** 373 input + 132 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/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 extracts token usage from response.**Why Needed:** The test prevents a potential bug where the provider does not accurately calculate token usage in certain scenarios.**Key Assertions:**

- The `prompt_tokens` attribute of the `token_usage` object is set to the correct value (100).
- The `completion_tokens` attribute of the `token_usage` object is set to the correct value (50).
- The `total_tokens` attribute of the `token_usage` object is set to the correct value (150).

Confidence: 80%**Tokens:** 426 input + 129 output = 555 total

COVERAGE

src/pytest_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: tests/test_llm_providers.py::TestLiteLLMProvider::test_api_base_passthrough verifies that the LiteLLM provider passes `api_base` to the completion call.

Why Needed: This test prevents regression where the API base is not passed correctly to the completion call, potentially causing issues with downstream integrations or data processing.

Key Assertions:

- The `api_base` attribute of the LiteLLM provider should be set to `https://proxy.corp.com/v1` before calling the completion function.
- The `litellm_api_base` configuration parameter should be set to `https://proxy.corp.com/v1` for the test case to pass.
- The `api_base` attribute of the LiteLLM provider should not be modified after setting it in the test configuration.
- The completion function should return a response with the correct `api_base` value if it is set correctly.
- If the `api_base` attribute is not set, the completion function should raise an exception or return an error message indicating that the API base was not provided.
- The `litellm_api_base` configuration parameter should be set to a valid URL for the test case to pass.
- If the `litellm_api_base` configuration parameter is not set correctly, the test case should fail with an error message indicating that the API base was not provided.

Confidence: 80%

Tokens: 387 input + 301 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/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_passed 1ms ⚡ 6
through

AI ASSESSMENT

Scenario: The `liteellm` provider passes the static API key to the completion call.

Why Needed: This test prevents a regression where the API key is not passed through to the completion function.

Key Assertions:

- The API key is captured and stored in the `captured` dictionary.
- The API key matches the expected value 'static-key-placeholder' as per the `response_data`.
- The `litellm_api_key` attribute of the `liteellm` object is set to 'static-key-placeholder'.

Confidence: 80%

Tokens: 384 input + 124 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	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

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/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: The test verifies that the LiteLLMProvider does not retry authentication if it fails on the second attempt.

Why Needed: This test prevents a bug where the provider retries authentication even after failing with an auth error, which could lead to unexpected behavior or errors in certain scenarios.

Key Assertions:

- The test checks that the annotation of the provider contains an 'error' key indicating an authentication failure.
- The test verifies that the 'error' value includes the string 'Authentication failed',
- The test ensures the 'error' value does not contain any other relevant information, such as the token or refresh command used to authenticate.
- The test checks that the provider's annotation is set correctly even after a second authentication attempt fails.
- The test verifies that the provider returns an error message indicating that authentication failed when it encounters this scenario.
- The test ensures that the provider does not retry authentication if it fails on the first attempt, as intended by the bug being tested.

Confidence: 80%

Tokens: 419 input + 221 output = 640 total

COVERAGE

src/pytest_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)

PASSED tests/test_llm_providers.py::TestLiteLLMProvider::test_context_too_long_error 1ms ⚡ 6

AI ASSESSMENT

Scenario: The test verifies that the LiteLLMProvider class correctly handles a context too long error when parsing a response.

Why Needed: This test prevents a regression where the provider fails to handle invalid responses with an error message.

Key Assertions:

- The `_parse_response` method of the provider should return `None` for an invalid response.
- The `_parse_response` method of the provider should raise a `ValueError` for an invalid response.
- The `error` attribute of the annotation returned by `_parse_response` should be set to '`Context too long for this model`'.
- The `key_assertions` in the annotation returned by `_parse_response` should include '`scenario`', '`why_needed`', and '`error`'.
- The value of the '`error`' key in the annotation returned by `_parse_response` should match the expected error message.
- The value of the '`key_assertions`' list in the annotation returned by `_parse_response` should be a list containing '`scenario`', '`why_needed`', and '`error`'.

Confidence: 80%

Tokens: 370 input + 218 output = 588 total

COVERAGE

src/pytest_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: test_get_max_context_tokens_dict_format

Why Needed: To ensure the correct dictionary format is returned when getting max context tokens.

Key Assertions:

- {'expected_value': 16384, 'actual_value': 16384}

Confidence: 80%

Tokens: 218 input + 67 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/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 LLMProvider can handle errors and return a fallback value for max context tokens.

Key Assertions:

- {'name': 'max_context_tokens', 'expected_value': 16, 'actual_value': 0}
- {'name': 'fallback_on_error', 'expected_value': 'Fallback token'}

Confidence: 80%

Tokens: 101 input + 101 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	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



AI ASSESSMENT

Scenario:

tests/test_llm_providers.py::TestLiteLLMProvider::test_get_max_context_tokens_success

Why Needed: To test the LiteLLM provider's ability to get the maximum context tokens from the litellm module.

Key Assertions:

- {'name': 'expected result', 'type': 'int', 'value': 8192}

Confidence: 80%

Tokens: 213 input + 93 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/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



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': 'setitem', 'expected_value': {'litellm': 'fake_litellm'}}}

Confidence: 80%

Tokens: 160 input + 83 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/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 properly refreshed, causing the LLM to use outdated tokens.**Key Assertions:**

- The API key should be 'dynamic-token-789' after token refresh.
- The provider should correctly set the API key in the configuration.
- The token refresh command and interval should be correctly configured.
- The LitellmProvider should properly handle token refresh requests.
- The case result should indicate that the test passed successfully.
- The captured data from the subprocess call should match the expected response.

Confidence: 80%**Tokens:** 442 input + 145 output = 587 total

COVERAGE

src/pytest_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_err or_retry

1ms



AI ASSESSMENT

Scenario: Test 'LiteLLM provider retries on transient errors'.**Why Needed:** Prevents a regression where the LLM provider fails to retry transient errors.**Key Assertions:**

- The function `fake_completion` raises a `ConnectionError` within 3 calls.
- The function `fake_completion` raises a `ConnectionError` within 2 additional calls.
- The function `fake_completion` does not raise a `ConnectionError` after the third call.
- The LLM provider's retry count is reset to 0 after each call.
- The LLM provider retries on transient errors, ensuring it can recover from such events.

Confidence: 80%**Tokens:** 426 input + 147 output = 573 total

COVERAGE

src/pytest_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:**

- {'name': 'Llama model is not properly initialized', 'description': 'The Llama model should be properly initialized before using it for annotation.'}
- {'name': 'Context length error is handled correctly', 'description': 'The context length error should be handled correctly by the LLM provider and return a fallback solution.'}

Confidence: 80%**Tokens:** 103 input + 122 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	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: The test verifies that the annotate function of OllamaProvider correctly handles call errors by checking if the last error message is 'boom'.

Why Needed: This test prevents a regression where the LLM provider fails to handle call errors and instead raises an exception.

Key Assertions:

- The annotation should have an error message indicating that the call was unsuccessful with the specified system prompt.
- The annotation should indicate that the last error occurred after 2 retries.
- The annotation should correctly identify the cause of the failure as 'boom'.

Confidence: 80%**Tokens:** 347 input + 126 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/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 potential bug where the provider incorrectly reports a non-existent issue, potentially leading to incorrect usage guidance or downstream errors.

Key Assertions:

- assert annotation.error == 'httpx not installed. Install with: pip install httpx'
- provider.annotate(test, 'def test_case(): assert True')
- test_case() should raise an ImportError when httpx is missing
- mock_import_error('httpx') should be called before annotating the test
- config.provider should be set to 'ollama' before creating the provider instance
- OllamaProvider(config) should create a provider instance with the correct config
- test should pass when httpx is installed correctly

Confidence: 80%

Tokens: 268 input + 182 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/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_runti
me_error_immediate_fail

1ms



AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: The test is likely to fail because the annotation of runtime errors by Ollama Provider is not implemented yet.

Key Assertions:

- {'name': 'OllamaProvider.annotate_runtime_error_immediate_fail returns correct result', 'description': 'The function should return a specific JSON object with certain key-value pairs.'}

Confidence: 80%

Tokens: 99 input + 95 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	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 that the annotate method correctly annotates a full flow with mocked HTTP responses.

Why Needed: Prevents authentication issues caused by unverified HTTP requests.

Key Assertions:

- check status of response after successful login
- validate token sent in response
- assert that no exception is raised during annotation process

Confidence: 80%

Tokens: 414 input + 80 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/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 LiteLLMProvider overrides the prompt when provided with a custom prompt.

Why Needed: To ensure that the LiteLLM provider uses the custom prompt when it is provided, rather than using the default one.

Key Assertions:

- The annotation returned by `_annotate_internal` method of LiteLLMProvider should not contain any error.
- The content of the captured messages should match 'CUSTOM PROMPT'.
- The key assertion 'a' in the captured message should be present and have a valid type.

Confidence: 80%

Tokens: 373 input + 121 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/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: Test LiteLLM provider annotates with token usage.**Why Needed:** Prevents regression in token usage calculation when annotating responses.**Key Assertions:**

- The total tokens should be equal to the sum of prompt and completion tokens.
- The prompt tokens should not exceed 100.
- The completion tokens should not exceed 50.
- The total tokens should be greater than or equal to 150.
- The token usage is not None.
- The token usage is correctly calculated based on the provided data.

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: Ollama provider makes correct API call for successful response.**Why Needed:** Prevents bug where OllamaProvider returns incorrect or incomplete responses.**Key Assertions:**

- The response from the Ollama model is as expected and matches the provided prompt.
- The URL of the generated text is correctly set to the specified host and port.
- The JSON payload sent by the Ollama provider includes the required 'model', 'prompt', 'system', and 'stream' fields.
- The timeout value is correctly set to 60 seconds as expected.
- The response from the Ollama model does not exceed the maximum allowed length of 2048 characters.

Confidence: 80%**Tokens:** 470 input + 154 output = 624 total

COVERAGE

src/pytest_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: Test that the Ollama provider uses the default model when not specified.**Why Needed:** This test prevents a regression where the model is not set to 'llama3.2' even if it's not provided in the configuration.**Key Assertions:**

- The captured response from the Ollama provider contains the expected model.
- The value of `model` in the captured response matches the default model ('llama3.2').
- The absence of a specified model in the configuration does not prevent the provider from using the default model.

Confidence: 80%**Tokens:** 344 input + 129 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/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: Test OLLAMA Provider: Check Availability Failure**Why Needed:** The test checks if the OLLAMA provider returns False when the server is unavailable.**Key Assertions:**

- {'name': 'Provider is not set up to return a response', 'expected': 'False', 'actual': 'True'}

Confidence: 80%**Tokens:** 183 input + 83 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	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:** To test that the Ollama provider returns False for non-200 status codes.**Key Assertions:**

- {'name': 'provider._check_availability()' is False, 'expected_value': False}

Confidence: 80%**Tokens:** 197 input + 87 output = 284 total

COVERAGE

src/pytest_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: Verify that the Ollama provider checks availability successfully by returning a 200 status code for the /api/tags endpoint.

Why Needed: This test prevents a potential bug where the provider returns an error or exception when checking availability, causing the application to fail.

Key Assertions:

- The '/api/tags' URL is present in the request.
- The response from the server has a status code of 200.
- The provider's check_availability method does not raise any exceptions when called with a valid configuration.
- The provider's check_availability method returns True for a successful check.
- The provider's check_availability method does not return False for an unsuccessful check.

Confidence: 80%

Tokens: 296 input + 155 output = 451 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_llm_providers.py**Why Needed:** To ensure 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 + 75 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	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 ensure that the `get_max_context_tokens` method returns a fallback value when an error occurs.**Key Assertions:**

- {'name': 'Expected `get_max_context_tokens` to return a fallback value', 'description': 'When an error occurs, `get_max_context_tokens` should return a fallback value'}
- {'name': 'Fallback value is returned for non-LLM errors', 'description': 'The fallback value should be returned for all types of errors (not just LLM errors)'}
- {'name': 'Fallback value is returned for LLM errors', 'description': 'The fallback value should be returned when an error occurs with the LLM provider'}

Confidence: 80%**Tokens:** 101 input + 162 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/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



AI ASSESSMENT

Scenario: tests/test_llm_providers.py**Why Needed:** To ensure that the `get_max_context_tokens` method returns the correct maximum context tokens for a given model info.**Key Assertions:**

- {'name': 'max_context_tokens', 'expected_value': 512, 'actual_value': 0}
- {'name': 'context_token_count', 'expected_value': 8, 'actual_value': 4}

Confidence: 80%**Tokens:** 99 input + 112 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	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

1ms



AI ASSESSMENT

Scenario: Tests for OLLAMA provider**Why Needed:** To ensure the correct number of context tokens is returned from the parameters**Key Assertions:**

- {'name': 'Context token count', 'expected_value': 8, 'actual_value': 5}
- {'name': 'Token type distribution', 'expected_value': "{'B': 55.0, 'I': 30.0, 'O': 15.0}", 'actual_value': "{'B': 55.0, 'I': 31.0, 'O': 14.0}"}

Confidence: 80%**Tokens:** 97 input + 135 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/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



AI ASSESSMENT

Scenario: Tests for LLM providers**Why Needed:** To ensure the LLM provider returns a valid response when the maximum context tokens is not 200.**Key Assertions:**

- {'name': 'Response status code should be 400', 'expected_value': 400, 'actual_value': 0}
- {'name': 'Response body should contain an error message', 'expected_value': 'Error: max_context_tokens must be at least 200', 'actual_value': ''}

Confidence: 80%**Tokens:** 101 input + 123 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/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



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 an instance of OllamaProvider', 'expected_value': 'True'}
- {'name': 'provider is configured with a valid provider name', 'expected_value': 'ollama'}

Confidence: 80%

Tokens: 123 input + 112 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	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



AI ASSESSMENT

Scenario: Ollama provider reports invalid JSON responses**Why Needed:** To ensure the Ollama provider correctly handles and reports invalid JSON responses in its tests.**Key Assertions:**

- {'name': 'annotation.error', 'value': 'Failed to parse LLM response as JSON'}

Confidence: 80%**Tokens:** 138 input + 76 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	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: test_parse_response_invalid_key_assertions

Why Needed: to test Ollama provider's behavior when receiving invalid key_assertions payloads

Key Assertions:

- {'message': 'Invalid response: key_assertions must be a list'}

Confidence: 80%

Tokens: 174 input + 95 output = 269 total

COVERAGE

src/pytest_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 test that the LLM provider correctly extracts JSON from markdown code fences.

Key Assertions:

- {'name': 'Expected JSON format', 'value': '{"scenario": "tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_code_fence", "why_needed": "To test that the LLM provider correctly extracts JSON from markdown code fences."}'}

Confidence: 80%

Tokens: 127 input + 124 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/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



AI ASSESSMENT

Scenario:

tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_plain_fence

Why Needed: To test the functionality of extracting JSON from plain markdown fences (no language).**Key Assertions:**

- {'name': 'Expected response is a dictionary', 'description': 'The extracted JSON should be in a dictionary format.'}
- {'name': 'Expected keys are correct', 'description': 'The keys in the extracted JSON should match the expected ones.'}

Confidence: 80%**Tokens:** 128 input + 122 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/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 Ollama provider parses valid JSON responses.**Why Needed:** Prevents bugs in the LLM providers by ensuring correct parsing of 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 + 102 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/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



AI ASSESSMENT

Scenario: Verify water-fill algorithm satisfies smaller files first.

Why Needed: Prevents regression where larger files are prioritized over smaller ones, potentially leading to incorrect token distribution.

Key Assertions:

- The total allocated tokens for small.py should be exactly 10.
- The total allocated tokens for large.py should be between 30 and 45 (inclusive).
- The allocation of tokens to small.py is sufficient to cover its content, leaving some wiggle room for overhead estimation changes.

Confidence: 80%

Tokens: 396 input + 114 output = 510 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_llm_utils.py::test_distribute_token_budget_empty**Why Needed:** Verify that the function handles empty input or no budget correctly.**Key Assertions:**

- {'assertion': {'message': 'Expected an empty dictionary to be returned when {} is passed with 100 as the token budget.'}, 'expected_result': {}}
- {'assertion': {'message': "Expected an empty dictionary to be returned when {'f1': 'c'} is passed with 0 as the token budget."}, 'expected_result': {}}

Confidence: 80%**Tokens:** 115 input + 131 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/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



AI ASSESSMENT

Scenario: Verify fair sharing when neither fits.

Why Needed: Prevents bug where one model gets significantly more tokens than the other, leading to unfair distribution of token budget.

Key Assertions:

- The expected range for allocations of `l1.py` is between 35 and 50 tokens.
- The expected range for allocations of `l2.py` is also between 35 and 50 tokens.
- The absolute difference in allocations should be less than or equal to 1 token.

Confidence: 80%

Tokens: 327 input + 116 output = 443 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_llm_utils.py::test_distribute_token_budget_max_files**Why Needed:** To ensure the LLM Utils library handles token budget allocation correctly when dealing with a large number of files.**Key Assertions:**

- {'name': 'assert len(allocations) == 3', 'expected_result': 3, 'actual_result': 0}

Confidence: 80%**Tokens:** 133 input + 94 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/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



AI ASSESSMENT

Scenario: Verify that all files receive sufficient token allocation when the budget is sufficient.

Why Needed: This test prevents a potential bug where files with larger content are not allocated enough tokens, leading to incomplete or corrupted code.

Key Assertions:

- The number of allocations for each file matches the required amount (10 tokens).
- Each allocation is within the estimated overhead range (6-16 tokens per file).
- All files receive full content as expected.

Confidence: 80%

Tokens: 332 input + 108 output = 440 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Verify the minimum token estimation for an empty string.

Why Needed: This test prevents a potential bug where the function does not return a valid estimate for an empty string, potentially leading to incorrect results or errors in downstream processing.

Key Assertions:

- assert estimate_tokens('') == 1
- # Expected minimum token estimation of 1

Confidence: 80%

Tokens: 217 input + 85 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/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



AI ASSESSMENT

Scenario: Test the `CoverageEntry` class to ensure it correctly serializes its attributes.

Why Needed: This test prevents a bug where the `CoverageEntry` object is not properly serialized, potentially causing issues with data transfer or storage.

Key Assertions:

- The 'file_path' attribute of the `CoverageEntry` object should match the expected value.
- The 'line_ranges' attribute of the `CoverageEntry` object should match the expected format.
- The 'line_count' attribute of the `CoverageEntry` object should be equal to the expected value.

Confidence: 80%

Tokens: 255 input + 130 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/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



AI ASSESSMENT

Scenario: Test that `CoverageEntry.to_dict()` correctly serializes a CoverageEntry object.

Why Needed: This test prevents a potential bug where the serialized data does not match the expected format, potentially causing issues with downstream processing or debugging.

Key Assertions:

- The 'file_path' key in the serialized dictionary should be equal to 'src/foo.py'.
- The 'line_ranges' key in the serialized dictionary should be equal to '1-3, 5, 10-15'.
- The 'line_count' key in the serialized dictionary should be equal to 10.
- The 'file_path' value in the expected dictionary should match the actual value in the test data.
- The 'line_ranges' value in the expected dictionary should match the actual value in the test data.
- The 'line_count' value in the expected dictionary should match the actual value in the test data.
- All keys and values in the serialized dictionary should be present in the expected dictionary.
- Any missing keys or values in the serialized dictionary should raise an AssertionError.

Confidence: 80%

Tokens: 255 input + 236 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/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



AI ASSESSMENT

Scenario: Tests CoverageEntry to_dict method.

Why Needed: This test prevents a potential bug where the coverage entry serialization fails or is incorrect when dealing with complex line ranges.

Key Assertions:

- The 'file_path' key in the d dictionary should match the expected value.
- The 'line_ranges' key in the d dictionary should contain the correct range values.
- The 'line_count' key in the d dictionary should be equal to the expected value.

Confidence: 80%

Tokens: 255 input + 109 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/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



AI ASSESSMENT

Scenario: An empty annotation should be created with default values.

Why Needed: This test prevents a potential bug where an empty annotation does not have any default values.

Key Assertions:

- annotation.scenario == "" (empty string)
- annotation.why_needed == "Empty annotation does not have any default values."
- annotation.key_assertions == [] (no key assertions are performed in this test)
- assert annotation.confidence is None (checks if confidence is set to None)
- assert annotation.error is None (checks if error is set to None)

Confidence: 80%

Tokens: 212 input + 131 output = 343 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: The test verifies that the `to_dict` method of `LlmAnnotation` returns a dictionary with all required fields.

Why Needed: This test prevents a potential bug where the minimal annotation is missing some required fields.

Key Assertions:

- Required keys: 'scenario', 'why_needed', and 'confidence'
- Optional key: 'key_assertions' (does not need to be present)
- Missing field: 'confidence' (should be excluded when None)

Confidence: 80%

Tokens: 230 input + 112 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	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



AI ASSESSMENT

Scenario: Test that the full annotation is included in the dictionary.

Why Needed: Prevents a potential bug where only some fields are included in the output.

Key Assertions:

- Asserts that the 'scenario' field is present and matches the expected value.
- Asserts that the 'confidence' field is present and matches the expected value.
- Asserts that the 'context_summary' field has the correct mode and byte count.

Confidence: 80%

Tokens: 284 input + 104 output = 388 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test Default Report

Why Needed: Prevents a potential bug where the default report is missing required schema version and empty test lists.

Key Assertions:

- assert d['schema_version'] == SCHEMA_VERSION
- assert d['tests'] == []
- assert 'warnings' not in d
- assert 'collection_errors' not in d

Confidence: 80%**Tokens:** 231 input + 93 output = 324 total

COVERAGE

src/pytest_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 test with collection errors to ensure it includes them.**Why Needed:** This test prevents a regression where the report does not include all collection errors.**Key Assertions:**

- The report should contain at least one collection error.
- Each collection error should have a unique nodeid.
- The first collection error should be for the specified nodeid 'test_bad.py'.

Confidence: 80%**Tokens:** 237 input + 94 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/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



AI ASSESSMENT

Scenario: TestReportRoot::test_report_with_warnings**Why Needed:** To test that the ReportRoot class correctly identifies warnings in a report.**Key Assertions:**

- {'name': "Expected length of 'warnings' key to be 1", 'description': 'The number of warnings in the report should be exactly one.'}
- {'name': "Expected value of 'code' in first warning to match 'W001'", 'description': "The code of the first warning should match 'W001'."}

Confidence: 80%**Tokens:** 144 input + 126 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/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



AI ASSESSMENT

Scenario: Test: tests/test_models.py::TestReportRoot::test_tests_sorted_by_nodeid**Why Needed:** This test ensures that the output of ReportRoot is sorted by nodeid.**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 + 123 output = 338 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: The `to_dict` method of the `ReportWarning` class is being tested.

Why Needed: This test is needed because it checks if the `detail` key is included in the dictionary returned by the `to_dict` method.

Key Assertions:

- {'name': 'assert detail is correct', 'expected_value': '/path/to/file', 'actual_value': "assert d['detail'] == '/path/to/file'")}

Confidence: 80%

Tokens: 131 input + 128 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/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)

AI ASSESSMENT

Scenario: Test to dictionary without detail should exclude it.

Why Needed: This test prevents a warning that could indicate a missing or misleading detail in the report.

Key Assertions:

- The 'detail' key is present and contains the actual detailed message.
- The 'code' key matches the expected code.
- The 'message' key matches the expected message.
- The 'warning_type' key (currently set to 'ReportWarning') is not included in the dictionary.

Confidence: 80%

Tokens: 223 input + 112 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/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



AI ASSESSMENT

Scenario: Verify that RunMeta has aggregation fields.

Why Needed: Prevents regression in aggregation feature where run_id, run_group_id and run_count are expected to be present.

Key Assertions:

- The 'run_id' key is present in the output dictionary.
- The 'run_group_id' key is present in the output dictionary.
- The 'is_aggregated' key is True.
- The 'aggregation_policy' key is set to 'merge'.
- The 'run_count' key is equal to 3.
- The length of 'source_reports' list is 2.

Confidence: 80%

Tokens: 343 input + 140 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/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



AI ASSESSMENT

Scenario: Test LLM fields are excluded when annotations not enabled.**Why Needed:** Prevents regression where LLM fields are included despite annotation being disabled.**Key Assertions:**

- The 'llm_annotations_enabled' key is present in the data.
- The 'llm_provider' key is present in the data.
- The 'llm_model' key is present in the data.

Confidence: 80%**Tokens:** 232 input + 94 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/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



AI ASSESSMENT

Scenario: Test LLM traceability fields are included when enabled.**Why Needed:** Prevents regression in case llm_annotations_enabled is disabled.**Key Assertions:**

- assert data['llm_annotations_enabled'] is True
- assert data['llm_provider'] == 'ollama'
- assert data['llm_model'] == 'llama3.2:1b'
- assert data['llm_context_mode'] == 'complete'
- assert data['llm_annotations_count'] == 10
- assert data['llm_annotations_errors'] == 2

Confidence: 80%**Tokens:** 327 input + 135 output = 462 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_models.py::TestRunMeta::test_non_aggregated_excludes_source_reports

Why Needed: To ensure that non-aggregated reports do not include source_reports.

Key Assertions:

- {'name': 'Non-aggregated report should not include source_reports', 'expected_result': 'False'}

Confidence: 80%

Tokens: 130 input + 81 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/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



AI ASSESSMENT

Scenario: Test RunMeta to dict with all optional fields.**Why Needed:** Prevents regression in case of missing or outdated plugin version, which could lead to incorrect aggregation policy.**Key Assertions:**

- The 'git_sha' field is set to 'abc1234'.
- The 'git_dirty' field is set to True.
- The 'repo_version' field is set to '1.0.0'.
- The 'repo_git_sha' field is set to 'abc1234'.
- The 'repo_git_dirty' field is set to True.
- The 'plugin_git_sha' field is set to 'def5678'.
- The 'plugin_git_dirty' field is set to False.
- The 'config_hash' field is set to 'def5678'.
- The length of the 'source_reports' list is 1.

Confidence: 80%**Tokens:** 483 input + 198 output = 681 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: TestRunMeta::test_run_status_fields verifies that the RunMeta class includes run status fields.

Why Needed: This test prevents a regression where the RunMeta class does not include all necessary run status fields, potentially causing incorrect interpretation of the results.

Key Assertions:

- The 'exit_code' field should be set to 1.
- The 'interrupted' field should be set to True.
- The 'collect_only' field should be set to True.
- The 'collected_count' field should match the number of runs collected (10).
- The 'selected_count' field should match the number of selected runs (8).
- The 'deselected_count' field should match the number of deselected runs (2).

Confidence: 80%

Tokens: 285 input + 173 output = 458 total

COVERAGE

src/pytest_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)

AI ASSESSMENT

Scenario: tests/test_models.py::TestSchemaVersion::test_schema_version_format

Why Needed: This test ensures that the schema version is formatted correctly as semver.

Key Assertions:

- {'name': 'schema_version should be semver format', 'expected_output': {'format': 'semver'}, 'actual_output': {'format': 'string'}}}

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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestSchemaVersion::test_schema_version_in_repo_rt_root

1ms



AI ASSESSMENT

Scenario: tests/test_models.py::TestSchemaVersion::test_schema_version_in_report_root

Why Needed: This test ensures that the ReportRoot class includes the schema version in its JSON representation.

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 + 106 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/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



AI ASSESSMENT

Scenario: The test verifies that a CoverageEntry object can be serialized correctly into a dictionary.

Why Needed: This test prevents a potential bug where the coverage data is not properly encoded in the JSON representation.

Key Assertions:

- The 'file_path' key should contain the expected value.
- The 'line_ranges' key should contain the expected string representation.
- The 'line_count' key should contain the expected integer value.
- Each assertion should be true for a CoverageEntry object.
- The dictionary structure should match the expected format.
- Any additional keys or values in the dictionary should not affect the test result.

Confidence: 80%

Tokens: 256 input + 143 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/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



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 dictionary should contain 'scenario' key
- The dictionary should contain 'why_needed' key
- The dictionary should contain 'key_assertions' key
- The dictionary should not contain 'confidence' key when it's None

Confidence: 80%

Tokens: 229 input + 114 output = 343 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: TestSourceReport::test_to_dict_with_run_id**Why Needed:** To ensure SourceReport objects have a 'run_id' attribute in their dictionary representation.**Key Assertions:**

- {'name': "Expected value for 'run_id'", 'value': 'run-1', 'type': 'string'}

Confidence: 80%**Tokens:** 134 input + 77 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/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



AI ASSESSMENT

Scenario: Test `test_to_dict` verifies that a CoverageEntry object is correctly serialized to a dictionary.

Why Needed: This test prevents regression in coverage entry serialization.

Key Assertions:

- The 'file_path' key in the dictionary should match the actual file path of the CoverageEntry.
- The 'line_ranges' key in the dictionary should match the expected line ranges.
- The 'line_count' key in the dictionary should match the actual number of lines in the CoverageEntry.
- All values in the dictionary should be strings, as they represent file paths, line ranges, and line counts.

Confidence: 80%

Tokens: 254 input + 138 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/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



AI ASSESSMENT

Scenario: The `TestTestCaseResult` class should be able to create a minimal result with the required fields.

Why Needed: This test prevents regression in cases where only the 'nodeid', 'outcome', and 'duration' are known.

Key Assertions:

- The value of `d['nodeid']` is equal to ``test_foo.py::test_bar``.
- The value of `d['outcome']` is equal to 'passed'.
- The value of `d['duration']` is equal to '0.0'.
- The value of `d['phase']` is equal to 'call'.

Confidence: 80%

Tokens: 244 input + 150 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/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)

AI ASSESSMENT

Scenario: Test verifies that the `TestCaseResult` includes a coverage list.

Why Needed: This test prevents regression in cases where the coverage is not included in the result.

Key Assertions:

- The coverage list should contain exactly one entry.
- The file path of the first coverage entry should match 'src/foo.py'.
- All line ranges and counts in the coverage entry should be present.

Confidence: 80%

Tokens: 256 input + 96 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/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



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 result.

Key Assertions:

- {'name': 'assert llm_opt_out is True', 'expected_value': True, 'message': 'Expected llm_opt_out to be True'}

Confidence: 80%

Tokens: 145 input + 94 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/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



AI ASSESSMENT

Scenario: tests/test_models.py::TestTestCaseResult::test_result_with_rerun**Why Needed:** The test result should include rerun fields.**Key Assertions:**

- {'name': 'rerun_count', 'expected_value': 2, 'message': 'Expected rerun count to be 2'}
- {'name': 'final_outcome', 'expected_value': 'passed', 'message': "Expected final outcome to be 'passed'"}

Confidence: 80%**Tokens:** 162 input + 117 output = 279 total

COVERAGE

src/pytest_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 is needed because it checks if the result of a test case does not include fields that indicate reruns.

Key Assertions:

- {'name': 'result should be an empty dictionary', 'description': 'The result of the test case should be an empty dictionary.'}
- {'name': 'rerun_count should be None', 'description': 'The rerun count should be None.'}
- {'name': 'final_outcome should be passed', 'description': "The final outcome of the test case should be 'passed'."}

Confidence: 80%

Tokens: 152 input + 161 output = 313 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: test_to_dict_with_all_optional_fields verifies that the `to_dict` method includes all optional fields when set.

Why Needed: This test prevents regression by ensuring that the `to_dict` method handles optional parameters correctly.

Key Assertions:

- assert result['param_id'] == 'a-b-c',
- assert result['param_summary'] == 'a=1, b=2, c=3',
- assert result['captured_std dout'] == 'std dout content',
- assert result['captured_stderr'] == 'stderr content',
- assert result['requirements'] == ['REQ-100'],
- assert result['llm_opt_out'] is True,
- assert result['llm_context_override'] == 'complete',
- assert len(result['coverage']) == 1,
- assert result['llm_annotation']['scenario'] == 'Tests foo'

Confidence: 80%

Tokens: 454 input + 203 output = 657 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test to_dict includes artifacts when set.**Why Needed:** This test prevents a regression where the report is not properly formatted with artifacts.**Key Assertions:**

- The length of `result['artifacts']` should be 2.
- The path of `result['artifacts'][0]` should be `report.html`.
- All artifact entries in `result['artifacts']` should have a 'path' key.

Confidence: 80%**Tokens:** 264 input + 105 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/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



AI ASSESSMENT

Scenario: Test to_dict includes collection_errors when set.

Why Needed: Prevents a potential bug where the test reports collection errors without including them in the dictionary.

Key Assertions:

- The length of `result['collection_errors']` is 1 and it contains only one item with key `nodeid` equal to `broken_test.py`.
- The value of `result['collection_errors'][0]` has a valid node id `broken_test.py`.
- The value of `result['collection_errors'][0]['message']` is `SyntaxError` which is the expected error message for `CollectionError` type.

Confidence: 80%

Tokens: 243 input + 144 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/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



AI ASSESSMENT

Scenario: Test to_dict includes custom_metadata when set.**Why Needed:** Prevents a bug where the custom metadata is not included in the report.**Key Assertions:**

- The 'custom_metadata' key should be present in the result dictionary.
- The 'custom_metadata' key should contain the expected project, environment, and build_number values.
- The 'build_number' value should match the provided value of 123.
- The custom metadata should override any default values set by the ReportRoot class.
- The custom metadata should be included in the report even if it's empty or None.
- The custom metadata should not be overwritten by other metadata attributes (e.g., 'project' and 'environment').
- The custom metadata should be preserved across different test runs with the same configuration.

Confidence: 80%**Tokens:** 264 input + 179 output = 443 total

COVERAGE

src/pytest_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: HMAC signature is included in the report for security purposes.

Key Assertions:

- {'expected_value': 'signature123', 'actual_value': 'None'}

Confidence: 80%

Tokens: 128 input + 76 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/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



AI ASSESSMENT

Scenario: Test to_dict includes sha256 when set.

Why Needed: Because the ReportRoot class has a SHA-256 hash stored in its instance variable.

Key Assertions:

- {'name': 'Expected result', 'expected_value': 'abcdef1234567890'}

Confidence: 80%

Tokens: 131 input + 97 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/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 does not verify that the report includes source coverage, potentially leading to incorrect reporting or missing important information.**Key Assertions:**

- The 'source_coverage' key in the result dictionary should contain exactly one SourceCoverageEntry.
- The 'file_path' of the first SourceCoverageEntry in the 'source_coverage' list should match the specified file path.
- All other keys and values in the 'source_coverage' list should be present and have the expected values.

Confidence: 80%**Tokens:** 282 input + 130 output = 412 total

COVERAGE

src/pytest_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: The test is necessary because it checks the behavior of `report.to_dict()` when warnings are present.

Key Assertions:

- {'name': "len(result['warnings']) == 1", 'description': "The length of the 'warnings' key in the result dictionary should be 1.", 'expected_value': 1, 'message': 'Expected length of warnings to be 1.'}
- {'name': "result['warnings'][0]['code'] == 'W001'", 'description': "The code of the first warning in the 'warnings' list should be 'W001'.", 'expected_value': 'W001', 'message': "Expected code of the first warning to be 'W001'."}

Confidence: 80%

Tokens: 151 input + 196 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	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: The test should include the coverage_total_percent key in the result dictionary.

Why Needed: Because of the following reason: The `to_dict()` method is expected to return a dictionary with all the assertions, including the `coverage_total_percent` value. Without it, the assertion would fail because the key is missing.

Key Assertions:

- {'description': 'The coverage_total_percent value should be equal to 85.5', 'expected_value': 85.5}

Confidence: 80%

Tokens: 153 input + 161 output = 314 total

COVERAGE

src/pytest_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 + 83 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/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



AI ASSESSMENT

Scenario: test_to_dict_with_all_optional_fields verifies that the `to_dict` method includes all optional fields when set.

Why Needed: This test prevents regression in case where `llm_opt_out=True` and `llm_context_override='complete'`.

Key Assertions:

- The 'param_id' field is present and matches 'a-b-c'.
- The 'param_summary' field is present and matches 'a=1, b=2, c=3'.
- The 'captured_st dout' field is present and matches 'st dout content'.
- The 'captured_st derr' field is present and matches 'st derr content'.
- All required parameters ('REQ-100') are included in the result.
- Optional fields (e.g. 'llm_opt_out', 'llm_context_override') are excluded from the result.
- The length of the coverage entry is 1, which is expected for a single test case.
- The 'llm_annotation' field contains the correct scenario name.

Confidence: 80%

Tokens: 454 input + 228 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/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



AI ASSESSMENT

Scenario:

tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_captured_stderr

Why Needed: Because the `to_dict` method includes captured stderr in the test case results.

Key Assertions:

- The captured stderr is correctly included in the result.

Confidence: 80%

Tokens: 149 input + 67 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/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: Captured stdout is necessary for accurate coverage analysis.

Key Assertions:

- {'name': 'result', 'expected_value': 'Debug output here'}

Confidence: 80%

Tokens: 149 input + 74 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	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



AI ASSESSMENT

Scenario:

tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_requirements

Why Needed: This test is necessary to ensure that the `to_dict` method includes requirements when set.

Key Assertions:

- {'assertion_type': 'contains', 'expected_value': ['REQ-001', 'REQ-002'], 'actual_value': ['REQ-001', 'REQ-002']}

Confidence: 80%

Tokens: 151 input + 102 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/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



AI ASSESSMENT

Scenario: Test the default exclude globs for LLM context.**Why Needed:** This test prevents a potential bug where the default exclude globs are not correctly set.**Key Assertions:**

- The function `llm_context_exclude_globs` returns a list of excluded files.
- The file `*.pyc` is included in the default exclude globs.
- The directory `__pycache__/*` is also included in the default exclude globs.
- The string `*secret*` is not included in the default exclude globs.
- The string `*password*` is correctly excluded as a file extension.
- The function `llm_context_exclude_globs` returns an empty list when no files or directories are specified.

Confidence: 80%**Tokens:** 222 input + 169 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/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



AI ASSESSMENT

Scenario: Verifies the default redact patterns used by the Config class.

Why Needed: Prevents a potential security vulnerability where sensitive information like passwords and tokens are not properly redacted.

Key Assertions:

- The `--password` pattern should match any string containing '--password' in its contents.
- The `--token` pattern should match any string containing '--token' in its contents.
- The `--api[_-]?key` pattern should match any string containing '--api[_-]?key' in its contents, where `_-` is a non-alphanumeric character and `key` is the actual key being redacted.

Confidence: 80%

Tokens: 228 input + 144 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/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



AI ASSESSMENT

Scenario: Test that default values are set correctly for the TestConfig class.

Why Needed: This test prevents a regression where the default values of the TestConfig class are not set correctly, potentially leading to unexpected behavior or errors.

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 + 205 output = 523 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_options.py::TestConfig::test_get_default_config**Why Needed:** To ensure that the default configuration is properly initialized and provides a 'provider' key with value 'none'.**Key Assertions:**

- {'name': 'cfg is an instance of Config', 'expected_type': 'Config'}
- {'name': "cfg.provider == 'none'", 'expected_value': 'none'}

Confidence: 80%**Tokens:** 104 input + 105 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	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



AI ASSESSMENT

Scenario: Test that the `is_llm_enabled` check returns False for providers without a specified LLM provider.

Why Needed: Prevents regression in case the `provider` is set to 'none' and the test relies on it returning True.

Key Assertions:

- The function should return False when the `provider` is not explicitly set or defaults to 'none'.
- The function should return True for providers with a specified LLM provider, such as 'ollama'.
- The function should return True for providers with a default LLM provider, such as 'litellm'.
- The function should not return False when the `provider` is set to 'gemini'.

Confidence: 80%

Tokens: 263 input + 163 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/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



AI ASSESSMENT

Scenario: test_validate_invalid_aggregate_policy

Why Needed: To test the validation of an invalid aggregation policy.

Key Assertions:

- {'description': "The error message should contain the string 'Invalid aggregate_policy ''', 'expected_value': "'Invalid aggregate_policy '''"}

Confidence: 80%

Tokens: 128 input + 73 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	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 test the validation of an invalid context mode.**Key Assertions:**

- {'name': 'Expected number of errors', 'value': 1, 'description': 'The function should return exactly one error message.'}
- {'name': 'Expected error message content', 'value': "Invalid llm_context_mode 'mega_max'", 'description': 'The error message should contain the invalid context mode string.'}

Confidence: 80%**Tokens:** 131 input + 126 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	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



AI ASSESSMENT

Scenario: test_validate_invalid_include_phase**Why Needed:** To ensure that the `include_phase` parameter is valid and raises an error when it's invalid.**Key Assertions:**

- {'assertion': 'The length of errors is 1', 'expected_result': 1}
- {'assertion': "The string 'Invalid include_phase ' is in errors[0]", 'expected_result': "Invalid include_phase 'lunch_break'"}

Confidence: 80%**Tokens:** 129 input + 112 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	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



AI ASSESSMENT

Scenario: tests/test_options.py::TestConfig::test_validate_invalid_provider**Why Needed:** To test the validation of an invalid provider.**Key Assertions:**

- {'message': "Invalid provider 'invalid_provider'", 'type': 'assertion'}

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



AI ASSESSMENT

Scenario: Test validation of numeric constraints for TestConfig.**Why Needed:** This test prevents regression where the default values for llm_context_bytes, llm_max_tests, llm_requests_per_minute, llm_timeout_seconds, and llm_max_retries are not validated correctly.**Key Assertions:**

- llm_context_bytes must be at least 1000
- llm_max_tests must be 0 (no limit) or positive
- llm_requests_per_minute must be at least 1
- llm_timeout_seconds must be at least 1
- llm_max_retries must be 0 or positive

Confidence: 80%**Tokens:** 329 input + 145 output = 474 total

COVERAGE

src/pytest_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



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 provided.

Key Assertions:

- {'name': 'errors should be empty', 'expected': [], 'actual': {'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 provided.', 'key_assertions': ['...']}}}

Confidence: 80%**Tokens:** 100 input + 131 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	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)

AI ASSESSMENT

Scenario: Test the ability to load aggregation options correctly.

Why Needed: This test prevents a bug where aggregation options are not loaded correctly, potentially causing issues with downstream processing.

Key Assertions:

- The `aggregate_dir` option is set to 'aggr_dir'.
- The `aggregate_policy` option is set to 'merge'.
- The `aggregate_run_id` option is set to 'run-123'.
- The `aggregate_group_id` option is set to 'group-abc'.

Confidence: 80%

Tokens: 295 input + 120 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/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)

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:**

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

Confidence: 80%**Tokens:** 138 input + 70 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	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:** This test prevents a bug where the LLM report generation fails with an error due to missing pyproject.toml file.**Key Assertions:**

- The function `load_config` should be able to handle the case when pyproject.toml is not present.
- The default value for `llm_max_retries` should be used in this scenario.
- The test should fail with a meaningful error message indicating that the LLM report generation failed due to missing pyproject.toml file.

Confidence: 80%**Tokens:** 413 input + 130 output = 543 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_options.py::TestLoadConfig::test_load_coverage_source**Why Needed:** To test the coverage source option.**Key Assertions:**

- {'expected': 'cov_dir', 'actual': 'None'}

Confidence: 80%**Tokens:** 126 input + 64 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	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 functionality of loading configuration with no options set.**Key Assertions:**

- {'name': 'cfg.provider == "none"', 'expected_value': 'None'}
- {'name': 'cfg.report_html is None', 'expected_value': 'None'}

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/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: Load configuration from CLI overrides in PyProject

Why Needed: To ensure that CLI options override PyProject settings.

Key Assertions:

- {'name': 'CLI options override PyProject.toml', 'expected_value': {'override': True}}
- {'name': 'PyProject.toml values not overridden by CLI options', 'expected_value': {}}

Confidence: 80%

Tokens: 134 input + 95 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	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: Load configuration from CLI provider override

Why Needed: To test that CLI provider option overrides pyproject.toml.

Key Assertions:

- {'name': 'CLI provider option overrides pyproject.toml', 'expected_value': 'pyproject.toml'}

Confidence: 80%

Tokens: 130 input + 71 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	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)

AI ASSESSMENT

Scenario: tests/test_options.py::TestLoadConfig::test_load_from_cli_retries**Why Needed:** To test the functionality of loading retries from CLI.**Key Assertions:**

- {'expected': {'llm_max_retries': 2}, 'actual': {'llm_max_retries': 2}}

Confidence: 80%**Tokens:** 130 input + 81 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/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



AI ASSESSMENT

Scenario: Test loading values from pyproject.toml.**Why Needed:** To test the functionality of the `load_config` method in the Options class.**Key Assertions:**

- {'name': 'pyproject.toml file exists', 'expected': 'True', 'actual': 'True'}
- {'name': 'pyproject.toml contents are correct', 'expected': "{...}", 'actual': {'scenario': '...', 'why_needed': '...', 'key_assertions': ['...']}}

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	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 loads token optimization options from CLI and verifies the expected configuration.

Why Needed: This test prevents a potential bug where the `llm_prompt_tier` option is set to 'minimal' but the `batch_parametrized_tests` option is still enabled, leading to unexpected behavior in the test.

Key Assertions:

- The `prompt_tier` of the configuration should be 'minimal'.
- The `batch_parametrized_tests` flag should be False.
- The `context_compression` of the configuration should be 'none'.

Confidence: 80%

Tokens: 264 input + 129 output = 393 total

COVERAGE

src/pytest_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)

PASSED

tests/test_options_coverage.py::TestCliOverrides::test_cli_dependency_snapshot

5ms

3

AI ASSESSMENT

Scenario: Testing the `test_cli_dependency_snapshot` test function to verify that it correctly sets the `llm_dependency_snapshot` option to 'deps.json' when a CLI override is used.

Why Needed: This test prevents a potential regression where the `llm_dependency_snapshot` option is not set to 'deps.json' when a CLI override is applied, which could lead to unexpected behavior or errors in dependency snapshot reporting.

Key Assertions:

- The `llm_dependency_snapshot` option is correctly set to 'deps.json' after applying a CLI override.
- The `report_dependency_snapshot` configuration value matches the expected value of 'deps.json'.
- A mock configuration object is created with the correct `llm_dependency_snapshot` option value.
- The `load_config` function successfully loads the mock configuration object.
- The `cfg.report_dependency_snapshot` attribute is set to the expected value of 'deps.json' after applying a CLI override.
- A critical check is performed: the `llm_dependency_snapshot` option is present and has the correct value in the configuration.
- A test case is executed with mock configuration data, ensuring that the `llm_dependency_snapshot` option is correctly set.

Confidence: 80%**Tokens:** 213 input + 261 output = 474 total

COVERAGE

src/pytest_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)

PASSED

tests/test_options_coverage.py::TestCliOverrides::test_cli_evidence_bundle 6ms ⚡ 3

AI ASSESSMENT

Scenario: Verify that the `report_evidence_bundle` option is correctly set to 'bundle.zip' when CLI override is enabled.

Why Needed: This test prevents a potential bug where the `report_evidence_bundle` option is not updated correctly when CLI overrides are applied.

Key Assertions:

- The value of `cfg.report_evidence_bundle` should be set to 'bundle.zip'.
- The value of `mock.option.llm_evidence_bundle` should match 'bundle.zip'.

Confidence: 80%

Tokens: 217 input + 114 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/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_js 6ms 3

AI ASSESSMENT

Scenario: Verify that the test CLI overrides the default report format to 'output.json'.**Why Needed:** This test prevents a potential bug where the default report format is not set correctly.**Key Assertions:**

- The `llm_report_json` option is set to 'output.json' in the mock configuration.
- The value of `report_json` in the loaded configuration matches 'output.json'.
- The `llm_report_json` option is present and has a value of 'output.json' in the loaded configuration.

Confidence: 80%**Tokens:** 212 input + 122 output = 334 total

COVERAGE

src/pytest_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 5ms 3

AI ASSESSMENT

Scenario: Testing the 'test_cli_report_pdf' test function to verify that it sets the 'report_pdf' option correctly.

Why Needed: This test prevents a bug where the 'report_pdf' option is not set correctly for CLI overrides.

Key Assertions:

- The 'report_pdf' option should be set to 'output.pdf'.
- Mocking the config object with a mock configuration and verifying it sets the 'report_pdf' option correctly.
- Verifying that the assert statement passes when the 'report_pdf' option is set correctly.
- Testing the error case where the 'report_pdf' option is not set at all.
- Mocking the config object without setting the 'report_pdf' option and verifying it does not affect the test.
- Verifying that the mock configuration overrides any default value for the 'report_pdf' option.
- Testing the case where the 'llm_report_pdf' option is also overridden by a custom value.

Confidence: 80%

Tokens: 212 input + 211 output = 423 total

COVERAGE

src/pytest_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 validation of invalid token output format

Why Needed: To ensure that the token output format is always valid and consistent with the expected format.

Key Assertions:

- {'assertion': "litellm_token_output_format should be one of ['json', 'xml']", 'description': 'The token output format should be either json or xml.'}

Confidence: 80%

Tokens: 130 input + 91 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	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: Because the token refresh interval is set to 30 seconds, which is too short and may cause issues with authentication.

Key Assertions:

- {'description': 'The token refresh interval must be at least 60 seconds', 'expected_value': 60, 'actual_value': 30}

Confidence: 80%

Tokens: 146 input + 93 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	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_llm_config

1ms



AI ASSESSMENT

Scenario: test_validate_valid_llm_config**Why Needed:** To ensure that the LiteLLM configuration is valid and does not raise any validation errors.**Key Assertions:**

- {'assertion_type': 'is_not_empty', 'expected_value': 'litellm_token_output_format', 'actual_value': 'text'}
- {'assertion_type': 'not_equal_to', 'expected_value': 3600, 'actual_value': 3600}

Confidence: 80%**Tokens:** 142 input + 118 output = 260 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_aggregate_include_history

Why Needed: To ensure that the aggregate_include_history feature is properly loaded from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'include_history = True\\naggregate_include_history = True', 'actual': 'include_history = True\\naggregate_include_history = True'}

Confidence: 80%

Tokens: 118 input + 108 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	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 1ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_aggregate_policy_from_pyproject

Why Needed: To ensure that the aggregate policy is loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists and has an aggregate_policy section', 'expected': 'pyproject.toml should exist and have an aggregate_policy section'}
- {'name': 'aggregate_policy is present in pyproject.toml', 'expected': 'aggregate_policy should be present in pyproject.toml'}

Confidence: 80%

Tokens: 121 input + 135 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_all_config_keys_combined

2ms



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 pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists and is not empty', 'expected': {'status': 0, 'message': ''}, 'actual': {'status': 0, 'message': ''}}
- {'name': 'pyproject.toml file has all config keys', 'expected': {'status': 1, 'message': 'Missing required config key'}, 'actual': {}}

Confidence: 80%

Tokens: 120 input + 150 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/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 'cache_dir' key", 'expected_value': 'True'}
- {'name': "pyproject.toml contains a valid 'cache_dir' value", 'expected_value': "The 'cache_dir' value in the pyproject.toml file is correct."}

Confidence: 80%**Tokens:** 113 input + 132 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/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/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_cache_ttl_seconds

Why Needed: To ensure that the cache TTL seconds is loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml file exists and has ttl_seconds setting', 'value': 'True'}
- {'name': 'pyproject.toml file content contains ttl_seconds setting', 'value': {'ttl_seconds': 3600}}

Confidence: 80%

Tokens: 116 input + 120 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	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` feature flag is correctly set when loading a PyProject file with an empty 'build' section.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'empty build section', 'actual': 'not empty build section'}

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_output_max_chars

1ms



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_output_max_chars

Why Needed: To ensure that the `capture_output_max_chars` option in `pyproject.toml` is correctly loaded and used to set the maximum number of characters for capturing output.

Key Assertions:

- {'name': 'Expected value for capture_output_max_chars', 'value': 100, 'expected_type': 'int'}
- {'name': 'Expected error message for invalid capture_output_max_chars value', 'value': 'Invalid capture_output_max_chars value. It should be an integer between 1 and 999.', 'expected_type': 'str'}

Confidence: 80%**Tokens:** 119 input + 160 output = 279 total

COVERAGE

src/pytest_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 functionality in Pytest is working correctly.

Key Assertions:

- {'name': 'Context bytes are loaded from pyproject.toml', 'expected_value': 'context_bytes', 'actual_value': 'pyproject.toml'}
- {'name': 'Context bytes are not loaded from other files', 'expected_value': 'other_files', 'actual_value': 'None'}

Confidence: 80%

Tokens: 113 input + 128 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	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 the PyProject is properly loaded and excluded from the test coverage.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'exclude globs from pyproject.toml', 'actual': 'include globs from pyproject.toml'}

Confidence: 80%

Tokens: 119 input + 110 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	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 properly set in the PyProject.toml file.

Key Assertions:

- {'name': 'Context file limit is correctly set to 1000', 'expected_value': 1000, 'actual_value': '1234'}
- {'name': 'Context file limit is not exceeded by default settings', 'expected_value': 1000}

Confidence: 80%

Tokens: 116 input + 125 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	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.py`**Why Needed:** To ensure that the `context_include_globs` option is correctly loaded from `pyproject.toml`.**Key Assertions:**

- {'name': 'Context include globs are correctly loaded', 'expected_value': 'True'}
- {'name': 'Context include globs are not empty', 'expected_value': 'False'}

Confidence: 80%**Tokens:** 119 input + 105 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	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 2ms 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 and used for signing requests.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': "hmac_key_file = 'path/to/hmac/key.txt'", 'actual': 'pyproject.toml contents'}
- {'name': 'hmac key file path', 'expected': '/path/to/hmac/key.txt', 'actual': 'pyproject.toml contents'}

Confidence: 80%

Tokens: 118 input + 140 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	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



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': {'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.'}}
- {'name': 'The include_param_values option should be a boolean value', 'expected_value': {'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.'}}
- {'name': 'The include_param_values option should be set to True', 'expected_value': {'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.'}}

Confidence: 80%**Tokens:** 116 input + 297 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/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,

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 1ms ⚡ 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': "include_phase = ['main', 'util']", 'actual': "include_phase = ['main', 'test']"}

Confidence: 80%

Tokens: 113 input + 102 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/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 1ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_pytest_invocation

Why Needed: To ensure that the include_pytest_invocation option is properly loaded from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'include_pytest_invocation = ["path/to/pytestInvocation.py"]', 'actual': 'include_pytest_invocation = []'}

Confidence: 80%

Tokens: 122 input + 114 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	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



AI ASSESSMENT

Scenario: test_load_invocation_redact_patterns**Why Needed:** To ensure that the 'invocation_redact_patterns' key in pyproject.toml is correctly loaded and redacted for sensitive information.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': 'Redacted invocation_redact_patterns key from pyproject.toml contents', 'actual': 'Redacted invocation_redact_patterns key in pyproject.toml contents'}

Confidence: 80%**Tokens:** 121 input + 111 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	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: test_load_litellm_api_base

Why Needed: To ensure that the `litellm_api_base` module is loaded correctly from the PyPI.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'The contents of pyproject.toml should be a valid Python project file with the following structure:\n\n[tool.pyproject.toml]\n\t[tool.litellm_api_base]\n\t\t"litellm_api_base":\n\t\t"path/to/litellm_api_base"\n', 'actual': 'The contents of pyproject.toml should be a valid Python project file with the following structure:\n\n[tool.pyproject.toml]\n\t[tool.litellm_api_base]\n\t\t"litellm_api_base":\n\t\t"path/to/litellm_api_base"\n', 'error_code': 0}

Confidence: 80%

Tokens: 122 input + 204 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	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 loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'api-key = "..."', 'actual': 'api-key = ...'}
- {'name': 'litellm_api_key loaded', 'expected': 'True', 'actual': 'True'}

Confidence: 80%

Tokens: 122 input + 131 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/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: Test loading litellm_token_json_key from pyproject.toml**Why Needed:** To ensure the coverage of litellm token JSON key is properly loaded in the project.**Key Assertions:**

- {'name': "pyproject.toml exists and has a 'litellm_token_json_key' section", 'expected_value': 'True'}
- {'name': "pyproject.toml has a 'litellm_token_json_key' section with the correct key", 'expected_value': {'key': 'litellm_token_json_key', 'value': 'some_value'}}}

Confidence: 80%**Tokens:** 125 input + 142 output = 267 total

COVERAGE

src/pytest_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_llm_token_output_format 2ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_llm_token_output_for

Why Needed: To ensure that the llm token output format is correctly loaded from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': "The contents of the pyproject.toml file should contain a section named 'tool' with a sub-section named 'llm' and a key named 'token_output_format'."}
- {'name': 'llm token output format', 'expected': "The value of the 'token_output_format' key in the pyproject.toml file should be a string representing the expected output format."}

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, 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_llm_token_refresh_command

2ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_llm_token_refresh_command

Why Needed: To ensure that the llm_token_refresh_command is loaded correctly from pyproject.toml.**Key Assertions:**

- {'name': 'pyproject.toml exists and has the correct content', 'expected_content': '''\ntoml\nversion 0.13.2\n[tool.pyproject.toml]\nversion = "0.13.2"\n\ntpy_requires = ["python >= 3.8"]\n\ntpython_requires = {exact: "py>=3.7,upper-exact"}\n\nttocreateresource = False\n\ttoctree = []\n\thelprootsync = True\n[tool.pyproject.toml]\nversion = "0.13.2"\n\ntpy_requires = ["python >= 3.8"]\n\ntpython_requires = {exact: "py>=3.7,upper-exact"}\n\nttocreateresource = False\n\ttoctree = []\n\thelprootsync = True\n'''\n, 'actual_content': '''\ntoml\nversion 0.13.2\n[tool.pyproject.toml]\nversion = "0.13.2"\n\ntpy_requires = ["python >= 3.8"]\n\ntpython_requires = {exact: "py>=3.7,upper-exact"}\n\nttocreateresource = False\n\ttoctree = []\n\thelprootsync = True\n[tool.pyproject.toml]\nversion = "0.13.2"\n\ntpy_requires = ["python >= 3.8"]\n\ntpython_requires = {exact: "py>=3.7,upper-exact"}\n\nttocreateresource = False\n\ttoctree = []\n\thelprootsync = True\n''', 'error_message': ''}'}

Confidence: 80%**Tokens:** 125 input + 457 output = 582 total

COVERAGE

src/pytest_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)

PASSED tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_llm_token_refresh_interval 3ms ⚡ 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_llm_token_refresh_int

Why Needed: To ensure that the llm_token_refresh_interval option is loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': "refresh_interval = '1h'", 'actual': 'None'}

Confidence: 80%

Tokens: 125 input + 101 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	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

1ms



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_max_concurrency

Why Needed: To ensure that the `max_concurrency` option is correctly loaded from the `pyproject.toml` file and used to configure the concurrency settings.

Key Assertions:

- {'name': 'Expected max_concurrency value in pyproject.toml', 'value': 4, 'expected_type': 'int'}

Confidence: 80%

Tokens: 116 input + 106 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_max_tests

2ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_max_tests

Why Needed: To ensure that the `max_tests` setting in `pyproject.toml` is correctly loaded and used to determine the number of tests to run.**Key Assertions:**

- {'name': 'pyproject.toml file exists', 'expected': 'pyproject.toml should exist'}
- {'name': 'pyproject.toml file content', 'expected': 'pyproject.toml should contain the correct setting for max_tests'}

Confidence: 80%**Tokens:** 113 input + 132 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/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 and provides accurate information about the project.

Key Assertions:

- {'name': 'pyproject.toml exists', 'expected_result': 'True'}
- {'name': 'pyproject.toml is a file', 'expected_result': 'True'}

Confidence: 80%

Tokens: 113 input + 105 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	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:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_ollama_host

Why Needed: To ensure that the ollama_host configuration is loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': "{'ollama_host': 'host_name'}", 'actual': "{'ollama_host': 'host_name'}"}

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/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 1ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load OMIT TESTS FROM COVERAGE

Why Needed: To ensure that omit_tests_from_coverage is correctly loaded from pyproject.toml when coverage is enabled.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'omitted tests are not included in the coverage report', 'actual': 'included tests are included in the coverage report'}

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 `max_chars` parameter in `pyproject.toml` is correctly loaded and validated.

Key Assertions:

- {'name': 'Expected value for max_chars', 'value': 100, 'expected_type': 'int'}
- {'name': 'Error message for invalid max_chars value', 'value': 'max_chars must be an integer between 1 and 9999', 'expected_type': 'str'}

Confidence: 80%

Tokens: 119 input + 138 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	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 file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'Collect only files', 'actual': 'Collect all files'}

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	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 feature is properly loaded from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists and has a correct timeout_seconds setting', 'value': 'True'}
- {'name': 'timeout_seconds value in pyproject.toml is within the expected range', 'value': 60}

Confidence: 80%

Tokens: 113 input + 117 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	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



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_batch_max_tests

Why Needed: To ensure that the `batch_max_tests` feature is correctly loaded from the PyProject file.

Key Assertions:

- {'assertion_type': 'file existence', 'expected_result': 'pyproject.toml exists in the test directory'}
- {'assertion_type': 'file content', 'expected_result': 'The contents of pyproject.toml match the expected format'}

Confidence: 80%

Tokens: 117 input + 123 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	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: Load batch parametrized tests**Why Needed:** Optimize Pytest configuration by reducing unnecessary test loading**Key Assertions:**

- {'assertion_type': 'File existence and content', 'condition': 'pyproject.toml exists and contains the expected file structure'}
- {'assertion_type': 'JSON structure consistency', 'condition': 'the JSON object in pyproject.toml is consistent with the expected structure'}

Confidence: 80%**Tokens:** 123 input + 109 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	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 is properly loaded and enabled in the PyPI token optimization process.

Key Assertions:

- {'name': 'Context compression is enabled', 'value': 'True'}
- {'name': 'Context compression is disabled', 'value': 'False'}

Confidence: 80%

Tokens: 117 input + 104 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	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` option in the PyProject is correctly loaded and applied to context lines.**Key Assertions:**

- {'name': 'Context line padding is applied correctly', 'expected_value': '\n# This is a test\n#\nwith line padding', 'actual_value': '```python\nThis is a test\nnwith line padding\nn```'}
- {'name': 'Context line padding is not applied when using the default value', 'expected_value': '\nThis is a test\nn', 'actual_value': '```python\nThis is a test\nn\n```'}

Confidence: 80%**Tokens:** 117 input + 170 output = 287 total

COVERAGE

src/pytest_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

5ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_prompt_tier

Why Needed: To ensure that the `prompt_tier` option is correctly loaded from the PyProject file.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': "The 'prompt_tier' key in the PyProject file should be present and contain a list of tier names."}
- {'name': 'prompt_tier value', 'expected': "The value of the 'prompt_tier' option should be a list of tier names."}

Confidence: 80%**Tokens:** 117 input + 141 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	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



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_batch_max_tests_too_small

Why Needed: This test is necessary because it checks if the `batch_max_tests` configuration option allows for too few tests to be validated.

Key Assertions:

- {'name': 'assert errors are not empty', 'message': "Expected no error messages, but got 'batch_max_tests must be at least 1'", 'type': 'error'}

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_padding

Why Needed: To ensure that the context line padding is correctly validated and raises an error when it's negative.

Key Assertions:

- {'message': 'context_line_padding must be 0 or positive', 'type': 'assertionError'}

Confidence: 80%

Tokens: 129 input + 88 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/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_compression

Why Needed: To test the validation of a scenario with an invalid context_compression setting.

Key Assertions:

- {'message': 'Invalid context_compression', 'type': 'AssertionError'}

Confidence: 80%

Tokens: 124 input + 78 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	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:

tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_invalid_prompt_tier

Why Needed: To ensure that the validation of invalid `prompt_tier` values does not produce any errors.

Key Assertions:

- {'assertion_type': 'contains', 'pattern': 'Invalid prompt_tier', 'value': "['Invalid prompt_tier']"}

Confidence: 80%

Tokens: 125 input + 95 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, 233-234, 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



AI ASSESSMENT

Scenario: tests/test_plugin_integration.py::TestPluginConfigLoading::test_config_defaults**Why Needed:** To ensure the config has safe defaults.**Key Assertions:**

- {'name': 'cfg is an instance of Config', 'description': 'The function should check if cfg is indeed an instance of Config.'}

Confidence: 80%**Tokens:** 119 input + 80 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	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



AI ASSESSMENT

Scenario:

tests/test_plugin_integration.py::TestPluginConfigLoading::test_markers_exist_in_config

Why Needed: The test requires that 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 97ms 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 to generate both JSON and HTML outputs.**Key Assertions:**

- The `report.json` file should exist after running the test.
- The `report.html` file should exist after running the test.
- Both files should have the expected contents before running the test.
- The `report.json` file should contain the correct data.
- The `report.html` file should not be empty or contain unexpected content.
- The plugin should correctly generate both JSON and HTML reports for the test function.

Confidence: 80%**Tokens:** 279 input + 146 output = 425 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 62ms ⚡ 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 59ms 7

AI ASSESSMENT

Scenario: Test that output directories are created if missing.**Why Needed:** This test prevents a regression where the plugin might not create nested directory structure for reports.**Key Assertions:**

- The `nested` directory should be created in the report.json file.
- The `report.json` file should exist in the specified directory path.
- The `nested` directory should have been created recursively if it does not already exist.
- The plugin should create a nested directory structure for reports even when output directories are missing.
- The test should fail if the report.json file is not created or exists outside of the specified directory path.

Confidence: 80%**Tokens:** 247 input + 142 output = 389 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 65ms 7

AI ASSESSMENT

Scenario: Test that fixture errors are captured in report.**Why Needed:** This test prevents a regression where the error from a failed plugin hook is not properly reported.**Key Assertions:**

- The 'error' key in the report should be set to 1 when there is an error.
- The 'summary' section of the report should contain an 'error' key with value 1.
- The 'error' value should match the number of times the test was run (in this case, once).
- The 'summary' section of the report should not be empty.
- The 'report.json' file should exist in the expected location.

Confidence: 80%**Tokens:** 286 input + 149 output = 435 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 174ms 7

AI ASSESSMENT

Scenario: Test pytest_runtest_makereport captures all outcomes.**Why Needed:** This test prevents a potential regression where the report does not capture all outcomes of the tests.**Key Assertions:**

- The test verifies that the report includes 'passed', 'failed', and 'skipped' outcomes.
- The test asserts that there are at least three types of outcomes in the report.
- The test checks if the report has a valid JSON structure with 'tests' and 'outcome' keys.
- The test ensures that the report does not contain any missing or invalid data.
- The test verifies that the report includes all test names (i.e., 'test_pass', 'test_fail', and 'test_skip').
- The test checks if the report has a correct order of outcomes (i.e., 'passed' before 'failed' and 'skipped').
- The test verifies that the report does not contain any duplicate outcomes.

Confidence: 80%**Tokens:** 335 input + 209 output = 544 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 57ms ⚡ 3

AI ASSESSMENT

Scenario:

tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_no_report_when_disabled

Why Needed: The test is failing because the generated report exists.

Key Assertions:

- `{'name': 'report_path.exists()', 'expected': {'status': 0, 'message': ''}, 'actual': {'status': 1, 'message': 'Report file not found'}}`

Confidence: 80%

Tokens: 150 input + 100 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	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

617ms



AI ASSESSMENT

Scenario: Test that --llm-pdf option enables the plugin.**Why Needed:** To prevent a regression where the plugin is not enabled due to an error in Playwright.**Key Assertions:**

- The test should run without any errors or warnings indicating that the plugin was disabled.
- The test should exit with code 0 (success) if the plugin was successfully enabled.
- The test should verify that the JSON output is generated even when not asked for a report.
- The test should verify that passing only --llm-pdf works to trigger the plugin logic.
- The test should verify that the plugin key validation passes without any issues.

Confidence: 80%**Tokens:** 435 input + 146 output = 581 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 60ms 7

AI ASSESSMENT

Scenario: Test that the pytest_sessionstart records start time is recorded correctly when running Pytest with --llm-report-json.

Why Needed: This test prevents a regression where the start time of the session might not be recorded correctly if the report is generated without it.

Key Assertions:

- The 'start_time' key should exist in the run_meta dictionary.
- The value of the 'start_time' key should be present in the run_meta dictionary.
- The 'start_time' value should match the current time when the test runs.
- The start time should be recorded correctly even if the test is not executed successfully.
- The report path should contain a 'start_time' key with a non-empty string value.
- The 'run_meta' dictionary should have a 'start_time' key with a datetime object value.
- The 'start_time' value should be in seconds since the epoch.
- The start time should not be affected by the test's execution status (e.g., failed, skipped).

Confidence: 80%

Tokens: 276 input + 225 output = 501 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_content_marker

1ms



COVERAGE

src/pytest_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_output_out_marker

1ms



COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_plugin_integration.py

Why Needed: To ensure that the requirement marker does not cause any errors.

Key Assertions:

- The requirement marker is used correctly.

Confidence: 80%

Tokens: 90 input + 46 output = 136 total

COVERAGE

src/pytest_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 46ms 6

AI ASSESSMENT

Scenario: Test the integration of report writer with pytest_llm_report models and ReportWriter.

Why Needed: This test prevents a regression where the report generation flow fails due to missing or corrupted JSON report files.

Key Assertions:

- Verify that the report.json file exists at the specified path.
- Check if the summary section of the report.json file contains the correct total and passed counts.
- Assert that the test_a.py nodeid is present in the HTML report.
- Verify that the test_b.py nodeid is also present in the HTML report.
- Check if the duration of each test case is correctly reported.
- Assert that the error message associated with test_b.py's failed outcome is 'AssertionError'.
- Verify that the total count of passed tests is 1, and the individual counts are correct.

Confidence: 80%

Tokens: 417 input + 184 output = 601 total

COVERAGE

src/pytest_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

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_disabled

Why Needed: This test is necessary to ensure that the collectreport plugin behaves correctly when it is disabled.

Key Assertions:

- {'name': 'pytest_collectreport', 'expected_result': 'Mock object session.config.stash.get was not called'}

Confidence: 80%

Tokens: 151 input + 90 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/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:

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_enabled

Why Needed: To test the functionality of collecting reports when pytest_collectreport is enabled.

Key Assertions:

- {'name': 'mock_collector.handle_collection_report.called_once_with(mock_report)', 'description': 'The handle_collection_report method should be called once with mock_report as an argument.'}

Confidence: 80%

Tokens: 204 input + 98 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/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: Because the plugin requires a valid session to function correctly.

Key Assertions:

- {'expected_type': 'Exception', 'message': 'pytest_collectreport is not available in this environment.'}

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: Because the test is checking if collectreport skips when session is None.

Key Assertions:

- {'name': 'pytest_collectreport', 'expected_result': 'None'}

Confidence: 80%

Tokens: 134 input + 78 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/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 is currently disabled by default. This test checks if the warning is raised when LLM is enabled.

Key Assertions:

- {'name': 'LLM is not disabled', 'value': 'True'}
- {'name': 'LLM is enabled', 'value': 'True'}

Confidence: 80%

Tokens: 143 input + 108 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	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 plugin is configured with invalid values.

Key Assertions:

- {'name': 'Expected a UsageError to be raised', 'expected': 'UsageError', 'actual': 'pytest_llm_report.plugin'}

Confidence: 80%

Tokens: 134 input + 90 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	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



AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_worker_skip

Why Needed: To ensure that the configure function skips on xdist workers and does not attempt to add markers for them.

Key Assertions:

- {'name': 'mock_config.addinvalue_line.called', 'expected': 1, 'message': 'Expected mock_config.addinvalue_line to be called once'}

Confidence: 80%

Tokens: 170 input + 105 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	17 lines (ranges: 328-330, 332-334, 336-338, 342-343, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginConfigureFallback::test_pyte_st_configure_fallback_load

3ms

2

AI ASSESSMENT

Scenario: Test fallback to load_config if Config.load is missing.

Why Needed: This test prevents a potential bug where the plugin would attempt to load configuration from an empty Config object, causing an error.

Key Assertions:

- mock_cfg.validate.asserts.return_value == []
- # Verify that validate() method returns an empty list when Config.load is missing

Confidence: 80%

Tokens: 747 input + 317 output = 1064 total

COVERAGE

src/pytest_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 3ms 3

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginLoadConfig::test_load_config_cli_overrides_pyproject

Why Needed: CLI options override in pyproject.toml is necessary for plugin functionality.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'pyproject.toml contents with CLI options overridden'}

Confidence: 80%

Tokens: 140 input + 88 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	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

116ms



AI ASSESSMENT

Scenario: Load configuration from PyProject.toml**Why Needed:** To test the plugin's ability to load configuration from a specific file format.**Key Assertions:**

- {'name': 'File extension', 'value': '.pyproject.toml'}

Confidence: 80%**Tokens:** 136 input + 68 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/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 regression in case the plugin is disabled and terminal summary is used.

Key Assertions:

- mocked stash.get() with _enabled_key set to False
- mocked stash.get() with _enabled_key set to True

Confidence: 80%

Tokens: 281 input + 78 output = 359 total

COVERAGE

src/pytest_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

COVERAGE

src/pytest_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

4ms



AI ASSESSMENT

Scenario: Test config loading from pytest objects (CLI) for maximal plugin functionality.

Why Needed: This test prevents regression in the maximal plugin's ability to load configuration files correctly, ensuring that the plugin can properly use pytest objects (CLI).

Key Assertions:

- The `report_html` option is set to `

Confidence: 80%

Tokens: 639 input + 383 output = 1022 total

COVERAGE

src/pytest_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 necessary because the `pytest_runtest_makereport` hookwrapper is not properly handling the generator when makereport is disabled.

Key Assertions:

- {'assertion_type': 'Mocking', 'expected_mock': ['mock_item', 'mock_call'], 'actual_result': {'config': {'stash': {}}}}

Confidence: 80%

Tokens: 220 input + 115 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/plugin.py	7 lines (ranges: 558-559, 562-563, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginRunttest::test_runttest_makereport_enabled

2ms



2

AI ASSESSMENT

Scenario: Test makereport calls collector when enabled.**Why Needed:** This test prevents a potential regression where the collector is not called when makereport is enabled.**Key Assertions:**

- The `mock_collector` should be called with the provided `mock_report` when `pytest_runttest_makereport` is called.
- The `mock_item.config.stash.get(_enabled_key)` method should return `True` when `_enabled_key` is present in the stash.
- The `mock_item.config.stash.get(_collector_key)` method should return `mock_collector` when `_collector_key` is present in the stash.
- The `mock_call.send(mock_outcome)` call should not raise an exception if `mock_outcome.get_result.return_value` is `None`.
- The `mock_collector.handle_runttest_logreport` method should be called with the provided `mock_report` and `mock_item` when `pytest_runttest_makereport` is called.

Confidence: 80%**Tokens:** 371 input + 220 output = 591 total

COVERAGE

src/pytest_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

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: TestPluginSessionHooks

Why Needed: To test that collection_finish calls collector when enabled.

Key Assertions:

- {'name': 'mock_collector.handle_collection_finish was called once with correct arguments', 'expected_args': ['[MagicMock(), MagicMock()]'], 'actual_args': [], 'assertion_type': 'method_call'}

Confidence: 80%

Tokens: 219 input + 90 output = 309 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_sessionstart_disabled

Why Needed: To ensure that the pytest_sessionstart hook is properly disabled in the stash configuration.

Key Assertions:

- {'name': 'mock_session.get.call_count', 'expected_value': 1, 'message': 'Expected mock_session.get to be called once.'}

Confidence: 80%

Tokens: 157 input + 95 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, 619-620)

PASSED

tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_sessionstart_enabled

1ms



AI ASSESSMENT

Scenario: Test that sessionstart initializes collector when enabled and the collector is properly created.

Why Needed: This test prevents a potential regression where the collector might not be initialized or created correctly if pytest_sessionstart is disabled.

Key Assertions:

- The key '_collector_key' should exist in the mock stash.
- The key '_start_time_key' should exist in the mock stash.
- A MagicMock instance with config.stash set to a MockStash instance should be created and assigned to pytest_sessionstart's config.
- The _collector_key and _start_time_key should both be present in the mock stash.
- The collector should have been successfully initialized when pytest_sessionstart is called on a session with enabled collectives.
- A KeyError or other exception should not be raised if pytest_sessionstart is called on a session without enable_collectives set to True.

Confidence: 80%

Tokens: 335 input + 191 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/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



AI ASSESSMENT

Scenario: Test pytest_addoption adds expected arguments to the parser.

Why Needed: This test prevents a bug where pytest_addoption does not add required arguments to the parser, potentially leading to unexpected behavior or errors.

Key Assertions:

- parser.getgroup.assert_called_with('llm-report', 'LLM-enhanced test reports')
- group.addoption.call_args_list[0][0].startswith('--llm-report')
- group.addoption.call_args_list[1][0].startswith('--llm-coverage-source')

Confidence: 80%

Tokens: 293 input + 123 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/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': [], 'actual_result': 'parser.addini.called'}

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/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_termin
al_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 summaries are generated.**Key Assertions:**

- The `report_html` option is set to 'out.html'.
- The `stash` dictionary contains the correct stash data.
- The `Coverage` class is mocked correctly with a valid coverage report.
- The `MockStash` class is created and its `load` method is called.
- The `report` method of the `Coverage` class is called once.
- The `patched_pathlib.Path.exists` function returns True when it should return False.
- The `patched_coverage.Coverage.report` method is called once with a valid coverage report.
- The `pytest_llm_report.coverage_map.CoverageMapper` class is patched correctly.

Confidence: 80%**Tokens:** 395 input + 189 output = 584 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_termin
al_summary_llm_enabled

3ms



AI ASSESSMENT

Scenario: Test that terminal summary with LLM enabled runs annotations correctly when provider is 'ollama'.

Why Needed: This test prevents regression where the plugin does not run annotations when LLM is enabled.

Key Assertions:

- Verify that `pytest_terminal_summary` is called once with correct configuration.
- Verify that `pytest_terminal_summary` passes a valid configuration to `llm.annotator.annotate_tests`.
- Verify that `pytest_llm_report.llm.annotator.annotate_tests` returns True for the provided configuration.

Confidence: 80%

Tokens: 477 input + 121 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	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_termin
al_summary_no_collector

2ms



AI ASSESSMENT

Scenario: Test terminal summary creates collector if missing.**Why Needed:** Prevents regression where terminal summary is not collecting metrics.**Key Assertions:**

- The stash object passed to pytest_terminal_summary() does not contain any _enabled_key or _config_key.
- The stash object passed to pytest_terminal_summary() contains True for _enabled_key but False for _config_key.
- The stash object passed to pytest_terminal_summary() is empty.
- _enabled_key is present in the stash object, but its value is False.
- The stash object passed to pytest_terminal_summary() does not contain any coverage mapper.
- The stash object passed to pytest_terminal_summary() contains a CoverageMapper object with an empty map.
- The stash object passed to pytest_terminal_summary() has no _enabled_key or _config_key.
- _enabled_key is present in the stash object, but its value is False and _config_key is not present.

Confidence: 80%**Tokens:** 391 input + 204 output = 595 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



AI ASSESSMENT

Scenario: Test terminal summary with aggregation enabled.

Why Needed: Prevents regression in aggregation functionality when terminal summary is enabled.

Key Assertions:

- The `aggregate_dir` option should be set to `/agg` for aggregation to work correctly.
- The `get()` method of the stash should return a report when aggregation is enabled.
- The `[]` index should also return a report when aggregation is enabled.
- The `ReportWriter` class should write JSON and HTML files when aggregation is enabled.
- The `Aggregator` class should aggregate reports correctly when aggregation is enabled.
- The `aggregate()` method of the Aggregator class should be called once with no arguments when aggregation is enabled.
- The `get()` method of the stash should return a report when aggregation is enabled and the stash has both `_enabled_key` and `_config_key` set to True.
- The `write_json()` and `write_html()` methods of the ReportWriter class should be called once with no arguments when aggregation is enabled.

Confidence: 80%

Tokens: 441 input + 225 output = 666 total

COVERAGE

src/pytest_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 during terminal summary.

Why Needed: This test prevents a regression where the plugin fails to calculate coverage percentages due to an OSError during load.

Key Assertions:

- The `load` method of `CoverageMapper` raises an `OSError` with message 'Disk full'.
- A warning is raised when trying to compute coverage percentage without loading the coverage map.
- The `report_writer` does not raise any warnings or errors during execution.
- The `pytest_terminal_summary` function returns a mock object that matches the expected behavior.
- The `MagicMock()` instance passed as an argument to `pytest_terminal_summary` is not modified by the test.
- The `mock_config.stash` dictionary contains the correct key-value pairs for the plugin configuration.
- The `mock_cov_cls.return_value` attribute of the mock object returns a new instance of `CoverageMapper` with the expected behavior.
- The `mock_cov.load.side_effect` attribute is set to an instance of `OSError` with the specified message.

Confidence: 80%

Tokens: 389 input + 235 output = 624 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)

PASSED

tests/test_prompts.py::TestContextAssembler::test_assemble_balanced_context 7ms 5

AI ASSESSMENT

Scenario: Tests the ContextAssembler with a balanced context configuration.

Why Needed: This test prevents regressions where the ContextAssembler is used with an unbalanced context configuration, which can lead to incorrect coverage metrics.

Key Assertions:

- The 'utils.py' file should be present in the assembled source code.
- The 'def util()' function should be found in the 'utils.py' file within the assembled source code.
- The ContextAssembler should assemble a balanced context configuration for the given test result.

Confidence: 80%

Tokens: 331 input + 117 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/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 assemble a complete context for the test 'test_a.py::test_1' and verify that it contains the expected code.

Key Assertions:

- {'assertion_type': 'contains', 'expected_value': 'def test_1():\n pass\\n'}

Confidence: 80%

Tokens: 176 input + 95 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	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: Verifies that the ContextAssembler can assemble a minimal context for a test file with a single test function.

Why Needed: This test prevents regression when using the 'minimal' llm_context_mode, as it ensures that only necessary code is included in the assembly.

Key Assertions:

- The source of the assembled context should contain only the specified test function.
- The context object should be empty (i.e., no additional code or variables are present).
- The 'test_1' assertion should be present in the source code of the assembled context.

Confidence: 80%

Tokens: 267 input + 130 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/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: Verify that the ContextAssembler does not exceed the specified context limits when assembling a test file.

Why Needed: This test prevents potential issues where the ContextAssembler exceeds the specified context limit, causing unexpected behavior or errors in the assembly process.

Key Assertions:

- The 'f1.py' file is present in the assembled context.
- The length of the 'f1.py' file does not exceed 40 bytes (20 bytes + truncation message).

Confidence: 80%

Tokens: 335 input + 109 output = 444 total

COVERAGE

src/pytest_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 context size limit.

Why Needed: This test prevents a bug where the ContextAssembler truncates long files when the context size is too small, potentially leading to incorrect results or data loss.

Key Assertions:

- The file content is preserved and can be accessed through the assembled context.
- The 'truncated' key in the context does not exist.
- The file path of the content is correct (i.e., 'f1.py')
- The length of the content matches its original value (20 bytes)
- The coverage report includes all lines and counts as expected
- The test result indicates that the assembly was successful (outcome='passed')

Confidence: 80%

Tokens: 361 input + 166 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/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_ed
ge_cases

1ms



AI ASSESSMENT

Scenario: Verify the function `'_get_test_source` returns an empty string when given a non-existent file.

Why Needed: This test prevents a potential bug where the function `'_get_test_source` throws a `ValueError` or raises an exception when given a non-existent file path.

Key Assertions:

- The function `'_get_test_source` returns an empty string for the given input.
- The function `'_get_test_source` does not raise any exceptions for the given input.

Confidence: 80%

Tokens: 275 input + 111 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/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

1ms



AI ASSESSMENT

Scenario: The test verifies that the ContextAssembler should exclude certain files from the llm context.

Why Needed: This test prevents a potential bug where the ContextAssembler incorrectly excludes files that are not actually present in the llm context.

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 + 110 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/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 in minimal mode returns no context files.

Why Needed: This test prevents regression where the assemble function does not generate any context files when run in minimal mode.

Key Assertions:

- context_files == {}
- def test_foo() in test_source

Confidence: 80%

Tokens: 298 input + 71 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/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 that occurs when llm_context_override is set to 'balanced' but the assembly process overrides it with a different mode.**Key Assertions:**

- Verify that the assembler uses the specified context override.
- Check if the module file is included in the context files.
- Ensure that the coverage entry points are correct for the module file.
- Verify that the llm_context_override is respected and not overridden by other factors.
- Test that the assembly process correctly overrides the default mode with the specified override.
- Confirm that the test passes even when the llm_context_override is set to 'balanced' but the assembly process overrides it with a different mode.

Confidence: 80%**Tokens:** 362 input + 168 output = 530 total

COVERAGE

src/pytest_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 balanced context excludes files matching exclude patterns.**Why Needed:** This test prevents a regression where the LLM context mode 'balanced' would include files that match the specified exclude patterns.**Key Assertions:**

- The file 'secret_config.py' should not be included in the balanced context.
- No files should be matched against the exclude pattern '*secret*'.
- The coverage entry for 'secret_config.py' should have a line count of 1, indicating it is only executed once.
- The LLM context mode 'balanced' should exclude all files matching the specified exclude patterns.
- No files should match any glob patterns in the LLM context mode 'balanced'.
- The coverage entry for the test file should not have any line ranges or line counts.
- The LLM context mode 'balanced' should only include files that do not match any exclude patterns.

Confidence: 80%**Tokens:** 331 input + 197 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/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_exists

Why Needed: To ensure that the ContextAssembler correctly handles cases where a balanced context file does not exist.

Key Assertions:

- {'name': 'context is empty', 'expected': {}, 'actual': {}}

Confidence: 80%

Tokens: 201 input + 85 output = 286 total

COVERAGE

src/pytest_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: Prevents a potential memory leak or unexpected behavior due to the large module size exceeding the allocated bytes limit.

Key Assertions:

- The content of the source file is truncated to prevent excessive memory usage.
- A message indicating truncation ('truncated') is included in the context.
- The total length of the content does not exceed the specified limit (120 bytes).
- The 'large_module.py' file itself contains a message indicating truncation ('truncated').

Confidence: 80%

Tokens: 405 input + 126 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/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



AI ASSESSMENT

Scenario:

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_no_coverage

Why Needed: To ensure that the ContextAssembler can correctly assemble a balanced context with no coverage.

Key Assertions:

- {'name': 'context is empty', 'expected': {}, 'actual': {}}

Confidence: 80%

Tokens: 162 input + 82 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	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 and the context is truncated.

Why Needed: This test prevents a potential bug where the ContextAssembler exceeds the maximum allowed bytes in the LLM context without encountering any errors, potentially leading to incorrect coverage metrics or unexpected behavior.

Key Assertions:

- context should be either empty (truncated) or contain only one file (non-truncated).
- context should not exceed the specified max_bytes limit.
- context should not contain more than llm_context_file_limit files. If it does, the test should fail with an appropriate error message.

Confidence: 80%

Tokens: 409 input + 139 output = 548 total

COVERAGE

src/pytest_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_delegates_to_balanced

Why Needed: Complete context delegates to balanced because it requires a full context for all node assertions, which can be complex and error-prone.

Key Assertions:

- {'assertion_type': 'includes', 'expression': 'module.py', 'value': 'True'}

Confidence: 80%

Tokens: 211 input + 98 output = 309 total

COVERAGE

src/pytest_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 function correctly handles an empty nodeid and returns an empty string.

Key Assertions:

- `{'description': 'The result of the test should be an empty string.', 'expected_result': '', 'type': 'assertion'}`

Confidence: 80%

Tokens: 148 input + 87 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/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_stops_at_next_def

Why Needed: To ensure that source extraction stops at the next function definition, even if it's not immediately following a test function.

Key Assertions:

- {'name': 'Expected source extraction to stop at next function definition', 'description': 'The code should contain a `def` statement immediately after a test function.', 'expected_value': 'True'}

Confidence: 80%

Tokens: 129 input + 117 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	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: Tests for the _get_test_source method

Why Needed: To ensure that the _get_test_source method handles cases where a test source file does not exist.

Key Assertions:

- {'assertion': 'The result of _get_test_source is an empty string when the input file does not exist.', 'expected_result': ''}

Confidence: 80%

Tokens: 142 input + 87 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	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: This test is necessary to ensure that the `_get_test_source` function correctly extracts functions with proper indentation.

Key Assertions:

- `{'expected': {'indentation': 4, 'expected_lines': ['def foo(): pass']}, 'actual': {'indentation': 1, 'expected_lines': []}}`

Confidence: 80%

Tokens: 118 input + 106 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	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

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_consecutive_lines**Why Needed:** To ensure that consecutive lines are compressed correctly.**Key Assertions:**

- {'expected': ['1-3'], 'actual': '1-3'}

Confidence: 80%**Tokens:** 106 input + 69 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/ranges.py	12 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65, 67)

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_duplicates**Why Needed:** To handle duplicate ranges in the input data.**Key Assertions:**

- {'assertion_type': 'equals', 'expected_value': '1-3'}

Confidence: 80%**Tokens:** 107 input + 68 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/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:** The test is necessary because the current implementation of 'compress_ranges' does not handle an empty input list correctly.**Key Assertions:**

- {'assertion': 'the function should return an empty string for an empty list', 'expected_result': '', 'message': 'Expected the function to return an empty string, but got [insert actual result here]'}

Confidence: 80%**Tokens:** 92 input + 109 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	2 lines (ranges: 29-30)

PASSED

tests/test_ranges.py::TestCompressRanges::test_mixed_ranges

1ms



AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_mixed_ranges**Why Needed:** To test the ability of the `compress_ranges` function to handle mixed ranges and singles.**Key Assertions:**

- {'expected': '1-3, 5, 10-12, 15', 'actual': '1-3, 5, 10-12, 15'}
- {'expected': 'True', 'actual': 'True'}

Confidence: 80%**Tokens:** 130 input + 108 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/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: Non-consecutive lines should be comma-separated.

Key Assertions:

- {'expected_result': '1, 3, 5', 'actual_result': '1, 3, 5'}

Confidence: 80%

Tokens: 113 input + 80 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/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: This test ensures that the `compress_ranges` function does not attempt to compress a single line of numbers.

Key Assertions:

- {'expected_value': '5', 'actual_value': 'True'}

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/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:** To test that two consecutive lines are compressed to a single range.**Key Assertions:**

- assert compress_ranges([1, 2]) == '1-2'

Confidence: 80%**Tokens:** 103 input + 65 output = 168 total

COVERAGE

src/pytest_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:** To ensure the function correctly handles unsorted input ranges.**Key Assertions:**

- {'expected': '1-3, 5', 'actual': '1-3, 5'}

Confidence: 80%**Tokens:** 110 input + 77 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/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: To ensure that the `expand_ranges` function handles empty strings correctly.

Key Assertions:

- {'name': 'assert expand_ranges returns an empty list for an empty string', 'description': 'The output of the `expand_ranges` function should be an empty list when given an empty string as input.'}

Confidence: 80%

Tokens: 90 input + 98 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/ranges.py	2 lines (ranges: 81-82)

PASSED

tests/test_ranges.py::TestExpandRanges::test_mixed

1ms



AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_mixed**Why Needed:** This test is necessary because the current implementation of `expand_ranges` only handles ranges and does not handle singles correctly.**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 + 119 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)
src/pytest_llm_report/util/ranges.py	11 lines (ranges: 81, 84-91, 93, 95)

PASSED

tests/test_ranges.py::TestExpandRanges::test_range

1ms



AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_range**Why Needed:** The range function should expand to a list of integers.**Key Assertions:**

- {'expected': [1, 2, 3], 'actual': '1-3'}
- {'expected': ['1', '2', '3'], 'actual': '[1, 2, 3]'}

Confidence: 80%**Tokens:** 99 input + 93 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/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



AI ASSESSMENT

Scenario: test_roundtrip**Why Needed:** To ensure that `compress_ranges` and `expand_ranges` are inverses.**Key Assertions:**

- {'name': 'original and compressed should be equal', 'expected_value': [1, 2, 3, 5, 10, 11, 12, 15], 'actual_value': [1, 2, 3, 5, 10, 11, 12, 15]}

Confidence: 80%**Tokens:** 134 input + 128 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/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:** The function `expand_ranges` is expected to handle a single input, producing a single output.**Key Assertions:**

- {'expected_value': [5], 'actual_value': ['5']}

Confidence: 80%**Tokens:** 95 input + 75 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/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



AI ASSESSMENT

Scenario: Test the format_duration function for milliseconds when input is less than 1 second.

Why Needed: Prevents a potential bug where the function does not correctly format durations in milliseconds when input is less than 1 second.

Key Assertions:

- The function should return '500ms' when input is 0.5 seconds.
- The function should return '1ms' when input is 0.001 seconds.
- The function should return '0ms' when input is 0.0 seconds.

Confidence: 80%

Tokens: 211 input + 120 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/render.py	2 lines (ranges: 65, 67)

PASSED

tests/test_render.py::TestFormatDuration::test_seconds

1ms



AI ASSESSMENT

Scenario: tests/test_render.py::TestFormatDuration::test_seconds

Why Needed: To ensure the `format_duration` function correctly formats durations in seconds for values greater than or equal to 1 second.

Key Assertions:

- {'name': "assert format duration of 1.23s returns '1.23s'", 'expected': '1.23s', 'actual': 'format_duration(1.23)'}
• {'name': "assert format duration of 60.0s returns '60.00s'", 'expected': '60.00s', 'actual': 'format_duration(60.0)'}

Confidence: 80%

Tokens: 116 input + 153 output = 269 total

COVERAGE

src/pytest_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 that all outcomes map to CSS classes.**Why Needed:** Prevents regression in rendering CSS classes for different outcomes.**Key Assertions:**

- outcome-to-css-class mapping is correct for each outcome.
- outcome-to-css-class mapping preserves the original outcome value.
- outcome-to-css-class mapping handles skipped outcomes correctly.
- outcome-to-css-class mapping handles xfailed and xpassed outcomes correctly.
- outcome-to-css-class mapping handles error outcome correctly.
- outcome-to-css-class mapping preserves the correct CSS class for passed, failed, and skipped outcomes.

Confidence: 80%**Tokens:** 263 input + 130 output = 393 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_render.py::TestOutcomeToCssClass::test_unknown_outcome**Why Needed:** Unknown outcomes are not handled correctly and may cause unexpected styling.**Key Assertions:**

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

Confidence: 80%**Tokens:** 102 input + 73 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/render.py	8 lines (ranges: 79-85, 87)

PASSED

tests/test_render.py::TestRenderFallbackHtml::test_renders_basic_report

1ms



3

AI ASSESSMENT

Scenario: Test renders basic report with fallback HTML.**Why Needed:** This test prevents a rendering issue where the full HTML document is not rendered correctly due to plugin or repository version issues.**Key Assertions:**

- assert '' in html
- assert 'Test Report' in html
- assert 'test::passed' in html
- assert 'test::failed' in html
- assert 'PASSED' in html
- assert 'FAILED' in html
- assert 'Plugin: v0.1.0' in html
- assert 'Repo: v1.2.3' in html

Confidence: 80%**Tokens:** 426 input + 153 output = 579 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test renders coverage for fallback HTML test.

Why Needed: Prevents rendering of non-coverage files and ensures accurate coverage reporting.

Key Assertions:

- The report includes the file "src/foo.py" which is part of the test.
- The assertion checks that there are exactly 5 lines in the rendered HTML.
- The assertion verifies that the file name 'src/foo.py' is present in the rendered HTML.
- The assertion ensures that the number of lines in the rendered HTML matches the coverage report.

Confidence: 80%

Tokens: 288 input + 121 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/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 LLM annotation rendering from failing due to missing confidence scores.**Key Assertions:**

- Tests login flow
- Prevents auth bypass
- Confidence: 85%
- Expected HTML content includes the following strings: 'Tests login flow', 'Prevents auth bypass', and 'Confidence: 85%'.

Confidence: 80%**Tokens:** 317 input + 110 output = 427 total

COVERAGE

src/pytest_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: The test verifies that the source coverage summary is included in the rendered HTML.

Why Needed: This test prevents a regression where the source coverage information is not displayed in the rendered HTML.

Key Assertions:

- Source Coverage summary should be present in the rendered HTML.
- Source code file path should be included in the HTML.
- Percentage of covered statements should be displayed in the HTML.
- Percentage of missed statements should be displayed in the HTML.
- Covered ranges should be correctly formatted in the HTML.
- Missed ranges should be correctly formatted in the HTML.

Confidence: 80%

Tokens: 331 input + 134 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/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: The test verifies that the 'XFailed' and 'XPassed' summary entries are included in the rendered HTML.

Why Needed: This test prevents a regression where the xfailed/xpassed summary is missing from the rendered HTML, potentially misleading users about the test results.

Key Assertions:

- The string 'XFailed' should be present in the rendered HTML.
- The string 'XPassed' should be present in the rendered HTML.
- Both 'XFailed' and 'XPassed' summary entries should be included in the rendered HTML.

Confidence: 80%**Tokens:** 283 input + 127 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	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:

- {'name': 'Hashes should be different', 'expected': {'hash1': '...'}}}
- {'name': 'Hashes should be different', 'expected': {'hash2': '...'}}}

Confidence: 80%

Tokens: 115 input + 100 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/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



AI ASSESSMENT

Scenario: tests/test_report_writer.py::TestComputeSha256::test_empty_bytes**Why Needed:** To ensure that the compute_sha256 function produces consistent hashes for empty byte sequences.**Key Assertions:**

- {'message': 'Expected hash to be equal', 'expected_value': '', 'actual_value': ''}
- {'message': 'Expected length of hash to be 64', 'expected_value': 64, 'actual_value': 64}

Confidence: 80%**Tokens:** 129 input + 104 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	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	1 lines (ranges: 55)

AI ASSESSMENT

Scenario: Test ReportWriter::test_build_run_meta verifies that the build run meta includes version info.

Why Needed: This test prevents regression where the report writer does not include version information in the build run meta.

Key Assertions:

- assert meta.duration == 60.0: The duration of the build should be 60 seconds.
- assert meta.pytest_version: The pytest version should have a value.
- assert meta.plugin_version == __version__: The plugin version should match the current version.
- assert meta.python_version: The python version should have a value.

Confidence: 80%

Tokens: 318 input + 132 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/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



AI ASSESSMENT

Scenario: Test verifies that the `build_summary` method correctly counts all outcome types and their corresponding tests.

Why Needed: This test prevents regression where a test might be skipped or ignored due to an incorrect count of outcome types.

Key Assertions:

- The total number of outcomes should match the sum of passed, failed, skipped, xfailed, xpassed, error outcome types.
- Each outcome type (passed, failed, skipped, xfailed, xpassed, error) should be correctly counted in the summary.
- The 'x' prefix for each outcome type indicates that it has an unknown number of tests.
- If a test is not included in any outcome type, its result should still be reported as 'error'.
- If all tests are skipped or ignored, the total count should remain 0.

Confidence: 80%

Tokens: 336 input + 182 output = 518 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test that the 'build_summary_counts' method counts outcomes correctly.

Why Needed: This test prevents a regression where the total count of passed, failed, and skipped tests is not accurate.

Key Assertions:

- The total number of tests should be equal to 4 (tests1, tests2, tests3, and tests4).
- The number of passed tests should be 2 (tests1 and tests3).
- The number of failed tests should be 1 (test3).
- The number of skipped tests should be 1 (test4).

Confidence: 80%

Tokens: 283 input + 137 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/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



AI ASSESSMENT

Scenario: Test that a new ReportWriter instance initializes correctly with its configuration.

Why Needed: This test prevents a potential bug where the writer's configuration is not properly initialized.

Key Assertions:

- The `config` attribute of the `ReportWriter` instance should be set to the provided `Config` object.
- The `warnings` attribute of the `ReportWriter` instance should be an empty list.
- The `artifacts` attribute of the `ReportWriter` instance should be an empty list.

Confidence: 80%

Tokens: 199 input + 118 output = 317 total

COVERAGE

src/pytest_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_as
sembles_tests

10ms



4

AI ASSESSMENT

Scenario: Test writes a report that includes all tests.

Why Needed: This test prevents regression by ensuring the ReportWriter can write reports with at least two tests.

Key Assertions:

- The length of the report.tests list should be equal to 2.
- The value of report.summary.total should be equal to 2.

Confidence: 80%

Tokens: 255 input + 81 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/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: The test is necessary to ensure that the ReportWriter class correctly includes the total coverage percentage in the report.

Key Assertions:

- {'assertion': 'report.summary.coverage_total_percent == 85.5', 'description': 'The total coverage percentage of the report should be equal to 85.5%'}

Confidence: 80%

Tokens: 132 input + 105 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/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 9ms 4

AI ASSESSMENT

Scenario: Test ReportWriter::test_write_report_includes_source_coverage verifies that the report includes source coverage summary.

Why Needed: This test prevents a regression where the report does not include source coverage information, which is crucial for understanding the code's quality and maintainability.

Key Assertions:

- The length of `report.source_coverage` should be 1.
- The file path of the first `source_coverage` entry in `report.source_coverage` should match `

Confidence: 80%

Tokens: 291 input + 237 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/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 9ms 4

AI ASSESSMENT

Scenario: Test ReportWriter::test_write_report_merges_coverage verifies that the report writer merges coverage into tests.

Why Needed: This test prevents a regression where the report writer does not merge coverage from individual tests to the overall report.

Key Assertions:

- The report should have at least one coverage entry for the first test.
- The file path of the first coverage entry should match the file path of the first test.
- Each coverage entry in the report should be a single object with 'file_path' and 'coverage' attributes.

Confidence: 80%

Tokens: 285 input + 125 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/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: The test prevents a regression where the atomic write operation fails, causing the report writer to fall back to direct write and potentially writing unnecessary or misleading warnings.

Key Assertions:

- The file `report.json` should exist at the specified path.
- Any warning messages written by the ReportWriterWithFiles class should have code 'W203'.

Confidence: 80%**Tokens:** 276 input + 110 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/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::testCreates_directory_if_missing

11ms



6

AI ASSESSMENT

Scenario: Test case

'tests/test_report_writer.py::TestReportWriterWithFiles::testCreates_directory_if_missing'

Why Needed: The test writer should create an output directory if it does not exist.**Key Assertions:**

- {'name': 'Directory exists after writing report', 'expected': True, 'actual': 'exists'}

Confidence: 80%**Tokens:** 171 input + 87 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/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: Verify that the `test_ensure_dir_failure` test captures a warning when creating a non-existent directory.

Why Needed: This test prevents a potential bug where the report writer fails to capture warnings for directories that cannot be created.

Key Assertions:

- The `writer.warnings` list contains any warnings with code 'W201' (indicating permission denied error).
- The `writer.warnings` list does not contain any warnings with code 'W100' (indicating directory creation failure).
- The `writer.warnings` list is empty if the directory cannot be created.
- The `writer.warnings` list contains at least one warning with code 'W201'.

Confidence: 80%

Tokens: 278 input + 158 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/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 'Should handle git command failures gracefully.'**Why Needed:** To prevent a test failure when the `git` command fails and no Git repository information can be retrieved.**Key Assertions:**

- The function `get_git_info()` should return an empty string for SHA and dirty flags if the `git` command fails.
- The function `get_git_info()` should not raise an exception if the `git` command is successful but returns no Git repository information.
- The function `get_git_info()` should handle the case where the `git` command fails with a specific error message ('Git not found') and return None for SHA and dirty flags.
- The function `get_git_info()` should handle the case where the `git` command fails but returns an empty string for SHA and dirty flags (e.g., when running on a system without Git installed).
- The function `get_git_info()` should not raise an exception if the `git` command is successful but returns no Git repository information, indicating that it cannot retrieve any information.
- The function `get_git_info()` should be able to handle different types of errors raised by the `git` command (e.g., permission issues, invalid options).
- The test should be able to pass even if the `git` command fails and no Git repository information can be retrieved due to external system limitations.

Confidence: 80%**Tokens:** 231 input + 297 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/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_htmlCreatesFile 46ms 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 fails to create an HTML file even when there are tests that fail or skip.

Key Assertions:

- The report.html file should exist.
- The report.html file should contain the expected content (test1, test2, PASSED, FAILED, Skipped, XFailed, XPassed, Errors).
- Test1 and Test2 should be present in the HTML content.
- PASSED and FAILED should be present in the HTML content.
- Skipped and XFailed/XPassed should be present in the HTML content.
- Errors should be present in the HTML content.

Confidence: 80%

Tokens: 366 input + 164 output = 530 total

COVERAGE

src/pytest_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 45ms 6

AI ASSESSMENT

Scenario: Test 'test_write_html_includes_xfail_summary' verifies that the report writer includes xfail outcomes in the HTML summary.

Why Needed: This test prevents regression of the issue where xfail outcomes are not included in the HTML summary.

Key Assertions:

- The test should find the string 'XFAILED' and 'XFailed' in the HTML content.
- The test should find the string 'XPASSED' and 'XPassed' in the HTML content.
- The test should assert that both 'XFAILED' and 'XFailed' are present in the HTML content.
- The test should assert that both 'XPASSED' and 'XPassed' are present in the HTML content.
- The test should verify that the report writer correctly includes xfail outcomes in the HTML summary.

Confidence: 80%

Tokens: 308 input + 181 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/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_jsonCreatesFile 10ms 5

AI ASSESSMENT

Scenario: Test verifies that the `ReportWriter` creates a JSON file with the specified configuration.

Why Needed: This test prevents regression where the `ReportWriter` fails to create a JSON file even when the report has an artifact.

Key Assertions:

- The file should exist at the expected path.
- At least one artifact should be tracked in the JSON file.

Confidence: 80%

Tokens: 265 input + 89 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/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_pdfCreatesFile 49ms 6

AI ASSESSMENT

Scenario: Test that `write_pdf` creates a PDF file when Playwright is available.**Why Needed:** This test prevents regression where the report writer does not create a PDF file even if Playwright is available.**Key Assertions:**

- The `report.pdf` path should be created in the temporary directory.
- Any artifacts written to the `report.pdf` path should match its original path.
- The `report.pdf` path should exist after the test completes.
- The `report.pdf` path should not be empty or None.
- The `writer.artifacts` list should contain any artifact with a matching path.
- Any artifacts written to the temporary directory should have a matching path in the `report.pdf` path.

Confidence: 80%**Tokens:** 478 input + 164 output = 642 total

COVERAGE

src/pytest_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 10ms 4

AI ASSESSMENT

Scenario: Test verifies that a warning is raised when Playwright is missing for PDF output.

Why Needed: This test prevents a potential bug where the report writer does not warn users about missing Playwright for PDF output.

Key Assertions:

- The `pdf_path` should exist after writing the report.
- At least one warning with code `W204_PDF_PLAYWRIGHT_MISSING` should be present in the `writer.warnings` list.

Confidence: 80%

Tokens: 311 input + 105 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	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 ensure that the report writer can handle cases where git info is not available.

Key Assertions:

- {'assertion': 'The function should return None for a nonexistent path.', 'expected_result': 'None'}

Confidence: 80%

Tokens: 123 input + 87 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/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



AI ASSESSMENT

Scenario: tests/test_report_writer_coverage.py::TestGetGitInfo::test_git_info_from_valid_repo

Why Needed: To ensure the `get_git_info` function returns a valid Git SHA or an error message when running in a non-Git repository.

Key Assertions:

- {'name': 'assert sha is None or isinstance(sha, str)', 'expected_result': {'type': 'NoneType', 'message': 'Expected `get_git_info` to return None or a string'}, 'actual_result': {'type': 'AssertionError'}}}

Confidence: 80%

Tokens: 159 input + 130 output = 289 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Tests that a fallback occurs when `'_git_info` import fails and the plugin's `get_plugin_git_info()` function returns an empty string or a string representation of a Git hash.

Why Needed: This test prevents regression where a fallback is not executed correctly when `'_git_info` import fails, potentially causing unexpected behavior in the application.

Key Assertions:

- The `sha` variable will be either None (if the fallback succeeds) or an empty string (if the fallback fails).
- The `dirty` variable will always be False because it is not modified during the test.
- The `get_plugin_git_info()` function will return a string representation of a Git hash if `'_git_info` import fails and the plugin's `get_plugin_git_info()` function returns an empty string or a string representation of a Git hash.
- The `sha` variable will be None after calling `get_plugin_git_info()` because it is not modified during this test.
- The `dirty` variable will always be False because it is not modified during the test.
- The `get_plugin_git_info()` function will return an empty string or a string representation of a Git hash if `'_git_info` import fails and the plugin's `get_plugin_git_info()` function returns None.
- The `sha` variable will be either None (if the fallback succeeds) or an empty string (if the fallback fails).
- The `dirty` variable will always be False because it is not modified during the test.
- The `get_plugin_git_info()` function will return a string representation of a Git hash if `'_git_info` import fails and the plugin's `get_plugin_git_info()` function returns an empty string or a string representation of a Git hash.

Confidence: 80%

Tokens: 253 input + 373 output = 626 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: test_get_plugin_git_info

Why Needed: To ensure that the plugin's Git info returns some values.

Key Assertions:

- {'name': 'returning a non-empty string for sha', 'description': 'The function should return either None or a non-empty string for the sha value.', 'expected_value': 'None'}
- {'name': 'returning a valid string for dirty', 'description': 'The function should return a valid string for the dirty value.', 'expected_value': 'False'}

Confidence: 80%

Tokens: 141 input + 133 output = 274 total

COVERAGE

src/pytest_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 11ms 5

AI ASSESSMENT

Scenario: Test atomic write fallback**Why Needed:** To ensure the report writer can handle errors and still produce a valid report.**Key Assertions:**

- {'name': 'The report file exists after writing', 'expected_result': 'True'}
- {'name': 'The report file is not empty after writing', 'expected_result': 'False'}

Confidence: 80%**Tokens:** 181 input + 93 output = 274 total

COVERAGE

src/pytest_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_pdfs_playwright_exception

122ms



AI ASSESSMENT

Scenario: Test PDF generation when playwright raises exception (lines 424-432) and expected a warning about PDF failure

Why Needed: Prevents regression where PDF generation fails due to playwright exception without raising a warning

Key Assertions:

- The `writer.warnings` list should contain a warning with code 'W201'
- The `writer.warnings` list should contain a warning with code 'W202'
- The `writer.warnings` list should contain a warning with code 'W203'
- The `writer.warnings` list should contain a warning with code 'W204'
- The `writer.warnings` list should contain a warning with code 'W205'
- The `writer.warnings` list should contain a warning with code 'W206'

Confidence: 80%

Tokens: 356 input + 175 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/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 that when playwright is not installed, a warning is generated about missing playwright and the report does not get created.

Why Needed: To prevent a potential bug where the PDF generation fails due to playwright not being installed.

Key Assertions:

- The test should have a warning indicating that playwright is not installed.
- Any of the warnings in the report should contain 'W204'.
- The report should not be created because playwright is missing.
- The file 'report.pdf' should not exist at the expected location.

Confidence: 80%**Tokens:** 293 input + 123 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	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_sourceCreatesTemp 35ms 6

AI ASSESSMENT

Scenario: Test _resolve_pdf_html_source creates temp file when no HTML source is provided.**Why Needed:** Prevents a potential issue where the report writer does not create a temporary PDF file if there are no HTML sources to resolve.**Key Assertions:**

- The path created by the _resolve_pdf_html_source method exists and has the correct suffix (.html).
- The path is not empty.
- The suffix of the path is correctly set to .html.

Confidence: 80%**Tokens:** 265 input + 109 output = 374 total

COVERAGE

src/pytest_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 36ms ⚡ 6

AI ASSESSMENT

Scenario: Test `_resolve_pdf_html_source` when configured HTML doesn't exist.

Why Needed: Prevents a bug where the test fails due to an incorrect assumption about the existence of a missing HTML file.

Key Assertions:

- The path to the resolved PDF report is not None and exists.
- The path to the resolved PDF report is not in the same directory as the temporary file.
- The resolved PDF report has the correct filename.
- The resolved PDF report does not contain any HTML content.
- The resolved PDF report is a valid PDF file with the correct metadata.
- The resolved PDF report is not corrupted or damaged.

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: Test _resolve_pdf_html_source uses existing HTML file.**Why Needed:** Prevents a potential bug where the report writer does not use an existing HTML file when resolving the PDF source.**Key Assertions:**

- The path to the existing HTML file is correctly set to `html_path`.
- The report writer correctly resolves the PDF source using `report`.
- The resolved path matches the expected value of `html_path`.
- The report is not created as a temporary file (`is_temp` is False).

Confidence: 80%**Tokens:** 266 input + 123 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/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



AI ASSESSMENT

Scenario: Test that `AnnotationSchema.from_dict` correctly creates an annotation from a dictionary with all required fields.

Why Needed: Prevents regression in case of missing or malformed input data, ensuring the application can handle invalid inputs without crashing.

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 + 119 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/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



AI ASSESSMENT

Scenario: Should convert to dictionary with all fields.

Why Needed: Prevent regression in schema validation when using to_dict method for complex schemas.

Key Assertions:

- assert 'scenario' in data
- assert 'why_needed' in data
- assert 'key_assertions' in data
- assert isinstance(data['scenario'], str)
- assert isinstance(data['why_needed'], str)
- assert all(isinstance(x, str) for x in data['key_assertions'])

Confidence: 80%

Tokens: 273 input + 112 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/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 95ms 8

AI ASSESSMENT

Scenario: The HTML report is created and exists as expected.

Why Needed: This test prevents a potential bug where the HTML report might not be generated correctly or exist even after running the tests.

Key Assertions:

- The file path for the report is correct and exists.
- The content of the report contains the expected string '
- The name 'test_simple' is present in the report content.

Confidence: 80%

Tokens: 264 input + 99 output = 363 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

135ms



8

AI ASSESSMENT

Scenario: test_html_summary_counts_all_statuses verifies that the HTML summary counts include all statuses.

Why Needed: This test prevents regression where the HTML summary does not include all statuses, potentially leading to incorrect reporting or misleading results.

Key Assertions:

- The function `assert_summary(labels: list[str], expected: int)` checks if each label in the `labels` list matches the corresponding count in the HTML report.
- It also checks for any missing summary labels and raises an assertion error with a descriptive message.
- For example, it will check if 'Total Tests' is present in the HTML report and its count matches the expected value of 6.
- Similarly, it will verify that each label (e.g., 'Passed', 'Failed', etc.) has a corresponding count in the HTML report.
- If any label is missing or does not match its expected count, it raises an assertion error with a message indicating which labels are missing and why.
- The test ensures that all statuses (including 'XFailed' and 'XPassed') are included in the summary counts.
- It also checks for any errors or exceptions raised during the execution of the `test_error` function.
- In case of an error, it raises an assertion error with a descriptive message indicating where the error occurred.

Confidence: 80%

Tokens: 621 input + 280 output = 901 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 70ms 7

AI ASSESSMENT

Scenario: The JSON report is created successfully.**Why Needed:** This test prevents a potential bug where the report generation fails due to missing or incorrect configuration files.**Key Assertions:**

- The report path exists and contains the expected data.
- The schema version of the report matches the expected value.
- The total number of tests passed is correct (2 in this case).
- At least one test failed, which is correctly reported as such.
- All test names are included in the summary.

Confidence: 80%**Tokens:** 295 input + 117 output = 412 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 62ms 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()` returns True.
- The 'why_needed' assertion checks if the annotation is present and prevents regressions.
- The 'key_assertions' list includes assertions related to the annotation presence.
- The 'choices' attribute of the mock completion function contains a message that matches the expected annotation content.
- The 'content' attribute of the chosen message in the mock completion function matches the expected annotation content.
- The 'message' attribute of the chosen message in the mock completion function is set to the expected annotation content.
- The 'choices' attribute of the mock completion function returns a list that includes an instance with the expected annotation content.
- The test passes if all assertions pass, preventing regressions.

Confidence: 80%**Tokens:** 385 input + 208 output = 593 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 98ms 14

AI ASSESSMENT

Scenario: Verify that LLM errors are surfaced in HTML output.**Why Needed:** Prevent regression where LLM errors are not reported correctly.**Key Assertions:**

- The test case `test_llm_error_is_reported` verifies that the LLM error is reported in the HTML output.
- The `pytester.makepytest` function creates a test function with an assertion.
- The `pytester.makeconftest` function patches the `litellm.completion` module to raise an error.
- The `pytester.makefile` function creates a pyproject.toml file with the `[tool.pytest_llm_report]` configuration.
- The test function `test_pass()` is called within the `makepytest` function.
- The `pytester.makepytest` function includes the `mock_completion` function in the test code.
- The `pytester.makeconftest` function sets the `litellm.completion` module to `mock_completion` before running the tests.

Confidence: 80%**Tokens:** 313 input + 224 output = 537 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)

AI ASSESSMENT

Scenario: Test the LLM opt-out marker.

Why Needed: Prevents regression in LLM opt-out functionality.

Key Assertions:

- The test verifies that the LLM opt-out marker is correctly recorded.
- The test asserts that the LLM opt-out marker is enabled for a single test.
- The test checks if the 'llm_opt_out' attribute of the first test in the report is set to True.
- The test verifies that the LLM opt-out marker is not present in the tests list.
- The test asserts that the number of tests in the report is 1.
- The test checks if the 'llm_opt_out' attribute of each test is set to False.
- The test verifies that the 'llm_opt_out' attribute of the first test matches the expected value.

Confidence: 80%

Tokens: 290 input + 184 output = 474 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)

AI ASSESSMENT

Scenario: Test the requirement marker functionality.

Why Needed: This test prevents a regression where the requirement marker is not recorded for tests with multiple requirements.

Key Assertions:

- The `pytest.mark.requirement` decorator should be applied to each test function that requires specific requirements.
- The `pytester.makepyfile()` method should create a file with the required pytest configuration.
- The `report_path` variable should be created and populated with the correct report path.
- The JSON data from the report path should contain a single test with the specified requirements.
- The 'requirements' key in the test's metadata should contain the expected requirements.
- The 'REQ-001' and 'REQ-002' strings should be present in the list of required requirements for each test.
- The `json.loads()` method should correctly parse the JSON data from the report path.

Confidence: 80%

Tokens: 307 input + 193 output = 500 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_out
comes 65ms 7

AI ASSESSMENT

Scenario: The test verifies that multiple xfailed tests are recorded in the report.

Why Needed: This test prevents regression of a scenario where multiple xfailed tests are not recorded in the report.

Key Assertions:

- Multiple xfailed tests are recorded in the report.
- Each xfailed test is counted only once.
- The count of xfailed tests matches the number of tests with xfailed outcomes.
- No xfailed tests are missed due to missing test outcomes.
- The test verifies that each xfailed test contributes to the total count.

Confidence: 80%

Tokens: 317 input + 127 output = 444 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

59ms  7

AI ASSESSMENT

Scenario: Verify that skipped tests are recorded and their count is accurate.

Why Needed: This test prevents a regression where the number of skipped tests might not be correctly reported in the LLM report.

Key Assertions:

- The 'summary' key under 'skipped' should contain an integer value equal to 1.
- The 'summary' key under 'skipped' should not contain any other values or strings.
- The number of skipped tests should be exactly 1 as per the pytester's configuration.
- The test skip reason is correctly set to 'test skip'.
- The test file name and path are correctly recorded in the report.json file.

Confidence: 80%

Tokens: 264 input + 152 output = 416 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

65ms  7

AI ASSESSMENT

Scenario: Test that xfailed tests are recorded and counted correctly in the report.

Why Needed: This test prevents a regression where xfailed tests are not properly reported in the report.

Key Assertions:

- The 'summary' key in the JSON report should contain the correct number of xfailed tests (1).
- The value of the 'xfailed' key under the 'summary' section should be equal to 1.
- If no xfailed tests are recorded, the 'summary' section should not have a 'xfailed' key or its value should be 0.

Confidence: 80%

Tokens: 264 input + 134 output = 398 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_parametrize_d_tests

62ms



7

AI ASSESSMENT

Scenario: Test parameterized tests are recorded separately.**Why Needed:** This test prevents regression in parametrized tests where the same input can produce different outputs.**Key Assertions:**

- The 'test_param' function is called with the correct argument 'x'.
- The assert statement inside the function checks if x is greater than 0.
- The total number of test cases is 3 as expected.
- The number of passed tests is also 3 as expected.
- The report.json file contains a summary with total and passed counts.

Confidence: 80%**Tokens:** 290 input + 127 output = 417 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

52ms



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:**

- {'name': 'result.stdout.fnmatch_lines', 'expected_result': ['*Example:*--llm-report*'], 'actual_result': [1, 2, 3]}

Confidence: 80%**Tokens:** 123 input + 105 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	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,

PASSED

tests/test_smoke_pytester.py::TestPluginRegistration::test_markers_registered 47ms 3

AI ASSESSMENT

Scenario: TestPluginRegistration test

Why Needed: To ensure that LLM markers are registered correctly.

Key Assertions:

- {'description': "Markers should be registered with pytester.runpytest('--markers')", 'expected_result': 'Markers should be registered'}
- {'description': 'Markers should match the expected lines in stdout.fnmatch_lines', 'expected_result': ['*llm_opt_out*', '*llm_context*', '*requirement*']}

Confidence: 80%

Tokens: 142 input + 114 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	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,

PASSED

tests/test_smoke_pytester.py::TestPluginRegistration::test_plugin_registered

53ms



AI ASSESSMENT

Scenario: tests/test_smoke_pytester.py::TestPluginRegistration::test_plugin_registered**Why Needed:** To verify that the plugin is registered correctly.**Key Assertions:**

- {'name': 'plugin registration', 'expected_result': 'The plugin was successfully registered.'}

Confidence: 80%**Tokens:** 118 input + 72 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	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_characters_in_nodeid 99ms 8

AI ASSESSMENT

Scenario: Test verifies that special characters in nodeid are handled correctly.

Why Needed: This test prevents a bug where special characters in the nodeid field cause the Pytest reporter to crash or produce invalid HTML reports.

Key Assertions:

- The string '
- The report path should exist and contain an 'html' substring.
- The content of the report path should contain the string 'html'.
- The nodeid field should not cause the Pytest reporter to crash or produce invalid reports.

Confidence: 80%

Tokens: 288 input + 127 output = 415 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



AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_boundary_one_minute**Why Needed:** To ensure the `format_duration` function can correctly format a duration of exactly one minute.**Key Assertions:**

- {'name': 'expected result', 'value': '1m 0.0s'}

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/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



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 should contain '\u00b5s' to indicate microsecond format.", 'expected_value': '\u00b5s'}
- {'description': 'The result should be equal to the expected value.', 'expected_result': '500\u00b5s'}

Confidence: 80%**Tokens:** 121 input + 116 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/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:

- {'assertion': "result == '500.0ms'", 'expected_result': '500.0ms'}

Confidence: 80%

Tokens: 119 input + 83 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/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:

- {'name': "result contains 'm'", 'expected': '1m 30.5s'}

Confidence: 80%

Tokens: 124 input + 83 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	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:

- {'expected': '3m 5.0s', 'actual': '3m 5.0s'}

Confidence: 80%

Tokens: 112 input + 84 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/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: The function `format_duration` should return a string representation of exactly one second in the format 'X.Xss'.

Key Assertions:

- {'expected_value': '1.00s', 'actual_value': '1.0'}

Confidence: 80%

Tokens: 101 input + 85 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_seconds_format

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_seconds_format**Why Needed:** To ensure that the seconds function correctly formats numbers under a minute.**Key Assertions:**

- {'name': "assert 's' in result", 'expected_result': "'s'", 'actual_result': '5.50s'}

Confidence: 80%**Tokens:** 110 input + 84 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/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:**

- {'expected_value': '1.0ms', 'actual_value': '1.0ms'}

Confidence: 80%**Tokens:** 111 input + 78 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	3 lines (ranges: 39, 41-42)

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_very_small_microseconds

Why Needed: To ensure that the `format_duration` function correctly handles 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)

AI ASSESSMENT

Scenario: Test ISO Format with UTC

Why Needed: To test the correct formatting of datetime objects with UTC timezone.

Key Assertions:

- {'description': "The formatted string should be in the format 'YYYY-MM-DDTHH:MM:SS+HH:MM:SS'", 'expected_value': '2024-01-15T10:30:45+00:00'}

Confidence: 80%

Tokens: 143 input + 98 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/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/test_time.py::TestIsoFormat::test_formats_naive_datetime

Why Needed: To ensure that the naive datetime (no timezone) is correctly formatted.

Key Assertions:

- {'expected': '2024-06-20T14:00:00', 'actual': '2024-06-20T14:00:00'}

Confidence: 80%

Tokens: 136 input + 92 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/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: Tests time module

Why Needed: To test the format of datetime objects with microseconds

Key Assertions:

- {'name': 'Result contains 12 digits', 'expected': '12345678901234567890', 'actual': '123456'}

Confidence: 80%

Tokens: 133 input + 73 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/time.py	1 lines (ranges: 27)

PASSED

tests/test_time.py::TestUtcNow::test_has_utc_timezone

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestUtcNow::test_has_utc_timezone

Why Needed: To ensure the datetime object has a valid UTC timezone.

Key Assertions:

- {'name': 'result.tzinfo is not None', 'expected': 'True'}
- {'name': 'result.tzinfo == UTC', 'expected': 'True'}

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)
src/pytest_llm_report/util/time.py	1 lines (ranges: 15)

PASSED

tests/test_time.py::TestUtcNow::test_is_current_time

1ms



AI ASSESSMENT

Scenario: Tests that the `is_current_time` function returns a valid JSON response when called with an empty input.

Why Needed: The test is necessary because the `utc_now()` function does not return a valid datetime object if no UTC time has been set. This can cause issues in tests that rely on this function returning a specific value.

Key Assertions:

- {'name': 'Expected result', 'type': 'assertion', 'value': 'datetime.datetime(2023, 1, 1, 0, 0, tzinfo=TimezoneInfo.get_default())'}

Confidence: 80%

Tokens: 116 input + 139 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/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:** To ensure that the `utc_now()` function returns a datetime object.**Key Assertions:**

- {'name': 'Type of result', 'expected': 'datetime', 'actual': 'isinstance(result, datetime)'}

Confidence: 80%**Tokens:** 94 input + 81 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/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, it should return a TokenRefreshError with the correct message.

Why Needed: This test prevents a potential bug where TokenRefresher does not handle command failures correctly and may silently fail or produce incorrect results.

Key Assertions:

- The function `get_token()` in `TokenRefresher` raises a `TokenRefreshError` with the correct message 'Authentication failed'.
- The error message returned by `get_token()` includes the string 'exit 1', which is expected for command failure.
- The test asserts that the error message contains the string 'Authentication failed' to verify its correctness.

Confidence: 80%

Tokens: 310 input + 150 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/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



AI ASSESSMENT

Scenario: The test verifies that the TokenRefresher raises an error when given an empty output.

Why Needed: This test prevents a potential bug where the TokenRefresher does not raise an error for an empty output, potentially masking a regression.

Key Assertions:

- assert 'empty output' in str(exc_info.value).lower()
- assert exc_info.value.value == 'empty output'
- assert isinstance(exc_info.value.value, str)
- assert len(exc_info.value.value) == 0
- # Check if the string is empty

Confidence: 80%

Tokens: 297 input + 127 output = 424 total

COVERAGE

src/pytest_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)

AI ASSESSMENT

Scenario: Test that 'force_refresh' bypasses cache and returns a new token when called with the same command.

Why Needed: This test prevents a bug where the TokenRefresher does not return a new token when force_refresh is called with the same command.

Key Assertions:

- The function `get_token()` of the `TokenRefresher` instance returns a new token after force refresh.
- The output of the `get_token()` method contains the expected token value.
- The number of calls to the `get_token()` method increases by one after force refresh.
- The `call_count` variable is updated correctly with each call to `get_token()`.
- The `token1` and `token2` variables are assigned the correct values based on the output of `get_token()`.
- The `force=True` parameter does not affect the return value of `get_token()`.

Confidence: 80%

Tokens: 346 input + 199 output = 545 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test the `TokenRefresher` class with a custom JSON key.**Why Needed:** This test prevents a bug where the `TokenRefresher` class does not properly handle custom JSON keys in the `get_token()` method.**Key Assertions:**

- The `token` variable should be equal to 'custom-key-token'.
- The `json_key` parameter passed to `TokenRefresher.get_token()` should match the expected value.
- The `subprocess.run()` function returns a `CompletedProcess` object with the correct JSON output.
- The `stdout` attribute of the `CompletedProcess` object is set to 'custom-key-token'.
- The `stderr` attribute of the `CompletedProcess` object is empty.
- The `json.dumps()` function correctly converts the expected JSON string to a Python dictionary.
- The `returncode` attribute of the `CompletedProcess` object is 0 (indicating successful execution).

Confidence: 80%**Tokens:** 303 input + 209 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	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



AI ASSESSMENT

Scenario: Verify that the `TokenRefresher` extracts a JSON token from the expected output.

Why Needed: This test prevents a potential bug where the `get-token` command returns an incorrect or malformed JSON response.

Key Assertions:

- The `token` key in the JSON output is set to 'json-token-value'.
- The `expires_in` value in the JSON output is set to 3600 (1 hour).
- The `output_format` parameter is set to 'json'.
- The `json_key` parameter is set to 'token'.
- The extracted token matches the expected string 'json-token-value'.

Confidence: 80%

Tokens: 308 input + 149 output = 457 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: The test verifies that the `TokenRefresher` extracts the correct token from the provided text output.

Why Needed: This test prevents a potential bug where the token is not extracted correctly due to an incorrect or incomplete text output.

Key Assertions:

- token == 'my-secret-token'
- stdout contains 'INFO: Processing...' and 'my-secret-token'
- stderr does not contain any relevant information
- the `get_token()` method returns the expected token value

Confidence: 80%

Tokens: 298 input + 114 output = 412 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test that TokenRefresher raises an error when receiving invalid JSON from the command.

Why Needed: This test prevents a potential bug where the TokenRefresher class does not handle invalid input correctly.

Key Assertions:

- The `get_token` method of the `TokenRefresher` instance should raise a `TokenRefreshError` exception when receiving an invalid JSON string.
- The error message returned by the `get_token` method should contain the word 'json' in lowercase.
- A non-empty string containing only whitespace characters or special characters (except for quotes) should be considered as an invalid JSON input.
- The `subprocess.CompletedProcess` result should include a returncode of 1, indicating that the command failed with a non-zero exit status.
- The `stdout` and `stderr` attributes of the `subprocess.CompletedProcess` result should contain strings containing 'not valid json' and '', respectively.
- The `TokenRefreshError` exception raised by the `get_token` method should be an instance of the `Exception` class.
- The error message returned by the `get_token` method should not be empty (i.e., it should contain at least one non-whitespace character).

Confidence: 80%

Tokens: 299 input + 272 output = 571 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test TokenRefresher.invalidate() clears cache and verifies it is invalidated after refresh.

Why Needed: This test prevents a potential bug where the TokenRefresher does not invalidate its cache after a successful refresh, leading to stale token values being returned.

Key Assertions:

- The function `invalidate()` of the `TokenRefresher` class clears the cache.
- A new token is generated and stored in the cache after calling `invalidate()`.
- The number of tokens in the cache increases by one after calling `invalidate()`.
- The cached token values are different from the actual token values obtained using `get_token()`.
- The function `invalidate()` does not return an error or exception when called with invalid arguments (e.g., no command, refresh interval, or output format).
- The function `invalidate()` clears the cache after a successful refresh, even if there are still tokens in the cache.
- The cached token values are different from the actual token values obtained using `get_token()` even after calling `invalidate()` multiple times.

Confidence: 80%

Tokens: 340 input + 230 output = 570 total

COVERAGE

src/pytest_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 does not raise an error when the required JSON key is missing, allowing the test to fail with a meaningful error message.

Key Assertions:

- The 'token' key should be present in the output of the get_token() method.
- The 'not found' string should be present in the error message.
- The error message should include the required JSON key ('token').
- The error message should not contain any other keys or values that are relevant to the test case.
- The 'token' key should be present in the output of the get_token() method even if it is missing from the input data.
- The 'not found' string should be present in the output of the get_token() method even if the required JSON key ('token') is not present in the input data.
- The error message should include a clear indication that the 'token' key was not found, without including any other relevant information.

Confidence: 80%

Tokens: 325 input + 244 output = 569 total

COVERAGE

src/pytest_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)

AI ASSESSMENT

Scenario: Test TokenRefresher thread safety by verifying that all threads receive the same token after concurrent execution.

Why Needed: This test prevents a potential bug where multiple threads may acquire the lock and access different tokens, leading to inconsistent results.

Key Assertions:

- All threads should acquire the lock before accessing the token.
- The first thread to acquire the lock should receive the same token as all other threads.
- If any thread acquires the lock after the initial call to get_token(), it should not affect the result of get_token() for subsequent calls.
- The output of get_token() should be consistent across all threads, with no variation in the token string.
- Subsequent calls to get_token() should return the same token as the first call made by any thread.
- If a thread acquires the lock before calling get_token(), it should not affect the results of subsequent calls to get_token().
- The output of subprocess.CompletedProcess() should be consistent across all threads, with no variation in the command string or return code.

Confidence: 80%

Tokens: 427 input + 229 output = 656 total

COVERAGE

src/pytest_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 timeout correctly by raising a TokenRefreshError when the 'get-token' command takes longer than 30 seconds to complete.

Why Needed: This test prevents a potential bug where the TokenRefresher does not handle command timeouts properly, potentially leading to unexpected behavior or errors.

Key Assertions:

- The 'get_token()' method of the TokenRefresher instance raises a TokenRefreshError with the message 'timed out' when the 'get-token' command takes longer than 30 seconds to complete.
- The 'get_token()' method does not raise a TokenRefreshError when the 'get-token' command completes within the specified timeout period (30 seconds in this case).
- The 'get_token()' method correctly raises a TokenRefreshError with the message 'timed out' when the 'get-token' command takes longer than 30 seconds to complete.
- The 'refresh_interval' attribute of the TokenRefresher instance is set to 3600 seconds (1 hour), which allows the 'get-token' command to timeout within this interval.
- The 'output_format' attribute of the TokenRefresher instance is set to 'text', which does not affect the timeout handling behavior of the 'get-token' command.

Confidence: 80%

Tokens: 279 input + 275 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/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)

AI ASSESSMENT

Scenario: Test Token Refreshing with Caching to verify that the function does not call the command again after caching.

Why Needed: This test prevents a potential bug where the `TokenRefresher` calls the `get-token` command multiple times if it is cached, potentially leading to unexpected behavior or errors.

Key Assertions:

- The output of the `get-token` command should be the same for both `token1` and `token2` after caching.
- The `call_count` variable should only increment once when the function is called with a cached token.
- Both `token1` and `token2` should have the same value after caching, indicating that they are the same token.
- The output of the `get-token` command for both calls should be identical to prevent multiple calls to the command.
- If the function is called again with a different cached token, it should not call the command again and return an empty string or a specific error message.
- The `subprocess.CompletedProcess` object returned by the fake run function should have an `stdout` field that contains the expected output for both calls to the `get-token` command.

Confidence: 80%

Tokens: 353 input + 254 output = 607 total

COVERAGE

src/pytest_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_coverage.py::TestTokenRefresherEdgeCases::test_command_failure_no_stderr

1ms



AI ASSESSMENT

Scenario: Test that a command failure results in an exception being raised and the expected error message is printed to stderr.

Why Needed: To ensure that TokenRefresher correctly handles command failures with no stderr output, preventing unexpected behavior or errors.

Key Assertions:

- The function `get_token()` should raise a `TokenRefreshError` when executed with a non-zero return code.
- The error message 'exit 1' should be printed to stderr.
- The string 'No error output' should be present in the error message.

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/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



AI ASSESSMENT

Scenario: TokenRefresherEdgeCases

Why Needed: Test handling of empty command string.

Key Assertions:

- {'message': "Expected a TokenRefreshError to be raised with the message 'empty' when the command string is empty.", 'type': 'assertion_error'}

Confidence: 80%

Tokens: 151 input + 75 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/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



AI ASSESSMENT

Scenario: Test the `test_invalid_command_string` function to verify it handles an invalid command string (shlex parse error).

Why Needed: Prevent a `TokenRefreshError` due to shlex parse errors when handling invalid shell syntax in the command.

Key Assertions:

- The `'TokenRefresher'` instance is not created with an invalid command string.
- The `'refresh_interval'` parameter is set to 3600 seconds.
- The `'output_format'` parameter is set to `'text'`.
- An exception of type `'TokenRefreshError'` is raised when calling `'get_token()'` on the `'refresher'` instance.
- The error message contains the string `'Invalid command string'`.

Confidence: 80%

Tokens: 251 input + 156 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/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



AI ASSESSMENT

Scenario: Test verifies that TokenRefresher raises a TokenRefreshError when the output is not a dictionary.

Why Needed: This test prevents regression where the function does not raise an error for non-dict JSON outputs.

Key Assertions:

- The 'output' argument of the get_token method should be a dict.
- The 'expected' key in the output should contain 'list'.
- The 'token' key in the expected dictionary should also be present and equal to 'array'.

Confidence: 80%

Tokens: 328 input + 117 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/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 potential TokenRefreshError due to invalid input.

Key Assertions:

- The function `json.dumps({`

Confidence: 80%

Tokens: 324 input + 120 output = 444 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test verifying that the `get_token` method raises a `TokenRefreshError` when the token value is not a string.

Why Needed: This test prevents regression where the `get_token` method incorrectly handles non-string token values.

Key Assertions:

- The `json.dumps()` function is called with an integer argument instead of a string.
- The `stdout` attribute contains an empty string.
- The `stderr` attribute is empty.
- An exception is raised with the message 'empty or not a string'.
- The `TokenRefreshError` is correctly raised.

Confidence: 80%

Tokens: 326 input + 136 output = 462 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test the TokenRefresher's handling of OSError when executing a command.**Why Needed:** Prevent regressions where the TokenRefresher fails to handle OSError during execution.**Key Assertions:**

- The `get_token()` method raises a `TokenRefreshError` with the message 'Failed to execute'.
- The `refresh_interval` parameter has no effect on the behavior of the `get_token()` method when an OSError occurs.
- The `output_format` parameter does not affect the handling of OSError during execution.
- The `fake_run()` function raises an `OSError` with a message 'Command not found'.
- The `TokenRefresher` instance is created with a command that returns an error.
- The `get_token()` method attempts to execute the command and raise an exception if it fails.

Confidence: 80%**Tokens:** 280 input + 184 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	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



AI ASSESSMENT

Scenario: Test that a TokenRefresher fails when the output has only blank lines after initial strip, but still contains non-whitespace wrapper lines.

Why Needed: Prevents a potential bug where a TokenRefresher incorrectly handles text output with only whitespace lines after an initial strip, potentially leading to incorrect token refresh results.

Key Assertions:

- The output should be empty after stripping blank lines.
- The output should contain non-whitespace wrapper lines but no whitespace content lines.
- The error message should indicate that there are no non-empty lines in the output.

Confidence: 80%

Tokens: 376 input + 128 output = 504 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test the test_whitespace_only_command to ensure it raises a TokenRefreshError when given an empty whitespace-only command string.

Why Needed: Prevent regression by ensuring the test covers the case where the input command is empty.

Key Assertions:

- The function `get_token()` should raise a `TokenRefreshError` with the message 'empty' when passed an empty whitespace-only command string.
- The error message should contain the word 'empty'.
- The test should fail when running on a system where the input command is empty.
- The test should not pass when running on a system where the input command contains non-whitespace characters.
- The function `get_token()` should raise a `TokenRefreshError` with a specific error message even if the input command is empty.
- The error message should contain the word 'empty'.
- The test should fail when running on a system where the input command does not contain any whitespace characters.

Confidence: 80%

Tokens: 236 input + 211 output = 447 total

COVERAGE

src/pytest_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_usage.py::test_token_usage_aggregation

5ms  2

AI ASSESSMENT

Scenario: Test token usage aggregation for multiple test cases**Why Needed:** Prevents regression in token usage reporting when aggregating results from multiple tests.**Key Assertions:**

- The total input tokens should be 30 and the total output tokens should be 15.
- The number of annotations should be 2.
- The total tokens should be 45.
- The annotation count should match the number of test cases.

Confidence: 80%**Tokens:** 775 input + 101 output = 876 total

COVERAGE

src/pytest_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)
---------------------------------	---