

Test Report

Run ID: 21197045261-py3.12 • Generated: 2026-01-21 04:20:47 • Duration: 119.85s

Plugin: v0.2.0 (a03dbe622cdc018f89b74731aed91adf1a582867) [dirty]

Repo: v0.2.0 (6ca9d0d9b8119ce18efb8514475229959a5a445b)

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

Token Usage: 131832 input, 75520 output (Total: 207352)

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: To ensure that tests with mocks and multiple assertions are correctly scored by the complexity estimator.

Key Assertions:

- The test is complex, requiring multiple assertions to accurately estimate its complexity.

Confidence: 80%

Tokens: 118 input + 73 output = 191 total

COVERAGE

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

PASSED

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

AI ASSESSMENT

Scenario:

tests/test_adaptive_prompts.py::TestComplexityEstimation::test_empty_source_zero_complexity

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

Key Assertions:

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

Confidence: 80%

Tokens: 136 input + 163 output = 299 total

COVERAGE

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

PASSED

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

AI ASSESSMENT

Scenario:

tests/test_adaptive_prompts.py::TestComplexityEstimation::test_simple_test_low_complexity

Why Needed: This test is needed because it checks for simplicity of tests and their complexity scores.

Key Assertions:

- {'name': 'simple_test', 'description': 'The simple test should have low complexity score.'}

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



AI ASSESSMENT

Scenario: Test invalid prompt tier

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:

- {'name': 'Invalid prompt tier', 'expected_value': 'invalid', 'actual_value': 'None'}

Confidence: 80%

Tokens: 126 input + 82 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	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` field is validated correctly and does not cause any issues.

Key Assertions:

- {'expected_value': 'minimal', 'actual_value': ['minimal', 'standard', 'auto'], 'error_message': 'Invalid prompt tier. Must be one of: minimal, standard, auto'}
- {'expected_value': 'standard', 'actual_value': ['minimal', 'standard', 'auto'], 'error_message': 'Invalid prompt tier. Must be one of: minimal, standard, auto'}

Confidence: 80%

Tokens: 142 input + 141 output = 283 total

COVERAGE

src/pytest_llm_report/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



AI ASSESSMENT

Scenario:

tests/test_adaptive_prompts.py::TestPromptTierSelection::test_auto_tier_complex_test

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

Key Assertions:

- {'name': 'standard_prompt_for_complex_tests', 'description': 'The auto-tier should use the standard prompt for complex tests.'}

Confidence: 80%

Tokens: 122 input + 86 output = 208 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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 system uses minimal prompts for simple tests, which can improve test execution speed and reduce memory usage.

Key Assertions:

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

Confidence: 80%

Tokens: 155 input + 94 output = 249 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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: Config override to minimal should always use minimal prompt.

Key Assertions:

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

Confidence: 80%

Tokens: 122 input + 74 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/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: To ensure consistent and reliable testing, it is essential to use the standard system prompt for all tests.

Key Assertions:

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

Confidence: 80%

Tokens: 148 input + 78 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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 the aggregation function with all policy when aggregating multiple reports.

Why Needed: This test prevents regression in case of multiple aggregated reports and ensures that both tests are retained.

Key Assertions:

- The aggregate result should have two tests (both retained).
- The aggregate result should contain both report1 and report2.
- Both retained tests should be included in the aggregate result.
- No test should be removed from the aggregate result when aggregating multiple reports.
- All aggregated reports should retain their original order.
- The aggregate result should not be None.
- The number of tests in the aggregate result should match the number of reports.

Confidence: 80%

Tokens: 364 input + 150 output = 514 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 ensure that the `aggregate` method of the Aggregator class raises an exception when the aggregation directory does not exist.

Key Assertions:

- {'name': 'Expected exception to be raised', 'description': 'The `aggregate` method should raise a `FileNotFoundException` when the aggregation directory does not exist.'}

Confidence: 80%

Tokens: 104 input + 104 output = 208 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: tests/test_aggregation.py::TestAggregator::test_aggregate_latest_policy

Why Needed: Prevents regression when comparing aggregate results across different times.

Key Assertions:

- The test verifies that the `aggregate` method correctly picks the latest policy for each report.
- It checks that the `tests` attribute of the result is populated with a single test, and its outcome is 'passed'.
- The assertion on `result.run_meta.is_aggregated` ensures that the run was aggregated.
- The assertions on `result.run_meta.run_count` and `result.summary.passed` verify the correct number of runs and passed tests respectively.
- Finally, it checks that only one test is reported in the summary.

Confidence: 80%

Tokens: 477 input + 166 output = 643 total

COVERAGE

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

PASSED

tests/test_aggregation.py::TestAggregator::test_aggregate_no_dir_configured

1ms

3

AI ASSESSMENT

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

Why Needed: To test that an aggregator function returns None when no directory configuration is provided.

Key Assertions:

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

Confidence: 80%

Tokens: 110 input + 77 output = 187 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` returns None when there are no reports.

Why Needed: Prevents a potential bug where the aggregate function does not handle cases with no reports.

Key Assertions:

- The `aggregate()` method should return `None` when called without any reports.
- No reports should be found in the directory.
- The absence of reports should prevent further aggregation.
- The `aggregate()` method should raise an exception or handle this case differently if it's not `None`.
- The test should fail when calling `aggregate()` with no reports.
- A test failure should occur when calling `aggregate()` without any reports.

Confidence: 80%

Tokens: 201 input + 146 output = 347 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 accurate token usage and LLM annotation serialization.

Key Assertions:

- Coverage was correctly deserialized from the JSON report.
- LLM annotation was correctly deserialized from the JSON report, including scenario, why needed, key assertions, and confidence.
- Token usage was correctly serialized from the JSON report.
- LLM token usage could not be re-serialized without the correct configuration.

Confidence: 80%

Tokens: 1002 input + 125 output = 1127 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 that source coverage summary is deserialized correctly.**Why Needed:** Prevents regression in source coverage reporting functionality.**Key Assertions:**

- The `source_coverage` list within the report should contain exactly one SourceCoverageEntry object.
- The `SourceCoverageEntry` object should have the correct file path and coverage statistics.
- The `file_path` attribute of the `SourceCoverageEntry` object should match the expected value.
- The number of statements in the source code should be equal to the reported coverage percentage.
- The missed statements should be less than or equal to the total statements in the source code.
- The covered statements should be greater than or equal to the reported coverage percentage.
- The coverage percentage should be a valid floating-point number between 0 and 100.
- The coverage range should be correctly formatted as '1-5, 7-11'.
- The missed range should be correctly formatted as '6, 12'.

Confidence: 80%**Tokens:** 395 input + 216 output = 611 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 `llm_coverage_source` option is not provided.**Key Assertions:**

- Verify that `load_coverage_from_source()` returns None when `llm_coverage_source` is not set.
- Verify that a UserWarning is raised when `llm_coverage_source` is not set and `coverage.py` does not exist.
- Verify that successful loading of coverage data occurs with mock coverage data.
- Verify that the `CoverageMapper` returns the correct coverage percentage.
- Verify that the `cov.report()` method is called on the mock coverage object.
- Verify that the `map_source_coverage()` method is called on the mock mapper object.
- Verify that the mock coverage class is instantiated correctly with mock cov and mock mapper objects.
- Verify that the mock cov report returns 80.0 as expected.

Confidence: 80%**Tokens:** 584 input + 206 output = 790 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 preserves the latest summary's total, passed, failed, skipped, xfailed, xpassed, and error counts when recalculating the summary.

Why Needed: This test prevents a regression where the latest summary is not correctly calculated if there are multiple tests with different outcomes.

Key Assertions:

- The total count of all tests should remain unchanged.
- The passed count should be equal to the number of tests that were passed in the latest summary.
- The failed count should be equal to the number of tests that were failed in the latest summary.
- The skipped count should be equal to the number of tests that were skipped in the latest summary.
- The xfailed count should be equal to the number of tests that had an error in the latest summary.
- The xpassed count should be equal to the number of tests that passed in the latest summary.
- The error count should remain unchanged from the latest summary.
- The coverage percentage should not change when recalculating the summary.

Confidence: 80%

Tokens: 473 input + 229 output = 702 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 file prevents the test from counting it as a valid aggregation run.

Why Needed: This test ensures that the aggregator correctly handles and skips reports containing invalid JSON data, preventing unnecessary runs.

Key Assertions:

- The `aggregator` instance is not skipped for the 'invalid.json' report.
- The `result` object contains a single run meta with a count of 1.
- The test asserts that the `result` object does not contain any additional run metadata.
- The test asserts that the `result` object has no additional keys or values.
- The test asserts that the `result` object is not `None`.
- The test verifies that only valid reports are counted in the aggregation result.

Confidence: 80%

Tokens: 352 input + 170 output = 522 total

COVERAGE

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

tests/test_aggregation_maximal.py

1 tests

PASSED

tests/test_aggregation_maximal.py::TestAggregationMaximal::test_recalculate_summary_coverage

1ms



4

AI ASSESSMENT

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

Why Needed: This test prevents regression in case of multiple tests having different outcomes, as it ensures that the correct number of tests are included in the coverage calculation.

Key Assertions:

- The total duration of all tests is 3.0 seconds.
- At least one test passed (outcome == 'passed')
- At least one test failed (outcome == 'failed')
- The total coverage percentage is 88.5%
- All tests have a unique nodeid ('t1', 't2')

Confidence: 80%**Tokens:** 299 input + 145 output = 444 total

COVERAGE

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

tests/test_annotator.py

13 tests

PASSED

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

AI ASSESSMENT

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

Why Needed: To test the batch optimization message

Key Assertions:

- {'name': 'mock_provider', 'expected_value': 'Mocked provider instance'}
- {'name': 'mock_cache', 'expected_value': 'Mocked cache instance'}
- {'name': 'mock_assembler', 'expected_value': 'Mocked 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 overwritten by subsequent requests.**Key Assertions:**

- {'name': 'Mocked cache should be updated after each request', 'expected_result': {'mocked_cache': {}}, 'actual_result': {'mocked_cache': {}}}
- {'name': 'Cache is not overwritten by subsequent requests', 'expected_result': {'mocked_cache': {}}, 'actual_result': {'mocked_cache': {}}}

Confidence: 80%**Tokens:** 101 input + 141 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	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 when the test cache is enabled.

Key Assertions:

- {'name': 'mock_cache.is_called_with', 'description': 'Mocked `is_called_with` method of mock_cache should be called with expected arguments.'}
- {'name': 'mock_assembler.is_called_with', 'description': 'Mocked `is_called_with` method of mock_assembler should be called with expected arguments.'}

Confidence: 80%

Tokens: 102 input + 135 output = 237 total

COVERAGE

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

PASSED

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

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_concurrent_annotation**Why Needed:** To ensure that annotators can process multiple requests concurrently without blocking or missing any annotations.**Key Assertions:**

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

Confidence: 80%**Tokens:** 98 input + 125 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	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 because concurrent annotation can lead to failures if not handled properly.**Key Assertions:**

- {'message': 'Mocked annotator failed to annotate the text.', 'expected_exception': 'annotator.exceptions.ConcurrentAnnotationError'}
- {'message': 'Mocked annotator returned an error code that indicates a failure.', 'expected_exception': 'annotator.exceptions.ConcurrentAnnotationError'}

Confidence: 80%**Tokens:** 116 input + 129 output = 245 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.**Key Assertions:**

- {'name': 'Mock provider should report progress', 'expected_value': 'progress', 'actual_value': 'report'}
- {'name': 'Mock cache should not affect progress reporting', 'expected_value': 'progress', 'actual_value': 'report'}

Confidence: 80%**Tokens:** 96 input + 115 output = 211 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 reports progress messages during the annotation process.**Key Assertions:**

- {'assertion_type': 'contains', 'pattern': 'progress message', 'expected_value': 'a progress message'}
- {'assertion_type': 'contains', 'pattern': 'percentage complete', 'expected_value': 'the percentage of the annotation task completed'}

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 mechanisms of the annotator.**Key Assertions:**

- {'name': 'mock_provider is called with an empty list', 'expected_result': [], 'actual_result': 1}
- {'name': 'mock_cache is not called when opt-out is True', 'expected_result': [], 'actual_result': 0}
- {'name': 'mock_assembler is called with the correct arguments when limit is True', 'expected_result': [{"text": "...", "type": "..."}, {"text": "...", "type": "..."}], 'actual_result': 1}

Confidence: 80%**Tokens:** 104 input + 185 output = 289 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 the annotator respects the rate limit for a given scenario.

Key Assertions:

- {'name': 'mock_provider.get_annotated_data()' should be called with a valid rate limit key', 'expected_value': 'valid_rate_limit_key'}
- {'name': 'mock_provider.get_annotated_data()' should be called within the allowed time window', 'expected_value': 'allowed_time_window'}

Confidence: 80%

Tokens: 112 input + 130 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, 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 prevent potential errors.**Key Assertions:**

- {'assertion': 'Annotations should be applied sequentially', 'expected_result': 'Annotations should be applied in the expected order'}
- {'assertion': 'Annotations without a previous annotation will not affect subsequent annotations', 'expected_result': 'Annotations without a previous annotation should not affect subsequent annotations'}

Confidence: 80%**Tokens:** 98 input + 111 output = 209 total

COVERAGE

src/pytest_llm_report/cache.py	1 lines (ranges: 153)
src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/annotator.py	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



AI ASSESSMENT

Scenario:

tests/test_annotator.py::TestAnnotateTests::test_sequential_annotation_error_tracking

Why Needed: Error tracking in sequential annotation is necessary to ensure that errors are properly reported and handled.**Key Assertions:**

- {'name': 'Mock provider should be called with error message', 'expected_result': 'Mock provider was called with an error message', 'actual_result': 'Mock provider did not call with an error message'}
- {'name': 'Mock cache should be cleared after error is reported', 'expected_result': 'Mock cache should be cleared after error is reported', 'actual_result': 'Mock cache was not cleared after error was reported'}
- {'name': 'Mock assembler should not report any errors in sequential annotation', 'expected_result': 'Mock assembler did not report any errors in sequential annotation', 'actual_result': 'Mock assembler reported an error in sequential annotation'}

Confidence: 80%**Tokens:** 105 input + 212 output = 317 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)

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

AI ASSESSMENT

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

Why Needed: LLM is disabled, so the test should do nothing.

Key Assertions:

- {'expected_value': '', 'actual_value': 'does nothing'}

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/llm/annotator.py	2 lines (ranges: 47-48)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

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

AI ASSESSMENT

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

Why Needed: The annotator should skip the annotation process if the provider is unavailable.

Key Assertions:

- {'name': 'mock_provider.is_available', 'expected_result': {'is_true': False}}

Confidence: 80%

Tokens: 101 input + 84 output = 185 total

COVERAGE

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

 tests/test_base_coverage_v2.py

2 tests

PASSED

tests/test_base_coverage_v2.py::test_base_parse_response_malformed_json_after_extract 1ms 5

AI ASSESSMENT

Scenario: Test Base Parse Response Malformed JSON After Extract**Why Needed:** To ensure that the `extract_json_from_response` function correctly handles malformed JSON responses and raises a meaningful error.**Key Assertions:**

- {'assertion_type': 'Error Code', 'expected_code': 400, 'actual_code': 404}
- {'assertion_type': 'Message', 'expected_message': 'Failed to parse LLM response as JSON'}

Confidence: 80%**Tokens:** 152 input + 113 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/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: Test that the `test_base_parse_response_non_string_fields` function correctly handles non-string fields in the response data.

Why Needed: This test prevents a potential bug where the function fails to parse non-string fields, potentially leading to incorrect results or errors.

Key Assertions:

- The function should be able to correctly identify and extract the expected key from the `response_data` dictionary.
- The function should be able to handle scenarios with multiple keys in the response data.
- The function should be able to ignore non-string values when extracting keys.
- The function should raise an error if it encounters a non-string value that is not recognized as a valid key.
- The function should correctly handle nested dictionaries or lists within the `response_data` dictionary.
- The function should preserve the original structure of the input data when parsing non-string fields.
- The function should be able to handle edge cases such as empty response data or null values.

Confidence: 80%

Tokens: 269 input + 209 output = 478 total

COVERAGE

src/pytest_llm_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)

PASSED

tests/test_base_maximal.py::TestGetProvider::test_get_gemini_provider

1ms



AI ASSESSMENT

Scenario: tests/test_base_maximal.py::TestGetProvider::test_get_gemini_provider**Why Needed:** The test is necessary because the `get_provider` function returns a `GeminiProvider` object which needs to be asserted as correct.**Key Assertions:**

- {'name': 'provider_type', 'expected_value': 'GeminiProvider'}
- {'name': 'provider_name', 'expected_value': 'gemini'}

Confidence: 80%**Tokens:** 104 input + 110 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_base_maximal.py::TestGetProvider::test_get_invalid_provider 2ms 4

AI ASSESSMENT

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

Why Needed: The test is necessary because it checks for the expected error when a valid configuration is passed to the get_provider function with an invalid provider.

Key Assertions:

- {'name': 'config is not None', 'expected': 'None', 'actual': "Config(provider='invalid')")}

Confidence: 80%

Tokens: 106 input + 97 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	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 test the functionality of getting a LiteLLM provider.**Key Assertions:**

- {'name': 'provider', 'expected_type': 'LiteLLMProvider'}

Confidence: 80%**Tokens:** 109 input + 75 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/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 a NoopProvider in the base maximization algorithm.

Key Assertions:

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

Confidence: 80%

Tokens: 104 input + 84 output = 188 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the `get_ollama_provider` function returns an instance of `OllamaProvider` correctly.

Key Assertions:

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

Confidence: 80%

Tokens: 108 input + 87 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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 LLM provider returns a boolean indicating availability.

Why Needed: This test prevents a potential regression where the LLM provider does not return a boolean indicating availability.

Key Assertions:

- The `is_available()` method should return True for both instances of the provider.
- The `is_available()` method should return True when called multiple times in quick succession.
- The `checks` attribute should increment correctly each time the `_check_availability()` method is called.

Confidence: 80%

Tokens: 280 input + 113 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/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 of `ConcreteProvider` returns the default model name specified in the configuration.

Key Assertions:

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

Confidence: 80%

Tokens: 114 input + 89 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	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: To ensure that the rate limits are set to None by default when no specific rate limit is specified.

Key Assertions:

- {'name': 'provider.get_rate_limits()' should return None, 'expected_result': 'None'}

Confidence: 80%

Tokens: 108 input + 76 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/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

Why Needed: To verify that the `is_local` method returns False for default LLM providers.

Key Assertions:

- {'name': 'provider.is_local() is False', 'expected_value': False, 'message': 'Expected provider.is_local() to return False'}

Confidence: 80%

Tokens: 105 input + 90 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	3 lines (ranges: 65-66, 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



AI ASSESSMENT

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

Why Needed: This test prevents a potential issue where context files are not added to the prompt, potentially causing unexpected behavior or errors.

Key Assertions:

- The 'src/module.py' file should be present in the prompt.
- The 'def helper()' function should be included in the prompt.

Confidence: 80%

Tokens: 261 input + 88 output = 349 total

COVERAGE

src/pytest_llm_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



AI ASSESSMENT

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

Why Needed: This test prevents a potential regression where the batch prompt is missing or incorrect due to incomplete parameterizations.

Key Assertions:

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

Confidence: 80%

Tokens: 330 input + 97 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/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: The test verifies that a single test generates a normal batch prompt.

Why Needed: This test prevents a potential regression where the batch prompt is missing for single tests.

Key Assertions:

- Test: test.py::test_foo should be included in the prompt.
- ``python

Confidence: 80%

Tokens: 269 input + 76 output = 345 total

COVERAGE

src/pytest_llm_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 the consistency of source hashes for a given function.

Why Needed: This test prevents regression where different versions of the same function produce different source hashes, potentially leading to inconsistencies in batch processing.

Key Assertions:

- The source code of the function should be the same across all hash computations.
- The length of each hash computation should be equal (32 bytes) for the given function.
- Different versions of the function should produce different source hashes.
- The source code should not change between hash computations for the same function.
- `_compute_source_hash(source)` should return a consistent hash value for the given source code.
- The length of the returned hash value should be equal (32 bytes) for the given source code.

Confidence: 80%

Tokens: 220 input + 165 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/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 key aspect of batching and caching in the Compute Source Hash system.

Key Assertions:

- {'name': 'hash1 != hash2', 'description': 'The two computed hashes are not equal.'}

Confidence: 80%

Tokens: 127 input + 93 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/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:** Because an empty source is a valid input for the ComputeSourceHash function.**Key Assertions:**

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

Confidence: 80%**Tokens:** 94 input + 89 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/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 option must be at least 1 to prevent invalid batch configurations.

Key Assertions:

- {'name': 'config.validate() returns errors', 'message': "Any error messages returned by the validate method should contain the string 'batch_max_tests'."}

Confidence: 80%

Tokens: 126 input + 97 output = 223 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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:

- {'value': 0, 'type': 'asserts', 'message': 'context_line_padding must be >= 0'}

Confidence: 80%

Tokens: 126 input + 81 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/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: Test invalid context compression

Why Needed: To ensure that the context compression mode is valid and does not cause any issues.

Key Assertions:

- {'description': "Context compression mode should be one of 'none', 'fast', or 'fastest'.", 'expected_value': 'none'}

Confidence: 80%

Tokens: 122 input + 80 output = 202 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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:

- {'assertion_type': 'not equal to', 'expected_value': 'context_compression', 'actual_value': 'none'}
- {'assertion_type': 'not in', 'expected_value': ['lines'], 'actual_value': ['none']}

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/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



AI ASSESSMENT

Scenario: tests/test_batching.py::TestGetBaseNodeid::test_nested_params**Why Needed:** This test ensures that complex parameters are fully stripped from the base node ID.**Key Assertions:**

- {'name': 'assert stripped base node ID', 'value': "_get_base_nodeid('test.py::test[a-b-c]') == 'test.py::test'", 'expected_result': 'test.py::test'}

Confidence: 80%**Tokens:** 109 input + 106 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/batching.py	2 lines (ranges: 53-54)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

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

AI ASSESSMENT

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

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

Key Assertions:

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

Confidence: 80%

Tokens: 133 input + 123 output = 256 total

COVERAGE

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

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

AI ASSESSMENT

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

Why Needed: To test the functionality of getting a base node ID without any parameters.

Key Assertions:

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

Confidence: 80%

Tokens: 123 input + 83 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/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.

Why Needed: This test prevents regression when large groups of tests are added to the batch.

Key Assertions:

- The length of each batch is correct.
- Each group has exactly two tests in it.
- Only one group has only one test in it.

Confidence: 80%

Tokens: 364 input + 81 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/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 TestGroupTestsForBatching

Why Needed: This test is needed because the batching feature is disabled.

Key Assertions:

- {'name': 'assert len(batches) == 2', 'expected': 2, 'actual': 1}

Confidence: 80%

Tokens: 170 input + 76 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/batching.py	6 lines (ranges: 92-93, 95, 97-99)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_batching.py::TestGroupTestsForBatching::test_parametrized _tests_grouped

1ms



4

AI ASSESSMENT

Scenario: Test 'Parametrized tests should be grouped together' verifies that parametrized tests are correctly grouped.

Why Needed: This test prevents regression by ensuring that parametrized tests are not scattered across multiple batches.

Key Assertions:

- The number of batches is equal to 1.
- Each batch has exactly 3 tests.
- Each batch is a parameterized test group.
- The base node ID of each batch is 'test.py::test_add'.
- All tests in the first batch are parametrized.

Confidence: 80%

Tokens: 346 input + 126 output = 472 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: The test verifies that single tests are grouped into batches without any grouping.

Why Needed: This test prevents a regression where single tests are not properly grouped in batches.

Key Assertions:

- Each test should be its own batch.
- There should be two batches with the same number of tests.
- The first batch should have exactly one test.
- The second batch should also have exactly one test.

Confidence: 80%

Tokens: 278 input + 99 output = 377 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	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 hash function is consistent across different inputs.**Key Assertions:**

- {'name': 'hash_source', 'type': 'function', 'expected_result': 'hash_source(source)'}

Confidence: 80%**Tokens:** 107 input + 73 output = 180 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:**

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

Confidence: 80%**Tokens:** 108 input + 74 output = 182 total

COVERAGE

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

PASSED

tests/test_cache.py::TestHashSource::test_hash_length

1ms



AI ASSESSMENT

Scenario: tests/test_cache.py::TestHashSource::test_hash_length**Why Needed:** To ensure the hash function is producing a fixed length output.**Key Assertions:**

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

Confidence: 80%**Tokens:** 100 input + 67 output = 167 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: Test that clearing the cache removes all entries.

Why Needed: To prevent a regression where multiple annotations are stored in the cache and then cleared, preventing them from being re-added.

Key Assertions:

- The number of cache entries should be reduced to 2 after clearing.
- The annotation 'test::a' with hash1 should be removed from the cache.
- The annotation 'test::b' with hash2 should also be removed from the cache.
- All annotations in the cache should have been cleared.

Confidence: 80%

Tokens: 283 input + 122 output = 405 total

COVERAGE

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

PASSED

tests/test_cache.py::TestLlmCache::test_does_not_cache_errors

1ms  4

AI ASSESSMENT

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

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

Confidence: 80%**Tokens:** 157 input + 73 output = 230 total

COVERAGE

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

PASSED

tests/test_cache.py::TestLlmCache::test_get_missing

1ms  4

AI ASSESSMENT

Scenario: tests/test_cache.py::TestLlmCache::test_get_missing**Why Needed:** To test that the get method returns None for missing entries in the cache.**Key Assertions:**

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

Confidence: 80%**Tokens:** 128 input + 74 output = 202 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 setting and getting an annotation from the cache preserves its original value.

Why Needed: Prevents bypass by ensuring annotations are stored in the cache, preventing them from being overwritten without user input.

Key Assertions:

- Check that the annotation is set correctly with the correct scenario.
- Check that the confidence of the annotation remains unchanged after retrieval.
- Check that the annotation's original key matches its retrieved value.
- Verify that no other annotations are stored in the cache without a corresponding user input.
- Ensure that the annotation's status (in this case, 'Tests login') is preserved across multiple retrievals.

Confidence: 80%

Tokens: 286 input + 143 output = 429 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 implementation of the CollectionError class does not validate its structure, which can lead to errors when collecting and processing these types of errors.

Key Assertions:

- {'name': 'nodeid', 'expected_value': 'test_bad.py'}
- {'name': 'message', 'expected_value': 'SyntaxError'}

Confidence: 80%

Tokens: 124 input + 109 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)

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

AI ASSESSMENT

Scenario:

tests/test_collector.py::TestCollectorCollectionErrors::test_get_collection_errors_initially_empty

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

Key Assertions:

- {'name': 'assert get_collection_errors() == []', 'expected_result': [], 'actual_result': []}

Confidence: 80%

Tokens: 114 input + 91 output = 205 total

COVERAGE

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

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

AI ASSESSMENT

Scenario: tests/test_collector.py::TestCollectorMarkerExtraction

Why Needed: Default llm_context_override should be None.

Key Assertions:

- {'name': 'llm_context_override', 'value': 'None'}

Confidence: 80%

Tokens: 136 input + 65 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)

PASSED

tests/test_collector.py::TestCollectorMarkerExtraction::test_llm_opt_out_default_false

1ms 2

AI ASSESSMENT

Scenario: tests/test_collector.py::TestCollectorMarkerExtraction

Why Needed: Default llm_opt_out should be False.

Key Assertions:

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

Confidence: 80%

Tokens: 136 input + 82 output = 218 total

COVERAGE

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

PASSED

tests/test_collector.py::TestCollectorOutputCapture::test_capture_enabled_by_default

1ms 3

AI ASSESSMENT

Scenario:

tests/test_collector.py::TestCollectorOutputCapture::test_capture_enabled_by_default

Why Needed: This test is needed because the collector is not enabled by default.

Key Assertions:

- {'name': 'assert config.capture_failed_output is True', 'expected_result': True}

Confidence: 80%

Tokens: 104 input + 77 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/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 4000. This is necessary to ensure that the output does not exceed this limit, which could cause issues with downstream processing.

Key Assertions:

- {'name': 'assert capture_output_max_chars is equal to 4000', 'expected_value': 4000, 'message': 'The default value of `capture_output_max_chars` is 4000.'}

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

PASSED

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

AI ASSESSMENT

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

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

Key Assertions:

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

Confidence: 80%

Tokens: 206 input + 78 output = 284 total

COVERAGE

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

PASSED

tests/test_collector.py::TestCollectorXfailHandling::test_xfail_passed_is_xpassed

1ms



AI ASSESSMENT

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

Why Needed: xfail passes should be recorded as xpassed.

Key Assertions:

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

Confidence: 80%

Tokens: 205 input + 75 output = 280 total

COVERAGE

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

PASSED

tests/test_collector.py::TestTestCollector::test_create_collector

1ms



AI ASSESSMENT

Scenario: Test the creation of a TestCollector instance.

Why Needed: This test prevents a potential bug where the collector is not initialized with any results, leading to incorrect assertions in subsequent tests.

Key Assertions:

- collector.results should be an empty dictionary.
- collector.collection_errors should be an empty list.
- collector.collected_count should be 0.

Confidence: 80%

Tokens: 205 input + 88 output = 293 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 test results by nodeid.

Key Assertions:

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

Confidence: 80%

Tokens: 227 input + 95 output = 322 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_f
inish 1ms 3

AI ASSESSMENT

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

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

Key Assertions:

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

Confidence: 80%

Tokens: 256 input + 112 output = 368 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

1ms



AI ASSESSMENT

Scenario:

tests/test_collector_maximal.py::TestCollectorInternals::test_capture_output_disabled_via_handle_report

Why Needed: Capturing output via handle_runttest_logreport is disabled in this scenario.

Key Assertions:

- {'name': 'result.captured_stdout is None', 'expected': {'type': 'NoneType', 'message': 'Should not capture if config disabled'}}}

Confidence: 80%

Tokens: 211 input + 97 output = 308 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 verify that the `collector._capture_output` method correctly captures stderr and returns it to the test case.

Key Assertions:

- {'name': 'captured_stderr', 'expected_value': 'Some error'}

Confidence: 80%

Tokens: 157 input + 77 output = 234 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 test that the collector captures stdout correctly.

Key Assertions:

- {'name': 'assert captured stdout is correct', 'expected_value': 'Some output'}

Confidence: 80%

Tokens: 157 input + 66 output = 223 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 creates a result with item markers and verifies the expected behavior.

Why Needed: This test prevents regression in case of incorrect marker extraction or incorrect usage of item markers.

Key Assertions:

- The `param_id` attribute of the created result is set to 'param1'.
- The `llm_opt_out` attribute of the created result is set to True.
- The `llm_context_override` attribute of the created result is set to 'complete'.
- All required requirements are included in the `requirements` list of the created result.

Confidence: 80%

Tokens: 382 input + 131 output = 513 total

COVERAGE

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

PASSED

tests/test_collector_maximal.py::TestCollectorInternals::test_extract_error_repr_crash

1ms

3

COVERAGE

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

PASSED

tests/test_collector_maximal.py::TestCollectorInternals::test_extract_error_string

1ms

3

AI ASSESSMENT

Scenario: test_collector_internals

Why Needed: To ensure that the `extract_error` method returns the correct string representation of an error.

Key Assertions:

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

Confidence: 80%

Tokens: 130 input + 75 output = 205 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



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 _extract_skip_reason returns None for no longrepr', 'expected_value': 'None'}

Confidence: 80%

Tokens: 130 input + 93 output = 223 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



AI ASSESSMENT

Scenario: When the `handle_collection_report` method is called with a report that indicates a collection error, it should record this error in the `collection_errors` list.

Why Needed: This test prevents a potential bug where a collection report fails and does not trigger any errors, potentially leading to missed issues or incorrect reporting.

Key Assertions:

- The length of `collector.collection_errors` is set to 1.
- The value of `collector.collection_errors[0].nodeid` is set to `

Confidence: 80%

Tokens: 273 input + 119 output = 392 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



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 might not handle reruns correctly, potentially leading to incorrect results or errors.

Key Assertions:

- res.rerun_count should be equal to 1 (expected)
- res.final_outcome should be 'failed' (expected)
- collector.results['t:r'] should contain the expected data

Confidence: 80%

Tokens: 281 input + 113 output = 394 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: When the `handle_runttest_setup_failure` test function is called with a setup log report that fails, then it should record an error in the collector's report.

Why Needed: This test prevents a potential bug where the setup log report is not properly recorded when it fails, potentially causing incorrect reporting of the test result.

Key Assertions:

- res.outcome == 'error'
- res.phase == 'setup'
- res.error_message == 'Setup failed'

Confidence: 80%

Tokens: 300 input + 111 output = 411 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



AI ASSESSMENT

Scenario: TestCollectorReportHandling test handle_runttest_teardown_failure: Should record error if teardown fails after pass.

Why Needed: Prevents regression by ensuring that the collector logs an error when a teardown failure occurs after a successful run.

Key Assertions:

- assert res.outcome == 'error', assert res.phase == 'teardown', assert res.error_message == 'Cleanup failed'
- assert not res.wasxfail
- assert res.duration is None

Confidence: 80%

Tokens: 391 input + 111 output = 502 total

COVERAGE

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

tests/test_context_compression.py

12 tests

PASSED

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

AI ASSESSMENT

Scenario: Test invalid compression mode

Why Needed: To ensure that the Context Compression validation correctly identifies and reports invalid compression modes.

Key Assertions:

- {'message': "Context compression is not one of the allowed values: 'gzip', 'deflate', 'lz4', 'snappy'."}

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_context_compression.py::TestConfigValidation::test_negative_padding_invalid

1ms



AI ASSESSMENT

Scenario:

tests/test_context_compression.py::TestConfigValidation::test_negative_padding_invalid

Why Needed: Negative padding should fail validation.

Key Assertions:

- {'message': 'context_line_padding is not a valid value for the context_line_padding parameter.', 'expected_value': -1}

Confidence: 80%

Tokens: 121 input + 77 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	27 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273-274, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

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

AI ASSESSMENT

Scenario: TestConfigValidation

Why Needed: To ensure that valid compression modes can pass validation without raising any errors.

Key Assertions:

- {'assertion_type': 'type', 'expected_type': 'str', 'actual_type': 'None'}

Confidence: 80%

Tokens: 135 input + 70 output = 205 total

COVERAGE

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

PASSED

tests/test_context_compression.py::TestConfigValidation::test_zero_padding_valid

1ms



AI ASSESSMENT

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

Why Needed: Zero padding should be valid.

Key Assertions:

- {'message': 'context_line_padding is not present in any error message', 'expected_value': 0}

Confidence: 80%

Tokens: 122 input + 73 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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:

tests/test_context_compression.py::TestContextCompression::test_compression_mode_lines

Why Needed: Lines compression mode should be available.

Key Assertions:

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

Confidence: 80%

Tokens: 113 input + 69 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_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 for default context compression.

Key Assertions:

- {'description': 'Line padding should default to 2.', 'expected_value': 2, 'actual_value': 0}

Confidence: 80%

Tokens: 106 input + 85 output = 191 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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



AI ASSESSMENT

Scenario: Test that contiguous covered lines do not have gap indicators.

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

Key Assertions:

- The count of '#' characters should be zero for contiguous lines without gaps.
- The string '# L3:' should be present in the result.
- The string '# L4:' should be present in the result.
- The string '# L5:' should be present in the result.
- The line numbers (L3, L4, and L5) should not be missing or appear out of order.

Confidence: 80%

Tokens: 293 input + 135 output = 428 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: The test is needed because it checks the behavior of the `'_extract_covered_lines` method when there are no lines to extract.

Key Assertions:

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

Confidence: 80%

Tokens: 130 input + 82 output = 212 total

COVERAGE

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

PASSED

tests/test_context_compression.py::TestExtractCoveredLines::test_extract_multiple_ranges

1ms



4

AI ASSESSMENT

Scenario: Test 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 function `extract_covered_lines` should find the correct range for each covered line and add a gap indicator between them.
- The function `extract_covered_lines` should extract two separate ranges, one starting from L3 and another from L15.
- The function `extract_covered_lines` should not miss any gaps in the extracted ranges.
- The function `extract_covered_lines` should correctly identify the start line of each range (L3 and L15).
- The function `extract_covered_lines` should handle overlapping covered lines correctly.
- The function `extract_covered_lines` should preserve the original order of covered lines in the input list.

Confidence: 80%

Tokens: 274 input + 196 output = 470 total

COVERAGE

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

PASSED

tests/test_context_compression.py::TestExtractCoveredLines::test_extract_single_line

1ms



4

AI ASSESSMENT

Scenario: Single covered line should be extracted with padding.

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

Key Assertions:

- The function `_extract_covered_lines()` includes lines 2, 3, and 4 in the result with 1 line padding.
- The function `_extract_covered_lines()` includes lines 2, 3, and 4 in the result without any padding.
- The function `_extract_covered_lines()` correctly handles cases where there are multiple covered lines.

Confidence: 80%

Tokens: 302 input + 129 output = 431 total

COVERAGE

src/pytest_llm_report/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: Verify that padding does not extend beyond file boundaries.

Why Needed: This test prevents a bug where padding exceeds the number of lines in the source code, causing incorrect coverage reports.

Key Assertions:

- The extracted covered lines should have line numbers within the range [1, 3].
- The uncovered lines should not have negative line numbers (0).
- The uncovered lines should be at least as many as the number of lines in the source code minus one.
- The uncovered lines should be at most three.

Confidence: 80%

Tokens: 288 input + 124 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/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 necessary because the current implementation truncates context when it's too long. This can cause issues with the expected output.**Key Assertions:**

- {'assertion': "The prompt should not contain 'truncated'.", 'expected_result': 'short content'}

Confidence: 80%**Tokens:** 158 input + 91 output = 249 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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: test_smart_distribution verifies that the smart distribution algorithm allocates budget to F1 and F2 correctly, resulting in zero waste.

Why Needed: This test prevents regression by ensuring that the smart distribution algorithm does not allocate too much budget to F1 or F2, leading to wasted tokens.

Key Assertions:

- F1 should get full allocation (40 tokens)
- F2 should get truncated allocation (~480 tokens)
- Total used tokens should be equal to total allocated tokens
- Wasted tokens should be less than total allocated tokens
- F2 content in prompt should contain 'f2' and have more characters than F1 content
- F2 content in prompt should not exceed 800 characters

Confidence: 80%

Tokens: 773 input + 163 output = 936 total

COVERAGE

src/pytest_llm_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 content and truncates them according to the budget.

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

Key Assertions:

- The file 'f1' is present in the prompt.
- The file 'f2' is present in the prompt.
- The string 'truncated' is present in the prompt.

Confidence: 80%

Tokens: 317 input + 109 output = 426 total

COVERAGE

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

PASSED

tests/test_context_limits.py::test_truncation_logic

1ms



AI ASSESSMENT

Scenario: When the test_truncation_logic function is called with a large context file, it should truncate the prompt to fit within the 100 token limit.

Why Needed: This test prevents a potential bug where the prompt exceeds the 100 token limit and is truncated or skipped.

Key Assertions:

- The length of the prompt should be less than 5 times the limit (100 tokens) minus some overhead.
- Either '[... truncated]' or 'Relevant context' should be present in the prompt if it's heavily truncated or skipped.

Confidence: 80%

Tokens: 397 input + 125 output = 522 total

COVERAGE

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

tests/test_context_util.py

28 tests

PASSED

tests/test_context_util.py::TestCollapseEmptyLines::test_collapse_three_empty_lines

1ms



3

AI ASSESSMENT

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

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

Key Assertions:

- {'description': 'The result should be the same as the expected output.', 'expected_output': 'line1\n\nline2'}

Confidence: 80%

Tokens: 128 input + 86 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	1 lines (ranges: 108)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestCollapseEmptyLines::test_many_empty_lines

1ms



3

AI ASSESSMENT

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

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

Key Assertions:

- {'description': 'The collapsed string should contain only one blank line.', 'expected_result': 'line1\n\nline2'}

Confidence: 80%

Tokens: 127 input + 85 output = 212 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestCollapseEmptyLines::test_preserve_two_empty_lines

1ms

3

AI ASSESSMENT

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

Why Needed: To ensure that the `collapse_empty_lines` function preserves up to 2 consecutive newlines.

Key Assertions:

- {'expected_value': 'line1\n\nline2', 'actual_value': 'line1\n\nline2'}

Confidence: 80%

Tokens: 125 input + 90 output = 215 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestCollapseEmptyLines::test_single_newline

1ms

3

AI ASSESSMENT

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

Why Needed: Preserves single newlines in collapsed text.

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: This test ensures that the `optimize_context` function always collapses empty lines, regardless of whether the `strip_docs` and/or `strip_comms` flags are set.

Key Assertions:

- {'assertion': 'The output of the function is a string containing only non-empty lines.', 'expected_output': 'line1\\nline2'}

Confidence: 80%

Tokens: 137 input + 112 output = 249 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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



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 without any unnecessary changes.**Key Assertions:**

- {'expected': {'message': 'Module docstring should be removed'}, 'actual': {'message': 'Module docstring remains unchanged'}}}
- {'expected': {'message': 'Namespace object should be created and populated with data'}, 'actual': {'message': 'Namespace object is empty'}}}

Confidence: 80%**Tokens:** 96 input + 127 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/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



AI ASSESSMENT

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

Why Needed: The default behavior of the `optimize_context` function should strip all docstrings, but leave comments intact.

Key Assertions:

- {'name': 'docstring stripping', 'expected': '', 'actual': 'def foo():'}

Confidence: 80%

Tokens: 100 input + 89 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/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:** To ensure that the context optimizer handles empty sources correctly.**Key Assertions:**

- {'name': 'result is an empty string', 'expected': '', 'actual': ''}

Confidence: 80%**Tokens:** 95 input + 74 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/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestOptimizeContext::test_source_with_only_whitespace

1ms

3

AI ASSESSMENT

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

Why Needed: This test is necessary because the current implementation of optimize_context does not handle source lines with only whitespace correctly.

Key Assertions:

- The function should return a string with one or more newline characters and/or tabs, but instead returns a single newline character.

Confidence: 80%

Tokens: 115 input + 84 output = 199 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	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 optimize context by removing unnecessary documentation from the code.**Key Assertions:**

- {'assertion_type': 'docstring stripping', 'reason': 'The function docstring is not necessary and can be removed.'}
- {'assertion_type': 'comment stripping', 'reason': 'Comments are also unnecessary and can be removed.'}

Confidence: 80%**Tokens:** 95 input + 109 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/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 remove unnecessary comments from the code without affecting its functionality.

Key Assertions:

- {'assertion_type': 'strip_comments', 'expected_result': 'The function foo() is defined, but with no docstring.'}

Confidence: 80%

Tokens: 95 input + 84 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/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: To ensure that the optimizer can correctly strip out unnecessary code in certain scenarios.

Key Assertions:

- {'assertion': 'The optimizer should be able to keep both `foo()` and its explicit call.', 'expected_result': {'code': 'def foo():', 'context': ''}, 'actual_result': {'code': '', 'context': ''}}

Confidence: 80%

Tokens: 94 input + 111 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	6 lines (ranges: 108, 124, 126, 129, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestStripComments::test_comment_after_string_with_hash

1ms

3

AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripComments::test_comment_after_string_with_hash

Why Needed: To ensure that comments are stripped from strings containing hash (#) symbols, regardless of their position.

Key Assertions:

- {'description': 'Source string should contain a comment after the string with hash (#).', 'expected_result': 'url = "http://example.com#anchor"'}
- {'description': 'Comment should be stripped from the source string.', 'expected_result': ''}

Confidence: 80%

Tokens: 134 input + 124 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/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 handle escaped quotes in strings without modifying the original string.

Key Assertions:

- {'name': 'Basic behavior', 'expected': '# comment', 'actual': '# comment'}

Confidence: 80%

Tokens: 133 input + 76 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	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 comments in Python code, which can be useful for parsing and processing scripts.

Key Assertions:

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

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/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 without affecting the functionality of the code.**Key Assertions:**

- The test should be able to successfully remove all comments from the source code.
- The output of the stripped source code should match the expected output.

Confidence: 80%**Tokens:** 91 input + 81 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	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:

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

Confidence: 80%

Tokens: 135 input + 90 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	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: Preserves # inside single-quoted strings in source code.

Key Assertions:

- {'description': 'Expected result', 'expected_value': "url = 'http://example.com#anchor'", 'actual_value': 'result'}

Confidence: 80%

Tokens: 135 input + 91 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/context_util.py	7 lines (ranges: 81-82, 86, 88-90, 93)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestStripComments::test_strip_simple_comment

1ms

3

AI ASSESSMENT

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

Why Needed: To test the functionality of the `strip_comments` function, which removes simple end-of-line comments from source code.

Key Assertions:

- {'assertion_type': 'equals', 'expected_value': 'x = 1'}

Confidence: 80%

Tokens: 119 input + 85 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/context_util.py	7 lines (ranges: 81-82, 86, 88-90, 93)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestStripComments::test_strip_standalone_comment

1ms

3

AI ASSESSMENT

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

Why Needed: To strip standalone comments from the code, which can improve readability and reduce noise.

Key Assertions:

- {'assertion_type': 'strip_comments', 'expected_result': ['This is a comment']}

Confidence: 80%

Tokens: 99 input + 80 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/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 `strip_docstrings` function handles syntax errors correctly.

Key Assertions:

- {'expected': {'source': 'def foo(unclosed paren'}, 'actual': {'source': 'def foo(unclosed parentheses'}}}

Confidence: 80%

Tokens: 119 input + 93 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/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



AI ASSESSMENT

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

Why Needed: To strip unnecessary docstrings from a module.

Key Assertions:

- {'assertion': 'The function should return an empty list for the given source code.', 'expected_output': []}
- {'assertion': 'The function should not modify the original source code.', 'expected_output': ''}

Confidence: 80%

Tokens: 95 input + 103 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/context_util.py	30 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-59, 61-62, 64, 66-69)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestStripDocstrings::test_preserves_multiline_data_strings

1ms

3

AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripDocstrings::test_preserves_multiline_data_strings

Why Needed: Preserve multiline data strings in docstrings for consistency and readability.

Key Assertions:

- {'assertion': 'The triple-quoted string is preserved.', 'expected_result': 'The triple-quoted string is preserved.'}
- {'assertion': 'The variable foo() does not contain the preserved string.', 'expected_result': 'The variable foo() does not contain the preserved string.'}

Confidence: 80%

Tokens: 103 input + 125 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/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 output.

Key Assertions:

- {'assertion': 'The string \'x = "hello world".\' is preserved in the test output.', 'expected_output': '\'x = "hello world".\''}

Confidence: 80%

Tokens: 102 input + 89 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/context_util.py	25 lines (ranges: 27, 29, 33, 35-36, 38-45, 49, 51-52, 55-56, 58, 61, 64, 66-69)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

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

AI ASSESSMENT

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

Why Needed: Preserving strings in structures is important for maintaining code readability and avoiding potential issues with string literals.

Key Assertions:

- {'description': 'The function should preserve the original string literals.', 'expected_result': 'Should preserve strings in lists/dicts.'}
- {'description': 'The function should not modify the original string literals.', 'expected_result': 'Does not modify the original string literals.'}

Confidence: 80%

Tokens: 146 input + 126 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/context_util.py	27 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-56, 58, 61, 64, 66-69)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestStripDocstrings::test_strip_multiline_docstring

1ms

3

AI ASSESSMENT

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

Why Needed: To strip multiline docstrings from Python code.

Key Assertions:

- {'assertion': 'The function should return a JSON object with the expected structure.', 'expected_result': {'scenario': 'tests/test_context_util.py::TestStripDocstrings::test_strip_multiline_docstring', 'why_needed': 'To strip multiline docstrings from Python code.', 'key_assertions': [...]}, 'actual_result': {"scenario": "tests/test_context_util.py::TestStripDocstrings::test_strip_multiline_docstring", "why_needed": "To strip multiline docstrings from Python code.", "key_assertions": [...]}}}

Confidence: 80%

Tokens: 97 input + 173 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/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



AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripDocstrings::test_strip_triple_double_quoted_docstring

Why Needed: To ensure that the context manager strips all types of docstrings, not just triple double-quoted ones.

Key Assertions:

- {'name': 'docstring removal', 'description': 'The function should remove all docstrings, including triple double-quoted ones.'}
- {'name': 'no exception raised', 'description': 'If a docstring is found, the function should not raise an exception.'}

Confidence: 80%

Tokens: 106 input + 131 output = 237 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestStripDocstrings::test_strip_triple_single_quoted_docstring

1ms

3

AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripDocstrings::test_strip_triple_single_quoted_docstring

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

Key Assertions:

- {'name': 'docstring removal', 'expected': 'def foo():\n pass'}

Confidence: 80%

Tokens: 106 input + 89 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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



AI ASSESSMENT

Scenario: Test the parsing of edge cases for Gemini models with different configurations.**Why Needed:** This test prevents a regression where the gemini model parser fails to parse models when the 'm1, m2' configuration is used.**Key Assertions:**

- assert 'm1' in models
- assert 'm2' in models
- assert provider._parse_preferred_models() == [] when config.model == None
- assert provider._parse_preferred_models() == [] when config.model == 'All'

Confidence: 80%**Tokens:** 273 input + 121 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	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	19 lines (ranges: 134-135, 137-141, 143-144, 476, 478, 524-531)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_coverage_boosters.py::TestCoverageBoosters::test_gemini_rate_limiter_edge_math

1ms

3

AI ASSESSMENT

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

Why Needed: This test prevents a potential bug where the rate limiter allows excessive token usage without penalizing requests.

Key Assertions:

- The next_available_in method should return a value greater than 0 if there are available tokens.
- The next_available_in method should return 0 if both limits have been reached.
- The record_tokens method should not increment the request count.

Confidence: 80%

Tokens: 273 input + 114 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/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` and `RunMeta` objects returns the expected values.

Why Needed: This test prevents regression in coverage booster models where the `coverage_percent` or `duration` fields are not being correctly converted to dictionary keys.

Key Assertions:

- The `coverage_percent` field of `SourceCoverageEntry` is equal to 50.0 when converted to a dictionary key.
- The `error` field of `LlmAnnotation` is equal to 'timeout' when converted to a dictionary key.
- The `duration` field of `RunMeta` is equal to 1.0 when converted to a dictionary key.

Confidence: 80%

Tokens: 318 input + 156 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/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:** To test the initialization of the Mapper with a given configuration.**Key Assertions:**

- {'name': 'mapper.config', 'expected_type': 'Config', 'actual_type': 'Config', 'message': ''}

Confidence: 80%**Tokens:** 109 input + 76 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/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 that the `get_warnings` method returns a list of warnings as expected.**Key Assertions:**

- {'name': 'assert isinstance(warnings, list)', 'expected_result': [1, 2, 3], "# Expected to be a list of warnings (e.g., from a file or variable with an error message) but actually returns something else (e.g., an empty list or a different type of value). This test ensures that the `get_warnings` method correctly identifies and returns warnings as expected in this scenario. If it doesn't, the test will fail and provide additional context for where the issue is occurring.": 'message_type', 'Test failed: TestCoverageMapper::test_get_warnings': 'This test has failed because the `get_warnings()` method of the CoverageMapper class does not return a list of warnings as expected. The actual output may vary depending on the configuration and environment, but it should contain at least one warning or an empty list.'}

Confidence: 80%**Tokens:** 110 input + 239 output = 349 total

COVERAGE

src/pytest_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

1ms



AI ASSESSMENT

Scenario: Test that the `map_coverage` method returns an empty dictionary when no coverage file is found.

Why Needed: Prevents a regression where the test fails to report any coverage information when there are no coverage files.

Key Assertions:

- The `mapper.map_coverage()` function should return an empty dictionary.
- The returned dictionary should be empty.
- There should be at least one warning in the `mapper.warnings` list.

Confidence: 80%

Tokens: 277 input + 104 output = 381 total

COVERAGE

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

PASSED

tests/test_coverage_map.py::TestCoverageMapperContextExtraction::test_extract_nodeid_all_phases

1ms



4

AI ASSESSMENT

Scenario: The test verifies that the `CoverageMapper` correctly extracts node IDs for all phases when `include_phase=all`.

Why Needed: This test prevents a regression where the coverage map might not include all phases if `include_phase=all` is used.

Key Assertions:

- When including phase 'all', the mapper should extract node IDs for all specified phases.
- The mapper should return the same node ID for each phase when included in 'all'.
- If `include_phase=all`, the mapper should not include any phase-specific node IDs in the coverage map.

Confidence: 80%

Tokens: 279 input + 131 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	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



AI ASSESSMENT

Scenario:

tests/test_coverage_map.py::TestCoverageMapperContextExtraction::test_extract_nodeid_filters_se

Why Needed: To ensure that the test does not filter out setup phase when including a run phase.

Key Assertions:

- {'name': "nodeid extraction should be performed for nodeid in 'setup' phase", 'expected_result': 'None'}

Confidence: 80%

Tokens: 139 input + 92 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	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|run'}

Confidence: 80%

Tokens: 145 input + 88 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/coverage_map.py	11 lines (ranges: 44-45, 216, 220, 224-225, 228-229, 231, 233, 236)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

 tests/test_coverage_map_coverage.py

17 tests

PASSED

tests/test_coverage_map_coverage.py::TestExtractContexts::test_contexts_by_lineno_exception

1ms



5

AI ASSESSMENT

Scenario: Test 'test_contexts_by_lineno_exception' verifies that the test_contexts_by_lineno function handles exceptions correctly when calling it on mock data.

Why Needed: This test prevents a regression where the test_contexts_by_lineno function fails to handle an exception raised by its context side effect.

Key Assertions:

- The function should not raise an exception when called with mock_data that has contexts.
- The function should return an empty dictionary when called with mock_data that does not have contexts.
- The function should increment the call count correctly when calling it on mock_data that raises an exception.
- The function should only be called once when calling it on mock_data that has contexts.
- The function should handle the exception gracefully and return a valid result.
- The function's context side effect should not affect its behavior when called with mock_data that does not have contexts.

Confidence: 80%

Tokens: 332 input + 193 output = 525 total

COVERAGE

src/pytest_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 no measured files are found.

Key Assertions:

- {'expected_value': {}, 'actual_value': {'__len__': 0}}

Confidence: 80%

Tokens: 136 input + 80 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/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 1ms 5

AI ASSESSMENT

Scenario:

tests/test_coverage_map_coverage.py::TestExtractContexts::test_skip_non_python_files

Why Needed: To ensure that non-Python files are skipped from coverage reports.

Key Assertions:

- {'assertion': "mock_data.measured_files.return_value == ['file.txt', 'data.json']", 'expected_result': ['file.txt', 'data.json'], 'message': "Expected mock_data.measured_files.return_value to be equal to ['file.txt', 'data.json']"}

Confidence: 80%

Tokens: 154 input + 120 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_coverage.py::TestLoadCoverageData::test_coverage_not_installed

1ms



AI ASSESSMENT

Scenario: Test Load Coverage Data

Why Needed: Coverage.py is required for this test.

Key Assertions:

- {'name': 'coverage.py is installed', 'description': 'The coverage.py module should be present in the environment.'}
- {'name': 'coverage.py is not installed', 'description': 'The coverage.py module should not be present in the environment.'}

Confidence: 80%

Tokens: 166 input + 98 output = 264 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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:

- {'description': 'The function should return None.', 'expected_result': 'None'}
- {'description': "The warnings should have a message containing 'W001'.", 'expected_result': ['W001']}

Confidence: 80%

Tokens: 153 input + 92 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/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



AI ASSESSMENT

Scenario: Test the exception handling feature when analyzing an exception.

Why Needed: This test prevents regression by ensuring that analysis2 raises an exception and is properly handled, preventing potential warnings from being added to the coverage report.

Key Assertions:

- The `map_source_coverage` function should return an empty list when analyzing an exception.
- A warning with the message 'COVERAGE_ANALYSIS_FAILED' should be added to the coverage report.

Confidence: 80%

Tokens: 286 input + 102 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/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_employee_statements

1ms



AI ASSESSMENT

Scenario: tests/test_coverage_map_coverage

Why Needed: To test the coverage map when a file has no statements.

Key Assertions:

- {'assertion_type': 'Equal to', 'expected_value': [], 'actual_value': []}

Confidence: 80%

Tokens: 178 input + 67 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/coverage_map.py	18 lines (ranges: 44-45, 243-244, 246-248, 250, 252-254, 259-261, 273-274, 299-300)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED

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

AI ASSESSMENT

Scenario: Test that test files are included when omit_tests_from_coverage is False.**Why Needed:** This test prevents a regression where test coverage is not reported for test files when omitting tests from coverage is enabled.**Key Assertions:**

- The `map_source_coverage` method returns a list with exactly one element.
- The `covered` attribute of the first element in the list is set to 2 (indicating that two lines were covered).
- The `missed` attribute of the first element in the list is set to 1 (indicating that one line was missed).

Confidence: 80%**Tokens:** 322 input + 136 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/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

2ms



AI ASSESSMENT

Scenario: {'id':

```
'tests/test_coverage_map_coverage.py::TestMapSourceCoverage::test_skip_non_python_files',
'description': 'Test case for test_skip_non_python_files'}
```

Why Needed: {'reason': 'Ensure that non-Python files are skipped from coverage reports.', 'description': 'This test is necessary to ensure that non-Python files are excluded from coverage reports.'}

Key Assertions:

- {'name': 'expected_result', 'type': 'list', 'description': 'Expected result of the test'}

Confidence: 80%

Tokens: 154 input + 173 output = 327 total

COVERAGE

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

PASSED

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

AI ASSESSMENT

Scenario: Test Map Source Coverage

Why Needed: To ensure that test files are skipped when omit_tests_from_coverage is True.

Key Assertions:

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

Confidence: 80%

Tokens: 182 input + 59 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/coverage_map.py	15 lines (ranges: 44-45, 243-244, 246-248, 250, 252-255, 257, 299-300)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_all_phase_config

1ms



4

AI ASSESSMENT

Scenario: Test that all phases are accepted when configured.**Why Needed:** Prevents regression in case of phase filtering, where only specific phases should be covered.**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') == 'test_foo.py::test_bar'
- mapper._extract_nodeid('test_foo.py::test_bar|teardown') == 'test_foo.py::test_bar'

Confidence: 80%**Tokens:** 305 input + 134 output = 439 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: To handle empty string inputs and ensure correct output.

Key Assertions:

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

Confidence: 80%

Tokens: 115 input + 74 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/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: To ensure that the `extract_nodeid` method returns None when an invalid input (None) is provided.

Key Assertions:

- {'name': 'assert mapper._extract_nodeid(None) == None', 'description': 'The `_extract_nodeid` method should return None for a None input.'}

Confidence: 80%

Tokens: 114 input + 103 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/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 a regression where the test would fail due to an incorrect default configuration.

Key Assertions:

- The function _extract_nodeid should return the nodeid when the phase matches 'run'.
- The function _extract_nodeid should return None when the phase doesn't match 'run' or 'setup'.
- The function _extract_nodeid should return None when the phase doesn't match 'teardown'.

Confidence: 80%

Tokens: 297 input + 116 output = 413 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	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: Prevents a bug where the test would incorrectly filter out nodes in the setup phase due to incorrect configuration.

Key Assertions:

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

Confidence: 80%

Tokens: 293 input + 125 output = 418 total

COVERAGE

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

PASSED

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_teardown_phase_config

1ms



4

AI ASSESSMENT

Scenario: Test that teardown phase is correctly filtered when configured.**Why Needed:** 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 + 129 output = 425 total

COVERAGE

src/pytest_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 test the scenario where a node id does not contain a pipe delimiter.

Key Assertions:

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

Confidence: 80%

Tokens: 136 input + 84 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/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 that the `extract_contexts` method of `CoverageMapper` correctly extracts contexts for all paths in `extract_contexts` when coverage is enabled.

Why Needed: This test prevents a regression where the coverage map does not include all contexts for a specific file, potentially leading to incomplete or inaccurate coverage analysis.

Key Assertions:

- The `test_one` context should be present in the coverage report for `app.py`.
- The number of lines in `test_one` should match the expected count (2).
- Each line in `test_one` should have a unique file path that includes `app.py`.
- If `test_two` is not present, then `test_one` should be included in the coverage report.
- The context for `test_two` should not include any lines from `app.py`.
- Each line in `test_two` should have a unique file path that includes `app.py`.
- If `test_one` is not present, then `test_two` should be included in the coverage report.
- The context for `test_two` should include all lines from `app.py`.
- Each line in `test_two` should have a unique file path that includes `app.py`.
- If `test_one` is not present, then `test_two` should be included in the coverage report.
- The context for `test_two` should include all lines from `app.py`.
- Each line in `test_two` should have a unique file path that includes `app.py`.
-]

Confidence: 80%

Tokens: 413 input + 345 output = 758 total

COVERAGE

src/pytest_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)

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

src/pytest_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 the ability to extract node IDs from coverage reports when no phases are specified.

Why Needed: This test prevents regression in coverage reporting when certain phases are not included in the report.

Key Assertions:

- The function _extract_nodeid returns the correct node ID for a given phase.
- The function _extract_nodeid ignores the 'run' phase and its corresponding node ID.
- The function _extract_nodeid correctly handles coverage reports without any phases specified.
- The function _extract_nodeid ignores context files that do not match any phase.
- The function _extract_nodeid returns the correct node ID for a given phase, even when no phase is included in the report.

Confidence: 80%

Tokens: 323 input + 157 output = 480 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test that the test_load_coverage_data_no_files function correctly handles cases where no coverage files exist.

Why Needed: This test prevents a potential bug or regression by ensuring that the function does not silently fail when there are no coverage files.

Key Assertions:

- The function should return None for _load_coverage_data() when no .coverage files exist.
- The number of warnings should be exactly 1.
- The first warning code should be 'W001'.

Confidence: 80%

Tokens: 276 input + 111 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/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 1ms 4

AI ASSESSMENT

Scenario: Test should handle errors reading coverage files and prevent regression.

Why Needed: This test prevents a potential regression where the CoverageMapper class does not properly handle errors when loading coverage data from corrupted files.

Key Assertions:

- The function `mapper._load_coverage_data()` returns None if an error occurs while reading the coverage file.
- Any warnings generated by the mapper are marked as indicating that the coverage data failed to be read.
- The test verifies that any messages containing 'Failed to read coverage data' are present in the warnings list.
- The test ensures that the coverage module's `CoverageData` class is mocked with an exception when it raises an error during reading.
- The test verifies that the `CoverageMapper` instance returns None after attempting to load coverage data from a corrupted file.

Confidence: 80%

Tokens: 343 input + 178 output = 521 total

COVERAGE

src/pytest_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 2ms 4

AI ASSESSMENT

Scenario: Test should handle parallel coverage files from xdist and verify that it correctly updates the CoverageData instances.

Why Needed: This test prevents regression in handling parallel coverage files, ensuring that the CoverageMapperMaximal class can correctly update its internal data structures when dealing with such scenarios.

Key Assertions:

- The mock CoverageData instance returned by _load_coverage_data() should have been updated at least twice.
- The mock CoverageData instances should not be None after calling update.
- The mock CoverageData instances should be of the same type (CoverageData) before and after calling update.
- The mock CoverageData instances should have different attributes (update_count, data) before and after calling update.
- The _load_coverage_data() function should return a new instance of CoverageData or None if no coverage files are available.
- The _load_coverage_data() function should call the update method on the mock CoverageData instances with at least two calls.
- The _load_coverage_data() function should not raise an exception when called multiple times without any changes to the CoverageData instances.

Confidence: 80%

Tokens: 378 input + 236 output = 614 total

COVERAGE

src/pytest_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 the `map_coverage` method with `_load_coverage_data returning None.

Why Needed: Prevents a potential bug where an empty dictionary is returned when there's no coverage data.

Key Assertions:

- The `_load_coverage_data` method should return `None` when called without any arguments.
- The `map_coverage` method should return an empty dictionary when the input is `None` or an empty list.
- The `map_coverage` method should not raise an exception when given a non-empty list of coverage data.
- The `map_coverage` method should preserve the original order of metrics in case of duplicate values.
- The `_load_coverage_data` method should be able to handle cases where the input is a dictionary with no keys.
- The `map_coverage` method should not throw an exception when given a non-list value for coverage data.
- The `map_coverage` method should return the original list of metrics if it's already present in the result.

Confidence: 80%

Tokens: 228 input + 217 output = 445 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	5 lines (ranges: 44-45, 58-60)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_map_source_coverage_analysis_error

1ms



5

AI ASSESSMENT

Scenario: Test that the CoverageMapper handles analysis errors correctly.

Why Needed: This test prevents a potential regression where an error during coverage analysis would prevent the mapper from working as expected.

Key Assertions:

- The mock_cov.analysis2 should be called with an Exception exception.
- mock_data.measured_files.return_value should return ['app.py']
- mock_cov.get_data.return_value should raise a MagicMock exception.
- entries should not contain any files that have errors.
- len(entries) should equal 0 after skipping error files.

Confidence: 80%

Tokens: 274 input + 125 output = 399 total

COVERAGE

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

PASSED

tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_map_source_coverage_comprehensive 2ms 6

AI ASSESSMENT

Scenario: Verify that the test maps all source files in map_source_coverage with a comprehensive coverage percentage.

Why Needed: This test prevents regression by ensuring that all source files are covered, even if analysis2 is not.

Key Assertions:

- The file path of the mapped entry should be 'app.py'.
- The number of statements in the mapped entry should be 3.
- The coverage percentage of the mapped entry should be 66.67%.
- All files in the source directory should be covered by the test.
- At least one file in the source directory should not be missed by the test.
- The coverage percentage for each file in the source directory should be calculated correctly.

Confidence: 80%

Tokens: 345 input + 159 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/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 to ensure it correctly returns a WarningCode.W001_NO_COVERAGE instance with the expected message and detail.

Why Needed: To prevent unexpected warnings when no coverage files are found, this test verifies that the make_warning factory function returns an appropriate WarningCode.W001_NO_COVERAGE instance.

Key Assertions:

- The returned warning code is correct (WarningCode.W001_NO_COVERAGE).
- The message of the warning is 'No .coverage file found'.
- The detail of the warning is 'test-detail'.

Confidence: 80%

Tokens: 236 input + 129 output = 365 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: Test that warning codes have correct values.**Why Needed:** This test prevents a regression where the warning code values are incorrect, potentially causing issues with the application's functionality.**Key Assertions:**

- {'message': 'assert WarningCode.W001_NO_COVERAGE.value == "W001"', 'description': 'Correct value for W001_NO_COVERAGE'}
- {'message': 'assert WarningCode.W101_LLM_ENABLED.value == "W101"', 'description': 'Correct value for W101_LLM_ENABLED'}
- {'message': 'assert WarningCode.W201_OUTPUT_PATH_INVALID.value == "W201"', 'description': 'Correct value for W201'}
- {'message': 'assert WarningCode.W301_INVALID_CONFIG.value == "W301"', 'description': 'Correct value for W301'}
- {'message': 'assert WarningCode.W401_AGGREGATE_DIR_MISSING.value == "W401"', 'description': 'Correct value for W401'}

Confidence: 80%**Tokens:** 240 input + 206 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/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 potential warning that could be misleading or incorrect.**Key Assertions:**

- The 'detail' key is present in the dictionary and its value matches the expected string.
- The 'code' key matches the expected string.
- The 'message' key matches the expected string.
- The 'no_detail' key is not present in the dictionary or its value does not match the expected string.
- The 'W001_NO_COVERAGE' code matches the expected string.
- The 'W101_LLM_ENABLED' code matches the expected string.

Confidence: 80%**Tokens:** 276 input + 143 output = 419 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the `make_warning` function with known code.

Why Needed: Prevents a warning from being generated for known code that should not be warned about.

Key Assertions:

- The function `make_warning` returns an instance of `WarningCode.W101_LLM_ENABLED`.
- The message associated with the warning is set to `WARNING_MESSAGES[WarningCode.W101_LLM_ENABLED]`.
- The detail attribute of the warning is None.

Confidence: 80%

Tokens: 222 input + 109 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/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_unknown_code

1ms



3

AI ASSESSMENT

Scenario: tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_unknown_code

Why Needed: To test the functionality of using a fallback message for unknown code (if enum allowed it)

Key Assertions:

- {'name': 'missing_code', 'expected_value': 'WarningCode.W001_NO_COVERAGE'}

Confidence: 80%

Tokens: 202 input + 84 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/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_with_detail

1ms



3

AI ASSESSMENT

Scenario: tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_with_detail

Why Needed: To test that a warning is created with the correct code and detail.

Key Assertions:

- {'name': 'code', 'value': 'WarningCode.W301_INVALID_CONFIG'}
- {'name': 'detail', 'value': 'Bad value'}

Confidence: 80%

Tokens: 127 input + 86 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/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_errors_maximal.py::TestWarningCodes::test_codes_are_strings

1ms

2

AI ASSESSMENT

Scenario: tests/test_errors_maximal.py::TestWarningCodes::test_codes_are_strings

Why Needed: The test is checking if the WarningCode enum values are strings.

Key Assertions:

- {'name': 'assert isinstance(code.value, str)', 'expected_result': 'True'}
- {'name': "assert code.value.startswith('W')", 'expected_result': 'True'}

Confidence: 80%

Tokens: 109 input + 101 output = 210 total

COVERAGE

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

PASSED

tests/test_errors_maximal.py::TestWarningDataClass::test_warning_to_dict_no_detail 1ms 3

AI ASSESSMENT

Scenario: Tests for error handling and serialization of Warning objects.

Why Needed: The test is necessary to ensure that the warning object can be serialized correctly without any additional details.

Key Assertions:

- {'name': 'data', 'expected_value': {'code': 'W001', 'message': 'No coverage'}, 'actual_value': {'code': 'W001', 'message': 'No coverage'}}}
- {'name': 'type', 'expected_value': 'dict', 'actual_value': 'dict'}

Confidence: 80%

Tokens: 147 input + 133 output = 280 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_errors_maximal.py::TestWarningDataClass::test_warning_to_dict_with_detail 1ms 3

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	8 lines (ranges: 70-71, 73-75, 77-79)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

 tests/test_fs.py

12 tests

PASSED

tests/test_fs.py::TestIsPythonFile::test_non_python_file

1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestIsPythonFile::test_non_python_file**Why Needed:** This test ensures that the `is_python_file` function correctly identifies non-.py files.**Key Assertions:**

- {'name': 'Expected result for foo/bar.txt', 'type': 'assertion', 'value': 'False'}
- {'name': 'Expected result for foo/bar.pyc', 'type': 'assertion', 'value': 'False'}

Confidence: 80%**Tokens:** 115 input + 119 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	1 lines (ranges: 79)

PASSED

tests/test_fs.py::TestIsPythonFile::test_python_file

1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestIsPythonFile::test_python_file**Why Needed:** The function `is_python_file` should be able to identify `.py` files.**Key Assertions:**

- {'message': 'Expected the function to return True for .py files', 'type': 'assertion'}
- {'message': 'The function is returning False for foo/bar.py', 'type': 'expected_result'}

Confidence: 80%**Tokens:** 98 input + 99 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/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 makes a path relative to the test directory.**Key Assertions:**

- {'assertion_type': 'path', 'expected_result': '/subdir/file.py', 'actual_result': 'subdir/file.py'}

Confidence: 80%**Tokens:** 144 input + 92 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/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  3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestNormalizePath::test_already_normalized**Why Needed:** To ensure that 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'}

Confidence: 80%**Tokens:** 96 input + 81 output = 177 total

COVERAGE

src/pytest_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:** strips trailing slash from path**Key Assertions:**

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

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	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 ensure that custom patterns are correctly excluded from the test directory.**Key Assertions:**

- {'name': 'should_skip_path', 'expected_result': True}
- {'name': 'assert', 'message': "Should return True for should_skip_path('tests/conftest.py', exclude_patterns=['test*'])"}
- {'name': 'assert', 'message': "Should return False for should_skip_path('src/module.py', exclude_patterns=['test*'])"}

Confidence: 80%**Tokens:** 126 input + 143 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/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



AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_normal_path**Why Needed:** The test is checking if the `should_skip_path` function correctly handles normal file system paths.**Key Assertions:**

- {'name': 'assert should_skip_path returns False for src/module.py', 'expected_value': False, 'actual_value': 'tests/test_fs.py::TestShouldSkipPath::test_normal_path'}

Confidence: 80%**Tokens:** 96 input + 103 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/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



AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_skips_git**Why Needed:** The current implementation of `should_skip_path` does not correctly handle Git directories.**Key Assertions:**

- {'name': 'assert should_skip_path returns True for .git/objects/foo', 'expected': True, 'actual': 'True'}

Confidence: 80%**Tokens:** 99 input + 89 output = 188 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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



AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_skips_pycache**Why Needed:** This test case is needed because the `should_skip_path` function does not currently handle `__pycache__` directories.**Key Assertions:**

- `{'name': 'assert should_skip_path returns True for __pycache__/bar.pyc', 'expected_result': True}`

Confidence: 80%**Tokens:** 109 input + 94 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/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 is checking if the `should_skip_path` function correctly identifies venv directories as being to be skipped.**Key Assertions:**

- {'name': 'assert should_skip_path returns True for venv directory', 'description': 'Expected the `should_skip_path` function to return True when given a path that is a venv directory.', 'value': True}
- {'name': 'assert should_skip_path returns True for .venv directory', 'description': 'Expected the `should_skip_path` function to return True when given a path that is a .venv directory.', 'value': True}

Confidence: 80%**Tokens:** 121 input + 171 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

 tests/test_fs_coverage.py

15 tests

PASSED

tests/test_fs_coverage.py::TestIsPythonFile::test_is_python_file_false

1ms



AI ASSESSMENT

Scenario: Testing that non-.py files do not return True.

Why Needed: Prevents a false positive assertion in the `is_python_file` method where it incorrectly returns True for non-python files.

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 + 97 output = 307 total

COVERAGE

src/pytest_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: Verifies that a module file (.py) returns True.

Why Needed: Prevents the test from incorrectly identifying non-python files as 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 + 91 output = 303 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	1 lines (ranges: 79)

PASSED

tests/test_fs_coverage.py::TestMakeRelative::test_make_relative_path_not_under_base

1ms

3

AI ASSESSMENT

Scenario: Test makes a relative path not under the base directory.

Why Needed: Prevents regression when make_relative is called with paths that are not relative to the base directory.

Key Assertions:

- The test verifies that make_relative returns a normalized absolute path even if the input path is not relative to the base.
- The test verifies that the 'project1' and 'file.py' strings are present in the returned path.
- The test verifies that the result of make_relative does not contain any '..' or '..\'
- The test verifies that the result of make_relative is a string (not a file object) as expected.
- The test verifies that the parent directory of the input paths is preserved in the output path.
- The test verifies that the relative_to parameter is not used to resolve the issue.
- The test verifies that the make_relative function behaves correctly when given unrelated paths.

Confidence: 80%

Tokens: 301 input + 203 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/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 correctly.

Key Assertions:

- {'expected_result': 'subdir/file.py', 'actual_result': 'subdir/file.py'}

Confidence: 80%

Tokens: 147 input + 63 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	11 lines (ranges: 30, 33, 36, 39, 42, 55, 58-60, 63-64)

PASSED

tests/test_fs_coverage.py::TestMakeRelative::test_make_relative_with_none_base

1ms

3

AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestMakeRelative::test_make_relative_with_none_base

Why Needed: The test is necessary because the `make_relative` function does not handle cases where the base path is None correctly.

Key Assertions:

- {'message': "Expected result to be 'path/to/file.py'", 'expected_result': 'path/to/file.py'}

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/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 file path object is necessary to ensure correct behavior in certain scenarios.

Key Assertions:

- {'expected_value': 'path/to/file.py', 'actual_value': "normalize_path(Path('path/to/file.py'))"}

Confidence: 80%

Tokens: 110 input + 86 output = 196 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	5 lines (ranges: 30, 33, 36, 39, 42)

PASSED

tests/test_fs_coverage.py::TestNormalizePath::test_normalize_path_trailing_slash

1ms



AI ASSESSMENT

Scenario: tests/test_fs_coverage.py::TestNormalizePath::test_normalize_path_trailing_slash

Why Needed: To ensure that the `normalize_path` function correctly removes trailing slashes from file paths.

Key Assertions:

- {'name': 'assert result is equal to expected value', 'expected_value': 'path/to/dir', 'actual_value': 'path/to/dir'}

Confidence: 80%

Tokens: 111 input + 95 output = 206 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	5 lines (ranges: 30, 33, 36, 39, 42)

PASSED

tests/test_fs_coverage.py::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:

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

Confidence: 80%

Tokens: 120 input + 91 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/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: The test should skip .git directories because they contain Git hooks that can interfere with the file system coverage.

Key Assertions:

- {'name': 'should be True', 'description': 'The function should return True when a .git directory is found.'}

Confidence: 80%

Tokens: 102 input + 91 output = 193 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED

tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_path_starting_with_skip_dir

1ms

3

AI ASSESSMENT

Scenario:

tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_path_starting_with_skip_dir

Why Needed: To ensure that the function correctly handles paths starting with a skip directory name.

Key Assertions:

- {'message': 'should be True', 'expected_value': True}
- {'message': '.venv', 'expected_value': True}

Confidence: 80%

Tokens: 124 input + 88 output = 212 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED

tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_pycache

1ms

3

AI ASSESSMENT

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

Why Needed: Because the test module __pycache__ is being tested.

Key Assertions:

- {'path': 'src/__pycache__/module.cpython-312.pyc'}

Confidence: 80%

Tokens: 116 input + 75 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	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: /usr/lib/python3.12/site-packages/pkg/mod.py

Why Needed: The test is needed because it checks for site-packages directories, which are not covered by the current implementation.

Key Assertions:

- {'assertion': "should_skip_path('/usr/lib/python3.12/site-packages/pkg/mod.py')", 'expected_result': 'True'}

Confidence: 80%

Tokens: 111 input + 91 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/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: Because the 'venv' directory contains a Python package (site.py) which should be skipped.

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 + 113 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	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED

tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_with_exclude_patterns 2ms 3

AI ASSESSMENT

Scenario:

tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_with_exclude_patterns

Why Needed: Custom exclude patterns are needed to skip certain paths.

Key Assertions:

- {'assertion': 'should_skip_path is called with the correct arguments', 'expected_result': 'True'}
- {'assertion': 'should_skip_path returns False for a path that matches an exclude pattern', 'expected_result': 'False'}

Confidence: 80%

Tokens: 132 input + 111 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)

 tests/test_gemini_provider.py

25 tests

PASSED

`tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_annotate_loop_daily_limit_hit` 3ms 4

AI ASSESSMENT

Scenario: Test that the `test_annotate_loop_daily_limit_hit` function prevents a regression where the daily limit is hit and the Gemini API returns an error.

Why Needed: This test prevents the regression where the daily limit is hit, causing the Gemini API to return an error when trying to annotate internal nodes with the 'src' nodeid.

Key Assertions:

- assert 'Gemini requests-per-day limit reached' in res.error
- assert mock_limiter.next_available_in.return_value == None
- assert provider._rate_limiters['m1'].next_available_in.return_value == None
- assert mock_limiter.next_available_in.return_value == None
- assert provider._rate_limiters['m1'].next_available_in.return_value == None
- assert mock_limiter.next_available_in.return_value is None
- assert mock_limiter.next_available_in.return_value == None

Confidence: 80%

Tokens: 367 input + 197 output = 564 total

COVERAGE

src/pytest_llm_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: Tests the coverage of exceptions when using GeminiProvider with a mock configuration.

Why Needed: This test prevents regression that occurs when using GeminiProvider with a mock configuration and encountering exceptions, such as generation failures or rate limit exceeded errors.

Key Assertions:

- Mocking the _GeminiRateLimiter to prevent RateLimitExceeded RPD (Line 300)
- _CallGemini to raise _GeminiRateLimitExceeded when RateLimitExceeded RPD is encountered
- Mocking the _ModelExhaustedAt dictionary to track model exhaustion and return an error message
- Asserting that exceptions are correctly propagated through the provider's methods
- Verifying that the correct exception messages are returned in the error object

Confidence: 80%

Tokens: 730 input + 166 output = 896 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	22 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	100 lines (ranges: 32-34, 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-210, 221-224, 228-230, 232-233, 235-236, 239-244, 263-265, 268, 293, 295, 299-303, 318-320, 340, 343, 471-473)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_coverage_gaps

167ms



AI ASSESSMENT

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

Why Needed: This test prevents a potential regression where the GeminiProvider's rate limiting and annotation logic may not be properly implemented, leading to uncovered coverage gaps.

Key Assertions:

- The `_rate_limiters` dictionary should contain '`m1`' with a value of 100.
- `_parse_rate_limits` method returns a dictionary with the correct requests per day limit (100).
- The `provider._models` attribute contains the expected list of fallback models ('`fallback`').

Confidence: 80%

Tokens: 821 input + 125 output = 946 total

COVERAGE

src/pytest_llm_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)

PASSED

tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_parse_p
referred_models_coverage

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

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_da
ily_requests

1ms



3

AI ASSESSMENT

Scenario: TestGeminiProvider**Why Needed:** To test the coverage gaps of Gemini provider after pruning daily requests.**Key Assertions:**

- {'name': 'assert len(_GeminiRateLimiter._daily_requests) == 0', 'description': 'The length of _GeminiRateLimiter._daily_requests should be 0 after pruning.'}

Confidence: 80%**Tokens:** 157 input + 90 output = 247 total

COVERAGE

src/pytest_llm_report/collector.py

14 lines (ranges: 90, 93, 96,
99, 110-112, 114-115, 124,
127, 140, 209-210)

src/pytest_llm_report/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

2ms



AI ASSESSMENT

Scenario: Test verifies that the GeminiRateLimiter does not allow excessive token usage to prevent TPM fallback.

Why Needed: This test prevents regression in case of extremely high token usage, causing the GeminiRateLimiter to return 0.0 at line 106/108 when a huge request is made.

Key Assertions:

- The total tokens used should be less than or equal to the limit (100).
- If `request_tokens` exceeds the remaining tokens, the loop should not return.
- The `remaining` variable should decrease as the tokens are used.
- The GeminiRateLimiter should not allow a huge token usage that implies it cannot fit all requests.
- If `request_tokens` is massive and `limit=100`, the test should pass without returning 0.0 at line 106/108.

Confidence: 80%

Tokens: 524 input + 186 output = 710 total

COVERAGE

src/pytest_llm_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 forced import error is annotated correctly when google-generativeai is not installed.

Why Needed: This test prevents a potential regression where the GeminiProvider does not correctly handle import errors due to missing dependencies.

Key Assertions:

- The annotation should contain an error message indicating that google-generativeai was not installed.
- The error message should include the exact string 'google-generativeai not installed'.
- The annotation should be able to distinguish between a forced import error and a normal import operation.
- The annotation should not report any other errors when the dependency is installed.
- The annotation should provide a clear indication of why the annotation failed (in this case, due to missing dependencies).
- The annotation should include all necessary information about the test environment (e.g., module flagging, test result).

Confidence: 80%

Tokens: 259 input + 186 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	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 annotation when token is missing.**Why Needed:** To prevent a potential bug where the Gemini API token is not set, and thus the annotation fails.**Key Assertions:**

- The 'GEMINI_API_TOKEN' key should be present in the annotation error message.
- The 'GEMINI_API_TOKEN' key should contain the value 'None'.
- The 'GEMINI_API_TOKEN' key should not be empty.
- The 'GEMINI_API_TOKEN' key should have a non-empty string value.
- The 'GEMINI_API_TOKEN' key should not be None.
- The 'GEMINI_API_TOKEN' key should contain the correct type (str).
- The 'GEMINI_API_TOKEN' key should not be an empty string.
- The 'GEMINI_API_TOKEN' key should have a non-empty string value.
-

Confidence: 80%**Tokens:** 313 input + 203 output = 516 total

COVERAGE

src/pytest_llm_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 annotates a rate limit retry scenario correctly.**Why Needed:** This test prevents regression in the GeminiProvider's ability to handle rate limit retries.**Key Assertions:**

- The annotation returned by the provider matches the expected scenario of 'Recovered Scenario'.
- The mock post call count is correct, indicating that the retry mechanism was successful.
- The _parse_response method returns a Mock object with the correct scenario and error.
- The provider's internal annotation logic does not modify the test result nodeid.
- The provider's internal annotation logic does not change the expected outcome of the test.
- The provider's internal annotation logic does not add any new assertions or checks.
- The provider's internal annotation logic does not remove any existing assertions or checks.
- The provider's internal annotation logic does not modify the mock post call count.

Confidence: 80%**Tokens:** 636 input + 196 output = 832 total

COVERAGE

src/pytest_llm_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 392ms 4

AI ASSESSMENT

Scenario: Test that the GeminiProvider correctly annotates a success scenario with the correct key assertions.

Why Needed: This test prevents regression where the provider incorrectly returns an error when annotating a successful response.

Key Assertions:

- The annotation returned by `_annotate_internal` has the expected scenario.
- The annotation does not have any errors.
- The annotation correctly extracts the 'text' part from the response.
- The annotation correctly extracts the 'tokens' part from the response.
- The annotation returns a valid LlmAnnotation object with the correct key assertions.
- The annotation does not return an error when the response is in the expected format.
- The annotation calls `_parse_response` correctly to extract the scenario and tokens.
-

Confidence: 80%

Tokens: 649 input + 170 output = 819 total

COVERAGE

src/pytest_llm_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-

src/pytest_llm_report/plugin.py

6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_gemini_provider.py::TestGeminiProvider::test_availability 2ms  5

AI ASSESSMENT

Scenario: Tests the availability of the Gemini provider when no API token is provided.**Why Needed:** Prevents a bug where the provider tries to access an unavailable environment.**Key Assertions:**

- The provider should return False indicating that the environment is not available.
- The provider should not throw any exceptions if the environment is already available.
- The provider's _check_availability method should be able to detect when the environment is not available.
- The provider's _check_availability method should not make any assumptions about the availability of the environment based on the API token.
- The provider's _check_availability method should only return False if the environment is truly unavailable.
- The provider's _check_availability method should be able to handle cases where the environment has been previously checked and found available.
- The provider's _check_availability method should not interfere with other tests that rely on the environment being available.

Confidence: 80%**Tokens:** 235 input + 203 output = 438 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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: Verify that the GeminiProvider class correctly annotates retry exceptions and updates model exhaustion states accordingly.

Why Needed: This test prevents regression where the GeminiProvider class fails to properly handle retry exceptions and update model exhaustion states when encountering a resource exhausted error.

Key Assertions:

- The `provider._model_exhausted_at` dictionary is updated correctly with the model ID 'm1' after an exception occurs.
- The `provider._cooldowns` dictionary contains the expected cooldown value for model 'm1'.
- The cooldown value for model 'm1' exceeds 5.5 seconds as expected.

Confidence: 80%

Tokens: 651 input + 139 output = 790 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	22 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	111 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-210, 221-224, 228-230, 232-233, 235-237, 239-244, 263-265, 268, 272-276, 279-281, 283-286, 288-292, 318-320, 322-323, 340, 343, 471-473)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_annotation_retry_loop_coverage

3ms



AI ASSESSMENT

Scenario: Test that the `GeminiProvider` ensures models are exhausted after a retry loop and coverage is maintained.

Why Needed: This test prevents regression where the Gemini API token limit exceeds the daily limit of 24 hours ago, causing the model to be exhausted without sufficient retries.

Key Assertions:

- The `'_model_exhausted_at` attribute is updated correctly after a retry loop.
- The `GeminiProvider` ensures that models are exhausted within the specified time frame (24h+).
- The test maintains coverage by ensuring that the model's exhaustion check passes even when the Gemini API token limit exceeds the daily limit.

Confidence: 80%

Tokens: 482 input + 144 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/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::test_ensure_rate_limits_error

Why Needed: This test ensures that the `GeminiProvider` raises an error when rate limiting is attempted with a non-integer value.

Key Assertions:

- {'name': 'Mocked config has rpm=10', 'expected_value': 10, 'actual_value': 0}
- {'name': 'limits.requests_per_minute == 10', 'expected_value': 10, 'actual_value': 0}

Confidence: 80%

Tokens: 156 input + 125 output = 281 total

COVERAGE

src/pytest_llm_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 an exception**Why Needed:** To test the error handling of GeminiProvider when a network error occurs**Key Assertions:**

- {'name': 'models are empty', 'description': 'The list of available models should be empty after a network error.', 'expected_result': [], 'actual_result': 'models == []'}
- {'name': 'limit_map is empty', 'description': 'The limit map should be empty after a network error.', 'expected_result': {}, 'actual_result': 'limit_map == {}'}

Confidence: 80%**Tokens:** 132 input + 139 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/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: Test that a fetch of available models with invalid JSON returns the expected result.**Why Needed:** Prevents a potential bug where an incorrect or malformed JSON is returned, potentially causing unexpected behavior or errors.**Key Assertions:**

- models containing non-list supportedGenerationMethods
- models not containing inputTokenLimit
- model m3 in limit_map
- model m1 and model m2 not in models
- model m3 in limit_map

Confidence: 80%**Tokens:** 340 input + 109 output = 449 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	34 lines (ranges: 134-135, 137-141, 143-144, 476-477, 537, 539-543, 547-548, 550-559, 562-563, 567, 569, 574)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_get_max_context_tokens_calls_ensure 2ms 4

AI ASSESSMENT

Scenario:

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_get_max_context_tokens_calls_ensu

Why Needed: To ensure that the `get_max_context_tokens` method of the GeminiProvider class calls the mock function `mock_ensure` when necessary.

Key Assertions:

- {'name': 'mock_ensure is called once', 'description': 'The `mock_ensure` function should be called exactly once during the test.', 'condition': 'mock_ensure.call_count == 1'}

Confidence: 80%

Tokens: 144 input + 124 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	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: TestGeminiProviderDetailed::test_parse_rate_limits_types**Why Needed:** To ensure that the GeminiProvider can correctly parse rate limits and return valid configuration values.**Key Assertions:**

- {'name': 'config.requests_per_minute is None', 'expected_value': 0, 'actual_value': 'None'}
- {'name': 'config.tokens_per_minute == 100', 'expected_value': 100, 'actual_value': 100}

Confidence: 80%**Tokens:** 156 input + 117 output = 273 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	24 lines (ranges: 134-135, 137-141, 143-144, 449-457, 459-460, 463-466)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiRateLimiter::test_prune_logic

1ms



3

AI ASSESSMENT

Scenario: Verify that the _prune method of _GeminiRateLimiter removes all old requests and updates token usage accordingly.

Why Needed: This test prevents a potential regression where the rate limiter would not prune older requests, leading to incorrect usage of tokens.

Key Assertions:

- The length of _request_times should be 1 after pruning.
- The length of _token_usage should be 1 after pruning.
- _request_times[0] should equal now - 10.0 after pruning.

Confidence: 80%

Tokens: 323 input + 119 output = 442 total

COVERAGE

src/pytest_llm_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



AI ASSESSMENT

Scenario: Tests for the Gemini Rate Limiter**Why Needed:** The test is necessary to ensure that the rate limiter correctly handles invalid token records.**Key Assertions:**

- {'name': 'Limiter should not store any tokens when record_tokens is called with a negative number', 'expected_result': 0, 'actual_result': {'scenario': 'Tests for the Gemini Rate Limiter', 'why_needed': 'The test is necessary to ensure that the rate limiter correctly handles invalid token records.', 'key_assertions': [...]}}

Confidence: 80%**Tokens:** 127 input + 129 output = 256 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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 with a single request per day

Why Needed: The test ensures that the rate limiting feature correctly limits requests to 1 per day.

Key Assertions:

- {'name': 'Limiter is initialized with correct rate limit', 'description': 'The limiter should be initialized with a rate limit of 1 requests per day', 'expected_value': 1, 'actual_value': 0}
- {'name': 'Rate limiting feature works as expected for single request per day', 'description': 'The rate limiting feature should not allow more than one request per day', 'expected_value': 0, 'actual_value': 100}

Confidence: 80%

Tokens: 129 input + 164 output = 293 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 RPM limit is correctly enforced for the first two requests and that it's not exceeded.

Why Needed: This test prevents a potential issue where the rate limiter exceeds the allowed number of requests per minute, potentially leading to unexpected behavior or errors in downstream applications.

Key Assertions:

- limiter.next_available_in(100) == 0.0
- limiter.record_request()
- assert limiter.next_available_in(100) == 0.0
- limiter.record_request()
- wait > 0 and wait <= 60.0

Confidence: 80%

Tokens: 280 input + 140 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/llm/gemini.py	27 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-97, 100-102)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiRateLimiter::test_seconds_until_tpm_available_branches

1ms

3

AI ASSESSMENT

Scenario: Verify that the rate limiter returns 0 seconds when no tokens are requested and within the limit.

Why Needed: This test prevents a potential regression where the rate limiter does not return 0 seconds for requests with no tokens.

Key Assertions:

- The function _seconds_until_tpm_available should return 0.0 when no tokens are requested.
- The function _seconds_until_tpm_available should return 0.0 when more than limit tokens are requested but empty usage.
- The function _seconds_until_tpm_available should return 0.0 for normal usage within the limit.
- The function _seconds_until_tpm_available should return a value less than or equal to 60 seconds when usage exceeds the limit by 1 second.

Confidence: 80%

Tokens: 377 input + 170 output = 547 total

COVERAGE

src/pytest_llm_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: Test that waiting for a daily limit exceeded throws an exception when the rate limit is set to one request per day.

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

Key Assertions:

- The `wait_for_slot` method raises `_GeminiRateLimitExceeded` with a `limit_type` of 'requests_per_day'.
- The `limit_type` attribute of the raised exception matches the expected value.
- `_GeminiRateLimitExceeded` has a `limit_type` attribute that is set to 'requests_per_day' when it is raised.
- The `value` attribute of the raised exception contains an object with a `limit_type` attribute set to 'requests_per_day'.
- The `limit_type` value matches the expected value for requests per day.
- `_GeminiRateLimitExceeded` raises an exception when the daily limit is exceeded, which prevents unexpected behavior in downstream applications.

Confidence: 80%

Tokens: 263 input + 228 output = 491 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	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 1ms 3

AI ASSESSMENT

Scenario: Test that the wait_for_slot function sleeps for a sufficient amount of time when the next available slot is not immediately available.

Why Needed: This test prevents a potential regression where the rate limiter does not sleep long enough between requests, potentially leading to performance issues or unexpected behavior.

Key Assertions:

- The wait_for_slot function sleeps for at least 10 seconds.
- The next_available_in method is called with an argument of 10.0.
- The mock_sleep assertion is called once with a value of 10.0.

Confidence: 80%

Tokens: 325 input + 126 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/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: To ensure that different configurations of the Compute API produce different hashes, which can be used to identify and fix potential issues.

Key Assertions:

- {'name': 'config1 and config2 are different', 'expected': 'different', 'actual': 'not equal'}

Confidence: 80%

Tokens: 119 input + 94 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/util/hashing.py	8 lines (ranges: 96-101, 103-104)

PASSED

tests/test_hashing.py::TestComputeConfigHash::test_returns_short_hash

1ms



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 efficiently in a database or other storage system.**Key Assertions:**

- {'assertion': 'h should have length 16', 'expected_result': 16}
- {'assertion': 'h should not contain any non-ASCII characters', 'expected_result': 'All characters in h are ASCII'}

Confidence: 80%**Tokens:** 109 input + 117 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	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: ComputeFileSha256 test

Why Needed: To ensure the computed SHA-256 hash of a file matches its content hash.

Key Assertions:

- {'expected_value': "b'test content'", 'actual_value': "b'test content'"}

Confidence: 80%

Tokens: 144 input + 71 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/util/hashing.py	6 lines (ranges: 32, 44-48)

PASSED

tests/test_hashing.py::TestComputeFileSha256::test_hashes_file

1ms

3

AI ASSESSMENT

Scenario: Test Compute File Sha256

Why Needed: To verify the correctness of the file hashing algorithm.

Key Assertions:

- {'expected': "A hash of length 64 for a file with contents 'hello world!'"}

Confidence: 80%

Tokens: 124 input + 64 output = 188 total

COVERAGE

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

- {'name': 'sig1', 'expected_type': 'bytes', 'actual_type': 'bytes'}
- {'name': 'sig2', 'expected_type': 'bytes', 'actual_type': 'bytes'}

Confidence: 80%

Tokens: 125 input + 91 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/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 + 72 output = 180 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	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 produces consistent results for the same input.

Key Assertions:

- {'expected': {'hash': 'd41d8cd98f00b804d0a131e86038e95'}, 'actual': {'hash': 'd41d8cd98f00b804d0a131e86038e95'}}}

Confidence: 80%

Tokens: 115 input + 113 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/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:

- {'message': 'Expected the hash length to be 64 hex chars.', 'expected_value': 64}

Confidence: 80%

Tokens: 103 input + 78 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: 32)

PASSED

tests/test_hashing.py::TestGetDependencySnapshot::test_includes_pytest 79ms 3

AI ASSESSMENT

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

Why Needed: To ensure that the 'pytest' package is included in the dependency snapshot.

Key Assertions:

- {'assertion': "snapshot contains 'pytest'", 'expected_result': 'True'}

Confidence: 80%

Tokens: 102 input + 78 output = 180 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	8 lines (ranges: 113-114, 116-121)

PASSED

tests/test_hashing.py::TestGetDependencySnapshot::test_returns_dict 81ms 3

AI ASSESSMENT

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

Why Needed: The function `get_dependency_snapshot()` should return a dictionary.

Key Assertions:

- snapshot is an instance of dict

Confidence: 80%

Tokens: 98 input + 55 output = 153 total

COVERAGE

src/pytest_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:**

- {'description': 'The loaded key should match the expected value.', 'expected_value': 'my-secret-key'}

Confidence: 80%**Tokens:** 145 input + 81 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	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	5 lines (ranges: 73, 76-77, 80-81)

PASSED

tests/test_hashing.py::TestLoadHmacKey::test_missing_key_file

1ms  4

AI ASSESSMENT

Scenario: Test Load HMAC Key with Missing Key File

Why Needed: To test the case where a key file does not exist.

Key Assertions:

- {'name': 'Expected result', 'type': 'NoneType', 'expected_value': 'None'}

Confidence: 80%

Tokens: 126 input + 72 output = 198 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	4 lines (ranges: 73, 76-78)

PASSED

tests/test_hashing.py::TestLoadHmacKey::test_no_key_file

1ms  4

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestLoadHmacKey::test_no_key_file

Why Needed: Because the HMAC key is not loaded.

Key Assertions:

- {'name': 'assert key is None', 'expected_result': 'None'}

Confidence: 80%

Tokens: 110 input + 70 output = 180 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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)

PASSED

tests/test_integration_gate.py::TestConfigDefaults::test_aggregation_defaults

1ms



AI ASSESSMENT

Scenario: Test aggregation default configuration.**Why Needed:** This test prevents regression where the default aggregation policy is set to 'latest' without including history.**Key Assertions:**

- config.aggregate_dir is None
- config.aggregate_policy == 'latest'
- config.aggregate_include_history is False

Confidence: 80%**Tokens:** 201 input + 73 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/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.capture_failed_output', 'expected_value': 'True'}

Confidence: 80%

Tokens: 107 input + 70 output = 177 total

COVERAGE

src/pytest_llm_report/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', 'expected_value': 'minimal'}

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/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: LLM is not enabled by default.

Key Assertions:

- {'name': 'is_llm_enabled() returns False for default config', 'expected_value': False, 'actual_value': 'get_default_config().is_llm_enabled()'}

Confidence: 80%

Tokens: 109 input + 91 output = 200 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the 'omit_tests_from_coverage' configuration option is set to True by default.

Key Assertions:

- config.omit_tests_from_coverage

Confidence: 80%

Tokens: 109 input + 66 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/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 that the provider is set to 'none' when it's not explicitly specified.

Key Assertions:

- {'name': "config.provider == 'none'", 'expected_result': 'none'}

Confidence: 80%

Tokens: 101 input + 79 output = 180 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 293)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_integration_gate.py::TestConfigDefaults::test_secret_exclude_globs

1ms



AI ASSESSMENT

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

Why Needed: This test is necessary because the default configuration does not exclude secret files by default.

Key Assertions:

- {'name': 'Excluding secret files', 'expected_result': ['secret'], 'actual_result': [False]}
- {'name': 'Including .env files', 'expected_result': ['.env'], 'actual_result': [True]}

Confidence: 80%

Tokens: 132 input + 117 output = 249 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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

7ms



5

AI ASSESSMENT

Scenario: The test verifies that the output of the deterministic pipeline is sorted by nodeid.

Why Needed: This test prevents a regression where the sorted nodeids are not as expected due to changes in the underlying data structure.

Key Assertions:

- nodeids should be equal to the sorted list of nodeids from the report.json file.
- nodeids should contain only the nodeids mentioned in the tests section of the report.json file.
- nodeids should not contain any duplicates.
- nodeids should be in ascending order (i.e., no nodes are skipped or duplicated)
- nodeids should not include any non-nodeid strings from the test results
- nodeids should only contain node ids that were actually tested and reported as passed
- nodeids should not be empty

Confidence: 80%

Tokens: 313 input + 177 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/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,

PASSED tests/test_integration_gate.py::TestFullPipeline::test_empty_test_suite 7ms ⚡ 6

AI ASSESSMENT

Scenario: Test that an empty test suite produces a valid report.

Why Needed: This test prevents regression where an empty test suite causes the report to be invalid.

Key Assertions:

- The `report.summary.total` property is set to 0.
- The `data` dictionary has a 'summary' key with a 'total' value of 0.
- The `json.loads()` function correctly parses the JSON data from the report file.

Confidence: 80%

Tokens: 240 input + 106 output = 346 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	62 lines (ranges: 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	123 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202-206, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED

tests/test_integration_gate.py::TestFullPipeline::test_html_report_generation 39ms 6

AI ASSESSMENT

Scenario: Test that the full pipeline generates an HTML report.**Why Needed:** This test prevents a regression where the HTML report is not generated correctly.**Key Assertions:**

- The file 'report.html' exists at the specified path.
- The content of the 'report.html' file contains the expected string '
- The content of the 'report.html' file contains the expected string 'test_pass'.

Confidence: 80%**Tokens:** 270 input + 100 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/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 64ms 7

AI ASSESSMENT

Scenario: The test verifies that the full pipeline generates a valid JSON report with the expected structure and content.

Why Needed: This test prevents regression when the pipeline fails to generate a JSON report due to an issue with the report writer configuration.

Key Assertions:

- The 'schema_version' key in the JSON output should be set to SCHEMA_VERSION.
- The 'summary' section of the JSON output should contain the correct total, passed, failed, and skipped counts.
- The 'passed', 'failed', and 'skipped' keys within the 'summary' section should have the expected values (1, 1, and 1 respectively).

Confidence: 80%

Tokens: 419 input + 148 output = 567 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 verifies that the ReportRoot object has required fields.

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

Key Assertions:

- ReportRoot has required fields: schema_version, run_meta, summary, and tests.
- Required fields are present in the data: 'schema_version', 'run_meta', 'summary', and 'tests'.
- The ReportRoot object contains all required fields.
- No report root is missing required fields.
- Missing required field would cause a validation error.

Confidence: 80%

Tokens: 250 input + 126 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/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 'test_run_meta_has_status_fields' verifies that RunMeta has run status fields.

Why Needed: This test prevents a potential regression where the RunMeta object does not have all required status fields.

Key Assertions:

- The 'exit_code' field is present in the data.
- The 'interrupted' field is present in the data.
- The 'collect_only' field is present in the data.
- The 'collected_count' field is present in the data.
- The 'selected_count' field is present in the data.

Confidence: 80%

Tokens: 237 input + 131 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/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 must be defined to ensure compatibility with the API.

Key Assertions:

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

Confidence: 80%

Tokens: 103 input + 88 output = 191 total

COVERAGE

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

PASSED

tests/test_integration_gate.py::TestSchemaCompatibility::test_test_case_has_required_fields

1ms



AI ASSESSMENT

Scenario: The test verifies that the TestCaseResult object has the required fields.

Why Needed: This test prevents a potential bug where the TestCaseResult object is missing one or more required fields, potentially leading to incorrect results or errors.

Key Assertions:

- nodeid
- outcome
- duration

Confidence: 80%

Tokens: 223 input + 76 output = 299 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 calls fail.

Why Needed: Prevents the test from passing if all retries are exhausted due to an API call failure.

Key Assertions:

- The `provider.annotate` method should raise an error with a meaningful message.
- The annotation should contain an `error` key with a non-None value.
- The `error` value should be a string indicating the cause of the failure.
- The `error` value should not be None.
- The `provider.annotate` method should not return any result if all retries are exhausted.
- The annotation should contain an `error` key with a non-None value.
- The `provider.annotate` method should raise an exception when the API call fails.
- The `provider.annotate` method should log an error message when the API call fails.

Confidence: 80%

Tokens: 346 input + 193 output = 539 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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 and no force refresh.**Key Assertions:**

- The API call should return an annotation with an error message.
- The error status code should be 500 (Internal server error).
- The test source file should not have any annotations.
- Any context files used by the test should also not have any annotations.
- If no force refresh is requested, the provider should not call LiteLLMProvider's annotate method.
- If a non-401 error occurs and no force refresh is requested, the result of annotate should be None.
- The annotation returned from annotate should contain an error message.

Confidence: 80%**Tokens:** 367 input + 167 output = 534 total

COVERAGE

src/pytest_llm_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 when API call fails twice before succeeding.

Why Needed: This test prevents regression in case of transient errors, where the LLM token refresh attempt fails but subsequent attempts succeed.

Key Assertions:

- The `test_foo()` function should be annotated with a successful result even though it was marked as transient.
- The API call to fail twice before succeeding should not raise an exception.
- The LLM token refresh attempt should eventually succeed after the transient error.

Confidence: 80%

Tokens: 433 input + 117 output = 550 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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

6.18s



7

AI ASSESSMENT

Scenario: Test that 401 error triggers token refresh (lines 123-126) when API call fails first, then succeeds.

Why Needed: To ensure the test catches and reports a retry of the token refresh after a 401 error occurs.

Key Assertions:

- The function `provider.annotate()` should have been called with an additional argument (2 or more).
- The error status code should be 401 for the first call to `mock_completion`.
- The error message should contain '{
- The response choices should include a new token.
- The retry count should be greater than or equal to 2 after a 401 error occurs.
- The annotation result should not be None when a 401 error occurs.

Confidence: 80%

Tokens: 473 input + 167 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	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 class is correctly instantiated when the 'gemini' provider is used.

Key Assertions:

- {'assertion': 'provider.__class__.__name__ == "GeminiProvider"', 'expected_result': 'GeminiProvider'}

Confidence: 80%

Tokens: 131 input + 91 output = 222 total

COVERAGE

src/pytest_llm_report/collector.py

14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)

src/pytest_llm_report/llm/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 None.**Key Assertions:**

- {'name': 'provider should be NoneType', 'expected_value': 'None'}

Confidence: 80%**Tokens:** 115 input + 80 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 created and returned from the get_provider function.

Key Assertions:

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

Confidence: 80%

Tokens: 154 input + 86 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/llm/base.py	8 lines (ranges: 65-66, 384, 386, 388, 391-392, 394)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_llm.py::TestGetProvider::test_unknown_raises 1ms ⚡ 4

COVERAGE

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

PASSED

tests/test_llm.py::TestLlmProviderContract::test_noop_implements_interface

1ms



5

AI ASSESSMENT

Scenario: Test that NoopProvider implements LlmProvider interface.**Why Needed:** Prevents a potential bug where NoopProvider is not implementing required methods of LlmProvider.**Key Assertions:**

- hasattr(provider, 'annotate')
- hasattr(provider, 'is_available')
- hasattr(provider, 'get_model_name')
- hasattr(provider, 'config')

Confidence: 80%**Tokens:** 232 input + 95 output = 327 total

COVERAGE

src/pytest_llm_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 the `NoopProvider` class returns an empty `LlmAnnotation` object when no annotation is provided.

Why Needed: This test prevents a regression where the `annotate` method does not return an error or warning message when no annotation is given, but instead returns an empty annotation.

Key Assertions:

- The `annotation` variable will be of type `LlmAnnotation`.
- The `scenario` attribute of the `annotation` object will be an empty string.
- The `why_needed` attribute of the `annotation` object will be an empty string.
- All `key_assertions` in the `annotation` object will be empty lists.

Confidence: 80%

Tokens: 249 input + 162 output = 411 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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 returns an empty string when the model name is not specified.**Key Assertions:**

- {'name': "assert get_model_name() == '', 'description': 'The `get_model_name` method should return an empty string.'}

Confidence: 80%**Tokens:** 114 input + 107 output = 221 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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:** To ensure the NoopProvider class is always available and does not raise any exceptions when instantiated.**Key Assertions:**

- {'name': 'provider.is_available()' should be a boolean value', 'expected_type': 'bool'}

Confidence: 80%**Tokens:** 108 input + 84 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/llm/base.py	5 lines (ranges: 65-66, 134, 137-138)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 59)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

tests/test_llm_contract.py

13 tests

PASSED

tests/test_llm_contract.py::TestAnnotationSchema::test_required_fields

1ms  2

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: The test verifies that the AnnotationSchema can parse a dictionary into a valid annotation.

Why Needed: This test prevents potential bugs where the AnnotationSchema does not correctly handle malformed input data.

Key Assertions:

- checks password
- checks username

Confidence: 80%

Tokens: 274 input + 66 output = 340 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/schemas.py	5 lines (ranges: 77-81)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_contract.py::TestAnnotationSchema::test_schema_handles_empty

1ms

3

AI ASSESSMENT

Scenario: This test checks if the AnnotationSchema can handle an empty input.

Why Needed: The test is necessary because it ensures that the AnnotationSchema can process and validate valid inputs, including empty ones.

Key Assertions:

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

Confidence: 80%

Tokens: 109 input + 107 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/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

COVERAGE

src/pytest_llm_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



AI ASSESSMENT

Scenario: The test verifies that the schema has required fields.

Why Needed: This test prevents a potential bug where the schema is not properly defined with required fields, potentially leading to errors or inconsistencies in the data.

Key Assertions:

- assert 'scenario' in ANNOTATION_JSON_SCHEMA['properties']
- assert 'why_needed' in ANNOTATION_JSON_SCHEMA['properties']
- assert 'key_assertions' in ANNOTATION_JSON_SCHEMA['properties']
- assert isinstance(ANNOTATION_JSON_SCHEMA, dict)
- assert len(ANNOTATION_JSON_SCHEMA) > 0
- assert all(key in ANNOTATION_JSON_SCHEMA for key in ['scenario', 'why_needed', 'key_assertions'])

Confidence: 80%

Tokens: 215 input + 164 output = 379 total

COVERAGE

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

PASSED

tests/test_llm_contract.py::TestAnnotationSchema::test_schema_to_dict

1ms



AI ASSESSMENT

Scenario: The test verifies that the `AnnotationSchema` instance correctly serializes to a dictionary.

Why Needed: This test prevents regression by ensuring that the `AnnotationSchema` class handles scenario and why_needed keys correctly when converting to a dictionary.

Key Assertions:

- assert data['scenario'] == 'Tests feature X'
- assert data['why_needed'] == 'Prevents bug Y'
- assert 'key_assertions' in data
- # Verify the presence of key_assertions

Confidence: 80%

Tokens: 247 input + 147 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/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: To test that the factory returns a NoopProvider for a specific provider.

Key Assertions:

- {'expected_type': 'NoopProvider', 'actual_type': 'get_provider(config)'}

Confidence: 80%

Tokens: 118 input + 80 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	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



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': 'type_of_provider', 'expected': 'LlmProvider', 'actual': 'NoopProvider'}

Confidence: 80%

Tokens: 117 input + 90 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	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_contract.py::TestNoopProvider::test_noop_returns_empty_annotation

1ms



5

AI ASSESSMENT

Scenario: The NoopProvider should return an empty annotation when the test function does not have any dependencies.

Why Needed: This test prevents a regression where the NoopProvider returns an incorrect annotation for tests with no dependencies.

Key Assertions:

- assert result.scenario == ""
- assert result.why_needed == ""
- assert result.key_assertions == []

Confidence: 80%

Tokens: 253 input + 93 output = 346 total

COVERAGE

src/pytest_llm_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 of the `ProviderContract` class returns an instance of `TestCaseResult` with the expected attributes.

Why Needed: This test prevents a potential regression where the `annotate` method does not return an instance of `TestCaseResult` with the expected attributes, potentially causing issues downstream in the testing framework.

Key Assertions:

- The `scenario` attribute is present and has the correct value.
- The `why_needed` attribute is present and has the correct value.
- The `key_assertions` list contains all the required assertions to verify the expected attributes of the returned object.

Confidence: 80%

Tokens: 263 input + 143 output = 406 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	8 lines (ranges: 65-66, 87-89, 97-98, 105)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 51)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_contract.py::TestProviderContract::test_provider_handles_empty_code

1ms

5

AI ASSESSMENT

Scenario:

tests/test_llm_contract.py::TestProviderContract::test_provider_handles_empty_code

Why Needed: To ensure the NoopProvider class handles empty code gracefully and returns a valid TestCaseResult.

Key Assertions:

- {'description': 'The provider should not return None for an empty test.', 'expected_result': 'Not None'}

Confidence: 80%

Tokens: 145 input + 87 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	8 lines (ranges: 65-66, 87-89, 97-98, 105)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 51)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_contract.py::TestProviderContract::test_provider_handles_none_context

1ms

5

AI ASSESSMENT

Scenario:

tests/test_llm_contract.py::TestProviderContract::test_provider_handles_none_context

Why Needed: To ensure the NoopProvider class can handle None context without raising an error.

Key Assertions:

- {'expected_value': 'None', 'actual_value': 'not None'}

Confidence: 80%

Tokens: 148 input + 77 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	8 lines (ranges: 65-66, 87-89, 97-98, 105)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 51)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_contract.py::TestProviderContract::test_provider_has_annotate_method 1ms 7

AI ASSESSMENT

Scenario:

tests/test_llm_contract.py::TestProviderContract::test_provider_has_annotate_method

Why Needed: To ensure that all providers have an annotate method.

Key Assertions:

- {'name': "has attribute 'annotate'", 'value': 'True'}
- {'name': 'is callable annotate method', 'value': 'True'}

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/llm/base.py	15 lines (ranges: 65-66, 384, 386, 388-389, 391-392, 394, 396-397, 399, 401-402, 404)
src/pytest_llm_report/llm/gemini.py	9 lines (ranges: 134-135, 137-141, 143-144)
src/pytest_llm_report/llm/litellm_provider.py	3 lines (ranges: 37-38, 41)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

 tests/test_llm_providers.py

52 tests

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_handles_context_too_large

1ms



5

AI ASSESSMENT

Scenario:

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_handles_context_too_large

Why Needed: The current implementation of annotate_handles_context can handle contexts up to 1000 tokens. However, with the increasing size of the input data, it may exceed this limit and cause performance issues.

Key Assertions:

- {'assertion_type': 'expected', 'expected_value': 1000, 'actual_value': 15000}

Confidence: 80%**Tokens:** 98 input + 114 output = 212 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	187 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 263-265, 299, 311-312, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 435, 437-439, 441-444, 449-452, 463-466, 471-473, 476-478, 497-498, 502-505, 507-508, 511, 514-516, 518-521, 524-525, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_missing_dependency

1ms



5

AI ASSESSMENT

Scenario: The LiteLLMProvider annotates the missing dependency 'litellm' in a test case.

Why Needed: This test prevents the provider from silently failing when a required dependency is not installed, potentially masking bugs or regressions.

Key Assertions:

- the annotation contains an error message indicating that 'litellm' is not installed.
- the annotation includes the correct installation instructions for 'litellm'.
- the annotation provides clear and concise feedback to the user about what they need to do to resolve the issue.

Confidence: 80%

Tokens: 270 input + 124 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	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 `GeminiProvider` requires an API token and prevents a missing token error.

Why Needed: The current implementation does not verify if the required API token is set before annotating the test. This can lead to unexpected behavior or errors when trying to annotate tests with missing tokens.

Key Assertions:

- assert annotation.error == 'GEMINI_API_TOKEN is not set'
- assert provider._config.api_token is None
- assert fake_genai.configure.called_with(None) == True
- assert fake_google.generativelai._path_.endswith[:] == []
- assert fake_google.generativelai._config.api_key is None
- assert provider._config._api_key is None
- assert fake_api_core.exceptions.ResourceExhausted.called_once() == True

Confidence: 80%

Tokens: 440 input + 180 output = 620 total

COVERAGE

src/pytest_llm_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: Test that annotate records tokens function works correctly by annotating a login test.

Why Needed: Prevents regressions and ensures accurate token usage tracking.

Key Assertions:

- Verify that the correct number of tokens are recorded on the limiter for the specified model.
- Check if the rate limits logic ran without error.
- Verify that the correct user is logged in based on the annotated test function.
- Ensure that the correct usage metadata is provided to the API.
- Verify that the total token count matches the expected value.
- Confirm that the annotation was successful and no errors occurred.

Confidence: 80%

Tokens: 783 input + 138 output = 921 total

COVERAGE

src/pytest_llm_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



AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: To ensure that the LLM provider can annotate retries correctly when rate limiting is in place.

Key Assertions:

- {'name': 'The annotation of retries should match the expected key', 'expected_value': 'annotation_retries', 'actual_value': 'annotation_retries'}
- {'name': 'The annotation of retries should be a dictionary with the correct keys', 'expected_value': {'key1': 'value1', 'key2': 'value2'}, 'actual_value': {}}

Confidence: 80%

Tokens: 98 input + 137 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	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: Rotating models on the daily limit is necessary because it prevents LLMs from being trained indefinitely and causing performance issues.**Key Assertions:**

- {'name': 'models are rotated', 'description': 'The model should be rotated after a certain number of iterations (e.g. 1000).'}
- {'name': 'no more than one model per day', 'description': 'No more than one LLM model should be trained on the system at any given time.'}

Confidence: 80%**Tokens:** 100 input + 139 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	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	210 lines (ranges: 39-42, 45-46, 48-50, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-230, 232, 235-236, 239-244, 246, 249-250, 252, 261, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457, 459, 461-466, 471-473, 476-478, 497-499, 502-505, 507-508, 511, 514-516, 518-521, 524, 526-527, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567-571, 574)

src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_skips_on_daily_limit

1ms



6

AI ASSESSMENT

Scenario: tests/test_llm_providers.py::TestGeminiProvider::test_annotate_skips_on_daily_limit**Why Needed:** To ensure that the LLM provider skips annotating tasks when the daily limit is exceeded.**Key Assertions:**

- {'name': 'Expected error message', 'description': 'The test expects an error message indicating that the daily limit has been reached.'}
- {'name': 'Expected exception type', 'description': 'The test expects a `LimitExceededError` to be raised when the daily limit is exceeded.'}

Confidence: 80%**Tokens:** 98 input + 134 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	47 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	216 lines (ranges: 39-42, 45-46, 48-50, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-230, 232-233, 235-236, 239-244, 246, 249-250, 252, 261, 318-320, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457, 459, 461-466, 471-473, 476-478, 497-499, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)

src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_success_with_mock_response

1ms



6

AI ASSESSMENT

Scenario: Test that LiteLLM provider annotates a successful response with the correct information.

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

Key Assertions:

- status ok
- redirect

Confidence: 80%

Tokens: 474 input + 60 output = 534 total

COVERAGE

src/pytest_llm_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::TestGeminiProvider::test_exhausted_model_recovers_after_24h

Why Needed: The test is needed because the model may not recover from an exhausted state after 24 hours. This could lead to a situation where the provider returns incorrect results for a certain period.

Key Assertions:

- {'name': 'model recovered', 'description': 'The model should be able to recover and return correct results after 24 hours.'}
- {'name': 'provider returns incorrect results', 'description': 'The provider should return incorrect results for a certain period after the model has recovered.'}

Confidence: 80%**Tokens:** 104 input + 152 output = 256 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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::TestGeminiProvider::test_fetch_available_models_error

Why Needed: To ensure that the `fetch_available_models` method raises an error when no models are available.

Key Assertions:

- {'name': 'method call', 'expected': 'fetch_available_models', 'actual': 'raise ValueError'}
- {'name': 'error message', 'expected': 'No models available.', 'actual': 'None'}

Confidence: 80%

Tokens: 92 input + 106 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	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 is refreshed after an interval.**Why Needed:** To ensure the model list is updated correctly and consistently with the LLM provider's refresh interval.**Key Assertions:**

- {'name': 'Model list is updated', 'description': 'The model list should be updated every time the test runs.'}
- {'name': 'Refresh interval is respected', 'description': "The LLM provider's refresh interval should be respected and not cause any issues with the test."}

Confidence: 80%**Tokens:** 96 input + 120 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/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	201 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246, 249-250, 252, 261, 340-343, 346-349, 352, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457-458, 463-466, 471-473, 476-478, 497-499, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)

src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_401_retry_with_token_refresh

1ms



7

AI ASSESSMENT

Scenario: Test that LiteLLM provider retries on 401 after refreshing token.**Why Needed:** Reason for retrying with token refresh.**Key Assertions:**

- Verify the correct API key is captured before and after token refresh.
- Verify the response data contains the expected scenario, why needed, and key assertions.
- Verify that the first call to fake_completion raises a FakeAuthError (401 Unauthorized).
- Verify that the second call to fake_completion returns a successful response with the correct API key.
- Verify that the captured keys are in the correct order (token-1 before token-2).

Confidence: 80%**Tokens:** 580 input + 142 output = 722 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	50 lines (ranges: 37-38, 41-42, 44-48, 60-61, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 122, 124-127, 170-174, 176-178, 182, 186-188, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/llm/token_refresh.py	28 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156, 160-162)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_handles_completion_error

1ms



5

AI ASSESSMENT

Scenario: The test verifies that the LiteLLMProvider annotates completion errors correctly.**Why Needed:** This test prevents a regression where LiteLLM providers do not surface completion errors in annotations.**Key Assertions:**

- The `error` attribute of the annotation is set to `boom` when a completion error occurs.
- The `boom` string is present within the `error` attribute.
- The test case asserts that the `error` attribute is not `None`
- The `boom` string is found in the `error` attribute

Confidence: 80%**Tokens:** 307 input + 128 output = 435 total

COVERAGE

src/pytest_llm_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 where the provider incorrectly handles invalid key_assertions payloads, making it harder to identify and fix issues.

Key Assertions:

- response_data must be a dictionary
- response_data must contain 'key_assertions'
- response_data must not be empty
- response_data must have exactly one key_assertion
- response_data.key_assertions should be a list

Confidence: 80%

Tokens: 346 input + 112 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/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



AI ASSESSMENT

Scenario: The LiteLLMProvider should report a missing dependency error when the required package is not installed.

Why Needed: This test prevents a potential bug where the LiteLLMProvider does not correctly handle cases where the required package is not installed, potentially leading to silent failures or incorrect results.

Key Assertions:

- The annotation returned by the annotate method should include an error message indicating that the required dependency 'litellm' is missing and how to install it.
- The error message should be in the format 'required package not installed. Install with: pip install '.
- The error message should provide a clear indication of what needs to be done to resolve the issue.
- The annotation should include the exact name of the required package, which is 'litellm' in this case.
- The annotation should include the correct installation command for the required package.
- The annotation should not silently fail or produce incorrect results if the required package is not installed.

Confidence: 80%

Tokens: 271 input + 217 output = 488 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	9 lines (ranges: 65-66, 87-89, 97-99, 105)
src/pytest_llm_report/llm/litellm_provider.py	8 lines (ranges: 37-38, 41, 82-86)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_success_with_mock_response

1ms



6

AI ASSESSMENT

Scenario: Test that the LiteLLM provider annotates a successful response correctly.**Why Needed:** Prevents regression by ensuring the annotation is correct for a valid response.**Key Assertions:**

- The annotation contains the expected scenario, why needed, and key assertions.
- The annotation has a non-zero confidence level.
- The captured model matches the one used in the test.
- The 'tests/test_auth.py::test_login' message is present in the response.
- The 'def test_login()' message is present in the response.
- The system role of the message is 'system'.
- The status OK assertion is present in the response.

Confidence: 80%**Tokens:** 475 input + 150 output = 625 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	34 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-187, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_with_prompt_override

1ms



6

AI ASSESSMENT

Scenario: Test that LiteLLMProvider overrides the prompt when provided.**Why Needed:** To ensure that the LiteLLM provider correctly handles prompt override scenarios.**Key Assertions:**

- The annotation returned by `_annotate_internal` method does not contain any messages.
- The content of the annotation's error message is 'CUSTOM PROMPT'.
- The annotation contains a custom prompt as expected.
- The key 'why_needed' in the annotation matches the provided reason.
- The key 'key_assertions' in the annotation matches the expected list of assertions.
- The value of the content key in the captured messages is 'CUSTOM PROMPT'.
- The type of the error message in the annotation is None.

Confidence: 80%**Tokens:** 373 input + 161 output = 534 total

COVERAGE

src/pytest_llm_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:

tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_with_token_usage

Why Needed: Prevents regression in token usage extraction for LiteLLM providers.**Key Assertions:**

- The `token_usage` attribute of the annotation returned by `provider.annotate(test, 'src')` is not None.
- The value of `prompt_tokens` in `annotation.token_usage` is set to `100`.
- The value of `completion_tokens` in `annotation.token_usage` is set to `50`.
- The value of `total_tokens` in `annotation.token_usage` is set to `150`.
- The `token_usage` attribute is correctly populated with the expected values even when there are no token usage data available.
- The test verifies that the `prompt_tokens`, `completion_tokens`, and `total_tokens` attributes have the correct values for a given test case.
- The test also verifies that the `token_usage` attribute has a value of `None` when it should be `None` (i.e., no token usage data is available).
- The test ensures that the `token_usage` attribute is correctly populated with the expected values even in cases where there are multiple choices or no completion tokens.

Confidence: 80%**Tokens:** 426 input + 278 output = 704 total

COVERAGE

src/pytest_llm_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)

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 completion call.

Why Needed: This test prevents regression in case when API base is not provided in the config.

Key Assertions:

- The value of `api_base` in the response data should be 'https://proxy.corp.com/v1'.
- The value of `litellm_api_base` in the config should be 'https://proxy.corp.com/v1'.
- The value of `api_base` in the completion call should match the one provided in the config.
- The response data should contain a key named `scenario`.
- The response data should contain a key named `why_needed`.
- The response data should contain a key named `key_assertions`.
- The value of `api_base` in the completion call should be 'https://proxy.corp.com/v1'.

Confidence: 80%

Tokens: 387 input + 221 output = 608 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	35 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182-183, 186-187, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_api_key_passthrough

1ms



AI ASSESSMENT

Scenario: The liteLLM provider should pass 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 call, potentially causing issues with the model's behavior or performance.

Key Assertions:

- The API key should be captured and stored in the `captured` dictionary.
- The response data from the fake completion function should include the expected 'api_key' key.
- The 'api_key' value in the response data should match the static API key provided by the environment (TEST_KEY).
- The 'key_assertions' list should contain the expected 'api_key' assertion.
- The captured dictionary should have an 'api_key' key with the correct value.
- The fake completion function should return a response data object with the expected 'response_data' key and 'api_key' value.
- The 'why_needed' assertion should indicate that this test prevents a regression in API key passing to the completion call.

Confidence: 80%

Tokens: 384 input + 222 output = 606 total

COVERAGE

src/pytest_llm_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)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_auth_error_without_refresher 1ms ⚡ 5

AI ASSESSMENT

Scenario: Test that the LiteLLM provider returns an auth error when no refresher is configured.

Why Needed: This test prevents a bug where the provider returns an authentication error without refreshing the token, potentially causing unexpected behavior or errors in downstream applications.

Key Assertions:

- The `provider.annotate(test, 'src')` annotation should include an error message indicating that authentication failed.
- The `annotation.error` attribute should contain the string 'Authentication failed'.
- The `annotation.error` attribute should not be `None` when the test passes.
- The `annotation.error` attribute should contain the exact phrase 'Authentication failed' to ensure it matches the expected error message.
- The `annotation.error` attribute should be a string, not a list or other data structure.
- The `annotation.error` attribute should have the correct type (str) to ensure it's a string value.

Confidence: 80%

Tokens: 338 input + 198 output = 536 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/litellm_provider.py	36 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 122, 132-133, 170-174, 176-178, 182, 186-187, 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_auth_retry_fails_on_second_attempt 2.00s 6

AI ASSESSMENT

Scenario: Test that the LiteLLM provider reports an authentication error when retrying after a second failure.

Why Needed: To prevent the provider from reporting an authentication error on subsequent retries, which could mask legitimate errors.

Key Assertions:

- The `litellm_token_refresh_command` is set to 'get-token'.
- The `llm_max_retries` is set to 2. If this value is exceeded, the provider will retry after a second failure.
- The `AuthenticationError` raised by the fake completion function is not caught by the provider's authentication error handler.
- The `litellm_token_refresh_command` is called with an argument 'token-new' when the retry attempt fails. This should trigger the authentication error handler.
- The `litellm_token_refresh_command` is called without any arguments when the retry attempt succeeds. The authentication error handler should not be triggered in this case.
- If the provider's authentication error handler catches the `FakeAuthError`, it should not report an authentication error on subsequent retries.
- The `litellm_token_refresh_command` is set to 'get-token' with a different argument when the retry attempt fails. This should trigger the authentication error handler and prevent the provider from reporting an authentication error on subsequent retries.
- If the provider's authentication error handler raises an exception, it should not be caught by the `fake_run` function and thus prevent the provider from reporting an authentication error on subsequent retries.

Confidence: 80%

Tokens: 419 input + 317 output = 736 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/litellm_provider.py	51 lines (ranges: 37-38, 41-42, 44-48, 60-61, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 122, 124-127, 129-130, 132-133, 141-142, 170-174, 176-178, 182, 186-188, 190)

src/pytest_llm_report/llm/token_refresh.py	31 lines (ranges: 59-60, 63-66, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156, 160-162)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

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

AI ASSESSMENT

Scenario: Test 'LiteLLMProvider::test_context_too_long_error' verifies that the LiteLLM provider handles context too long error correctly.

Why Needed: The test prevents a potential bug or regression where the LiteLLM provider throws an error when handling responses with invalid contexts.

Key Assertions:

- The response is not None, indicating that the function does not throw an exception
- The response contains the expected JSON structure
- The 'error' key in the response matches the expected value
- The 'why_needed' key in the response is empty, as it's not necessary for this test
- The 'key_assertions' list in the response is empty, indicating that the function does not throw an exception

Confidence: 80%

Tokens: 370 input + 166 output = 536 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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 calling get_max_context_tokens.

Key Assertions:

- {'name': 'result', 'expected_value': 16384, 'message': 'Expected result to be 16384'}

Confidence: 80%

Tokens: 218 input + 79 output = 297 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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



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': 'get_max_context_tokens returns an error when no context is provided', 'expected_value': 0, 'actual_value': 1}
- {'name': 'get_max_context_tokens returns a fallback value for max context tokens when an error occurs', 'expected_value': 10, 'actual_value': 20}

Confidence: 80%**Tokens:** 101 input + 130 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/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/litellm_provider.py	10 lines (ranges: 37-38, 41, 221-222, 224, 227, 232-234)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_get_max_context_tokens_success

1ms

5

AI ASSESSMENT

Scenario:

tests/test_llm_providers.py::TestLiteLLMProvider::test_get_max_context_tokens_success

Why Needed: To ensure the LiteLLM provider correctly returns the maximum context tokens.

Key Assertions:

- {'description': 'The function should return 8192 as the maximum context tokens.', 'expected_value': 8192}

Confidence: 80%

Tokens: 213 input + 89 output = 302 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/litellm_provider.py	9 lines (ranges: 37-38, 41, 221-222, 224, 227-229)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_is_available_with_module 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py::TestLiteLLMProvider::test_is_available_with_module

Why Needed: To test the `is_available` method of the `LiteLLMProvider` class, which checks if a required module is installed.

Key Assertions:

- {'name': 'Mocking sys.modules', 'expected': {'litellm': 'fake_litellm'}, 'actual': 'fake_litellm'}

Confidence: 80%

Tokens: 160 input + 109 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	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 LiteLLM provider's token refresh integration.**Why Needed:** The test prevents a bug where the provider does not refresh tokens in a timely manner, potentially causing issues with dynamic token usage.**Key Assertions:**

- Verify that the `litellm_token_refresh_command` is set to 'get-token'.
- Check if the `litellm_token_refresh_interval` is set to 3600 seconds (1 hour).
- Assert that the `api_key` variable in the captured output matches the expected value.
- Verify that the provider's `LiteLLMProvider` instance has been updated with the new configuration.
- Check if the `test_case()` method of the `LiteLLMProvider` instance returns a `CaseResult` object with an outcome of 'passed'.
- Assert that the captured output contains the expected API key value.

Confidence: 80%**Tokens:** 442 input + 194 output = 636 total

COVERAGE

src/pytest_llm_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)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_transient_err or_retry

1ms



6

AI ASSESSMENT

Scenario: Test that the LiteLLMProvider retries transient errors and that the test passes when there are more than 3 retry attempts.

Why Needed: This test prevents a regression where the provider does not retry transient errors, potentially causing the test to fail with a ConnectionError.

Key Assertions:

- The provider should raise a ConnectionError on the first attempt (0 calls).
- The provider should raise a ConnectionError on the second attempt (1 call).
- The provider should raise a ConnectionError on the third attempt (2 calls) and pass the test.
- The provider should not retry transient errors if there are less than 3 attempts left.
- The provider should not retry transient errors if there are more than 3 attempts left.

Confidence: 80%

Tokens: 426 input + 169 output = 595 total

COVERAGE

src/pytest_llm_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 can handle context length errors and still return valid JSON.**Key Assertions:**

- {'name': "Expected a dictionary with 'scenario', 'why_needed', and 'key_assertions' keys", 'expected_value': {'scenario': 'tests/test_llm_providers.py', 'why_needed': 'To ensure that the LLM provider can handle context length errors and still return valid JSON.', 'key_assertions': [...]}}

Confidence: 80%**Tokens:** 103 input + 125 output = 228 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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 method returns an error message when a call to Ollama raises a call error.

Why Needed: This test prevents regression where the annotation fails to detect call errors in Ollama providers.

Key Assertions:

- error == 'Failed after 2 retries. Last error: boom'
- annotation.error == 'Failed after 2 retries. Last error: boom'
- provider._call_ollama().__name__ == 'boom'
- test_case.__name__ == 'test_case'
- test_case.__doc__ is None

Confidence: 80%

Tokens: 347 input + 136 output = 483 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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 reports missing httpx dependency when annotating a test case.

Why Needed: This test prevents a bug where the provider incorrectly assumes that httpx is installed and reports an error instead of suggesting to install it.

Key Assertions:

- assert annotation.error == 'httpx not installed. Install with: pip install httpx'
- provider.annotate(test, 'def test_case(): assert True')
- test_case()
- assert test_case().__name__ == 'test_case'

Confidence: 80%

Tokens: 268 input + 121 output = 389 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	9 lines (ranges: 65-66, 87-89, 97-99, 105)
src/pytest_llm_report/llm/ollama.py	5 lines (ranges: 42-46)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_annotate_runtime_error_immediate_fail

1ms



AI ASSESSMENT

Scenario:

tests/test_llm_providers.py::TestOllamaProvider::test_annotate_runtime_error_immediate_fail

Why Needed: The test is failing because the OLLAMA provider is not annotating runtime errors immediately.

Key Assertions:

- {'name': 'OllamaProvider should annotate runtime error', 'expected_value': 'True'}
- {'name': 'OllamaProvider should return immediate annotation result', 'expected_value': 'Immediate annotation result'}

Confidence: 80%

Tokens: 99 input + 119 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/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 and returns a CaseResult object.

Why Needed: Prevents authentication bugs by ensuring that the Ollama provider returns an error when the login process fails.

Key Assertions:

- check status
- validate token
- assert True

Confidence: 80%

Tokens: 414 input + 82 output = 496 total

COVERAGE

src/pytest_llm_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: This test prevents regression in case providers override the default prompt.

Key Assertions:

- {'description': 'Verify that the provider sets the correct error message.', 'condition': 'annotation.error is None', 'assertion': "captured_messages[0][1]['content'] == 'CUSTOM PROMPT'"}
- {'description': 'Verify that the custom prompt is used for annotation.', 'condition': "captured_messages[0][1]['content'] != 'default prompt'", 'assertion': 'False'}

Confidence: 80%

Tokens: 373 input + 149 output = 522 total

COVERAGE

src/pytest_llm_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: The test verifies that the `annotate` method of `LiteLLMProvider` correctly extracts token usage from a response.

Why Needed: This test prevents regression in handling cases where token usage is not provided in the response.

Key Assertions:

- The number of prompt tokens should be equal to 100.
- The number of completion tokens should be equal to 50.
- The total number of tokens should be equal to 150.
- The `token_usage` attribute should not be `None`.
- The value of `prompt_tokens` should match the expected value of 100.
- The value of `completion_tokens` should match the expected value of 50.
- The value of `total_tokens` should match the expected value of 150.

Confidence: 80%

Tokens: 426 input + 175 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/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/ollama.py	40 lines (ranges: 42-43, 49, 52, 55, 58, 60-61, 63-67, 71, 74-80, 83, 92, 190, 192-200, 204-207, 209, 211-212)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_call_ollama_success

1ms



5

AI ASSESSMENT

Scenario: Test Ollama provider makes correct API call to generate response.**Why Needed:** Prevents regression in API call functionality of the Ollama provider.**Key Assertions:**

- The function `_call_ollama()` returns a dictionary with the expected 'response', 'model', 'prompt', and 'system' values.
- The captured URL and JSON data match the expected values from the API call.
- The timeout value is set to the expected 60 seconds.
- The response is not empty or None, as expected for a successful API call.
- The model and prompt are correctly retrieved from the captured JSON data.
- The system prompt is used in the generated response, as expected.
- The stream parameter is False, indicating no streaming of the response.
- The timeout value is respected during the API call.

Confidence: 80%**Tokens:** 470 input + 187 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/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



AI ASSESSMENT

Scenario: Test that the default model is used when not specified for Ollama provider.

Why Needed: This test prevents a regression where the default model is not used when it should be.

Key Assertions:

- The captured JSON response contains the default model 'llama3.2'.
- The captured JSON response does not contain any custom model specified by the user.
- The captured JSON response has the correct key 'model' with value 'llama3.2'.

Confidence: 80%

Tokens: 344 input + 114 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/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



AI ASSESSMENT

Scenario: Ollama provider returns False when server is unavailable

Why Needed: The Ollama provider should return False when the server is unavailable.

Key Assertions:

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

Confidence: 80%

Tokens: 183 input + 81 output = 264 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the availability of an Ollama provider when it returns a non-200 status code.**Key Assertions:**

- {'name': 'provider._check_availability()' should return False for a 500 status code', 'expected_value': False}

Confidence: 80%**Tokens:** 197 input + 97 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	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	5 lines (ranges: 113-114, 116-118)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_check_availability_success

1ms



5

AI ASSESSMENT

Scenario: The test verifies that the Ollama provider checks for availability via the /api/tags endpoint successfully.

Why Needed: This test prevents a potential regression where the provider fails to check availability due to a misconfigured or outdated API.

Key Assertions:

- The URL '/api/tags' is included in the provided URL.
- A response with a status code of 200 is returned from the /api/tags endpoint.
- The OllamaProvider instance's _check_availability method returns True after calling this test.

Confidence: 80%

Tokens: 296 input + 121 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/ollama.py	5 lines (ranges: 113-114, 116-118)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_get_max_context_tokens_context_length_key

1ms



5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py**Why Needed:** To ensure that the `get_max_context_tokens_context_length` method returns the correct key for the context length in the JSON response.**Key Assertions:**

- {'name': 'Context Length', 'expected_value': 100, 'actual_value': 0}

Confidence: 80%**Tokens:** 99 input + 79 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	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::TestOllamaProvider::test_get_max_context_tokens_fallback_on_error

Why Needed: To handle cases where the maximum context token count is exceeded during training.**Key Assertions:**

- {'name': 'Expected max_context_tokens to be greater than 0', 'description': 'The `max_context_tokens` parameter should have a value greater than 0.'}
- {'name': "Expected error message to contain 'maximum context tokens exceeded'", 'description': "The error message should contain the phrase 'maximum context tokens exceeded'."}

Confidence: 80%**Tokens:** 101 input + 141 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/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	11 lines (ranges: 138, 140, 142-147, 175-176, 178)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_get_max_context_tokens_from_model_info

1ms



5

AI ASSESSMENT

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

- {'name': 'max_context_tokens', 'value': 10}
- {'name': 'context_token_count', 'value': 20}

Confidence: 80%**Tokens:** 99 input + 84 output = 183 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	16 lines (ranges: 138, 140, 142-147, 149-150, 156, 165-167, 172-173)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_get_max_context_tokens_from_parameters

1ms



5

AI ASSESSMENT

Scenario: Tests for OLLAMA provider**Why Needed:** To ensure that the maximum context tokens can be retrieved from parameters correctly.**Key Assertions:**

- {'name': 'max_context_tokens', 'expected_value': 100, 'actual_value': 64}
- {'name': 'context_token_length', 'expected_value': 128, 'actual_value': 96}

Confidence: 80%**Tokens:** 97 input + 102 output = 199 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/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 Ollama provider returns an error when the input is too large to be processed in a single context.**Key Assertions:**

- {'name': 'Status code', 'value': 200}
- {'name': 'Response content type', 'value': 'application/json'}
- {'name': 'Content length', 'value': 0}

Confidence: 80%**Tokens:** 101 input + 97 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	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:

- {'assertion': 'provider.is_local() is True', 'expected_result': True}

Confidence: 80%

Tokens: 123 input + 82 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/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: Test Ollama Provider: test_parse_response_invalid_json**Why Needed:** To ensure the OllamaProvider correctly handles invalid JSON responses and reports an error.**Key Assertions:**

- {'name': 'Error message', 'expected': 'Failed to parse LLM response as JSON'}

Confidence: 80%**Tokens:** 138 input + 78 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/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: This test checks that the Ollama provider rejects invalid key_assertions payloads.**Why Needed:** The Ollama provider should reject responses with invalid key_assertions payloads.**Key Assertions:**

- {'message': 'Invalid response: key_assertions must be a list', 'code': 400}

Confidence: 80%**Tokens:** 174 input + 135 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/llm/base.py	16 lines (ranges: 65-66, 325-326, 329-330, 333-334, 337-339, 342, 344, 346-348)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_code_fence

1ms

5

AI ASSESSMENT

Scenario:

tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_code_fence

Why Needed: To ensure that the Ollama provider correctly extracts JSON from markdown code fences.

Key Assertions:

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

Confidence: 80%

Tokens: 127 input + 178 output = 305 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	20 lines (ranges: 65-66, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/schemas.py	6 lines (ranges: 38, 42-44, 46-47)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_plain_fence

1ms

5

AI ASSESSMENT

Scenario:

tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_plain_fence

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

Key Assertions:

- {'name': 'response is a string', 'expected_value': 'string', 'actual_value': ''}

Confidence: 80%

Tokens: 128 input + 97 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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



AI ASSESSMENT

Scenario: Test Ollama provider parses valid JSON responses with correct scenario, why needed, and key assertions.

Why Needed: Prevents bugs by ensuring the Ollama provider correctly identifies scenarios and key assertions in valid JSON responses.

Key Assertions:

- assert a
- assert b

Confidence: 80%

Tokens: 292 input + 69 output = 361 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: Test verify water-fill algorithm satisfies smaller files first with constrained token budget.

Why Needed: This test prevents regression in the water-fill algorithm when distributing tokens to files, ensuring that smaller files are given priority and have sufficient content to fill their allocated budget.

Key Assertions:

- The function `distribute_token_budget` should allocate a total of 10 tokens to `small.py` and no more than 45 tokens to `large.py` when distributing the token budget.
- The allocation for `small.py` should be at least 10 tokens, but not exceed 30 tokens.
- The allocation for `large.py` should be between 30 tokens and 45 tokens inclusive.
- The total allocated tokens should sum up to 60 (budget) - 16 (tokens needed by small.py) = 44 tokens for large.py.
- The remaining budget after allocating to small.py should be sufficient for large.py, allowing it to get at least 38 content tokens (44 - 6 overhead).

Confidence: 80%

Tokens: 396 input + 223 output = 619 total

COVERAGE

src/pytest_llm_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:** This test ensures that the `distribute_token_budget` function behaves correctly when given an empty input or no budget.**Key Assertions:**

- {'assertion': {'message': 'Expected distribute_token_budget to return {}'}, 'description': 'Test case for empty input'}
- {'assertion': {'message': 'Expected distribute_token_budget to return {}'}, 'description': 'Test case for no budget'}

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/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 regression where both files are large and the budget is insufficient to cover their content.**Key Assertions:**

- The contents of `l1.py` and `l2.py` should be roughly equal.
- The allocation for `l1.py` should be between 35% and 50% of its total content (44 tokens).
- The allocation for `l2.py` should also be between 35% and 50% of its total content (44 tokens).

Confidence: 80%**Tokens:** 327 input + 129 output = 456 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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:** Verify the limit of max_files in the distribute_token_budget function to prevent over-allocation of files.**Key Assertions:**

- {'name': 'length of allocations should be equal to 3', 'expected_value': 3, 'actual_value': 1, '# This test is failing because only one file was allocated. Verify why this happened and correct it if necessary!': "reason_for_failure": 'Only one file was allocated instead of three.'}"}

Confidence: 80%**Tokens:** 133 input + 134 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/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 the function `distribute_token_budget` correctly allocates tokens to files when the budget is sufficient.

Why Needed: This test prevents a potential bug where the function does not allocate enough tokens to all files, resulting in incomplete content.

Key Assertions:

- The length of allocations should be equal to 2.
- Each file's allocation should match its required amount (10 tokens for f1.py and 10 tokens for f2.py).
- All files should have their allocated tokens within the total budget (32 tokens in this case).

Confidence: 80%

Tokens: 332 input + 130 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/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 rough token estimation (chars / 4) for an empty string.

Why Needed: Prevents a potential division by zero error when estimating tokens in the absence of any input.

Key Assertions:

- assert estimate_tokens([]) == 1
- assert estimate_tokens('') == 1
- assert estimate_tokens('a') == 1
- assert estimate_tokens('aaaa') == 1
- assert estimate_tokens('aaaa' * 10) == 10

Confidence: 80%

Tokens: 217 input + 119 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/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 that `CoverageEntry.to_dict()` correctly serializes the test data.

Why Needed: This test prevents a potential bug where coverage data is not properly serialized to JSON.

Key Assertions:

- The 'file_path' key in the dictionary should match the expected value.
- The 'line_ranges' key in the dictionary should match the expected value.
- The 'line_count' key in the dictionary should match the expected value.
- All values in the dictionary should be strings or integers, as they represent coverage data.
- Any non-string or integer values should be ignored during serialization.

Confidence: 80%

Tokens: 255 input + 138 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/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 the test entry.

Why Needed: This test prevents a potential bug where the serialized `CoverageEntry` is not as expected, potentially leading to incorrect coverage data.

Key Assertions:

- The 'file_path' key in the dictionary should match the original file path.
- The 'line_ranges' key in the dictionary should match the original line ranges.
- The 'line_count' key in the dictionary should match the original line count.

Confidence: 80%

Tokens: 255 input + 117 output = 372 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	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 serialization correctly.

Why Needed: CoverageEntry may not serialize correctly if line ranges are invalid or missing.

Key Assertions:

- The 'file_path' key in the dictionary matches the expected value.
- The 'line_ranges' key in the dictionary matches the expected value.
- The 'line_count' key in the dictionary matches the expected value.
- All assertions pass for a single CoverageEntry instance.

Confidence: 80%

Tokens: 255 input + 103 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	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 key-value pairs or default values.

Key Assertions:

- annotation.scenario == "" (empty string)
- annotation.why_needed == "Empty annotation should have default values."
- annotation.key_assertions == [] (no key-value pairs or default values)
- assert annotation.confidence is None (default confidence value)
- assert annotation.error is None (default error message)

Confidence: 80%

Tokens: 212 input + 124 output = 336 total

COVERAGE

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

PASSED

tests/test_models.py::TestLlmAnnotation::test_to_dict_minimal

1ms



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 regression where the minimal annotation is missing some required fields.

Key Assertions:

- required_fields = ['scenario', 'why_needed', 'key_assertions']
- confidence field is optional and not included when None
- asserts that 'confidence' is not present in the dictionary

Confidence: 80%

Tokens: 230 input + 107 output = 337 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	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 to dictionary with all fields**Why Needed:** Prevents incorrect output when not providing all required fields for LlmAnnotation.**Key Assertions:**

- Asserts that the 'scenario' field is correctly set to 'Tests user login'.
- Asserts that the 'confidence' field is correctly set to 0.95.
- Asserts that the 'context_summary' field has the expected mode and bytes value.
- Asserts that the 'error' field is None (correct behavior for LlmAnnotation).
- Asserts that the 'key_assertions' list contains all required assertions.
- Asserts that the dictionary d has all the correct keys and values.

Confidence: 80%**Tokens:** 284 input + 157 output = 441 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 schema version and empty lists.**Why Needed:** Prevents a potential bug where the default report does not have a valid schema version or contains non-empty lists of tests, collection errors.**Key Assertions:**

- d['schema_version'] == SCHEMA_VERSION
- d['tests'] == []
- not in d ['warnings', 'collection_errors']
- d['schema_version'] not in d

Confidence: 80%**Tokens:** 231 input + 105 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/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 with Collection Errors should include them.**Why Needed:** This test prevents a regression where the report does not include collection errors when they exist.**Key Assertions:**

- The 'collection_errors' key in the report dictionary is present and has exactly one item.
- The value of the 'nodeid' field within the first collection error object is set to 'test_bad.py'.
- The 'message' field within the first collection error object contains the specified message 'SyntaxError'.

Confidence: 80%**Tokens:** 237 input + 117 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	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: Test Report Root**Why Needed:** To ensure the test report includes warnings.**Key Assertions:**

- {'assertion': "The length of the 'warnings' list in the report is 1.", 'expected_value': 1, 'actual_value': 0}
- {'assertion': "The code in the first warning is 'W001'.", 'expected_value': 'W001', 'actual_value': 'No coverage'}

Confidence: 80%**Tokens:** 144 input + 116 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/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	55 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestReportRoot::test_tests_sorted_by_nodeid 1ms ⚡ 3

AI ASSESSMENT

Scenario: Tests should be sorted by nodeid in output.

Why Needed: Because the current implementation does not sort tests by nodeid, it may cause unexpected behavior when sorting or filtering test results.

Key Assertions:

- {`'assertion': "nodeids == ['a_test.py::test_a', 'm_test.py::test_m', 'z_test.py::test_z']",`
`'expected_result': ['a_test.py::test_a', 'm_test.py::test_m', 'z_test.py::test_z'], 'actual_result':`
`['a_test.py::test_a', 'm_test.py::test_m', 'z_test.py::test_z']}`}

Confidence: 80%

Tokens: 215 input + 159 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	73 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestReportWarning::test_to_dict_with_detail 1ms ⚡ 3

COVERAGE

src/pytest_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: Prevents a warning about missing detailed information.

Key Assertions:

- The 'detail' key is expected to be present in the dictionary.
- The value of 'detail' is not provided when it's absent.
- The test verifies that the 'detail' key does not exist.

Confidence: 80%

Tokens: 223 input + 88 output = 311 total

COVERAGE

src/pytest_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 present in the test data.

Why Needed: This test prevents regression where RunMeta is missing or incorrectly configured aggregation fields.

Key Assertions:

- The 'run_id' field should be present and equal to 'run-123'.
- The 'run_group_id' field should be present and equal to 'group-456'.
- The 'is_aggregated' field should be True.
- The 'aggregation_policy' field should be 'merge'.
- The 'run_count' field should be 3.
- The 'source_reports' list should have exactly two elements.

Confidence: 80%

Tokens: 343 input + 147 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/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 are disabled.**Why Needed:** This test prevents a regression where the LLM fields (llm_annotations_enabled, llm_provider, and llm_model) are included in the RunMeta object even when annotations are not enabled.**Key Assertions:**

- The 'llm_annotations_enabled' key is present in the data.
- The 'llm_provider' key is not present in the data.
- The 'llm_model' key is not present in the data.

Confidence: 80%**Tokens:** 232 input + 121 output = 353 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: Verify that LLM traceability fields are included when enabled for the provided RunMeta instance.

Why Needed: This test prevents regression where the presence of annotations is not properly tracked or reported by the model.

Key Assertions:

- The value of 'llm_annotations_enabled' in the data dictionary should be True.
- The value of 'llm_provider' in the data dictionary should match the provided 'ollama' string.
- The value of 'llm_model' in the data dictionary should match the provided 'llama3.2:1b' string.
- The value of 'llm_context_mode' in the data dictionary should match the provided 'complete' string.
- The value of 'llm_annotations_count' in the data dictionary should be 10.
- The value of 'llm_annotations_errors' in the data dictionary should be 2.

Confidence: 80%

Tokens: 327 input + 197 output = 524 total

COVERAGE

src/pytest_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: This test ensures that non-aggregated reports do not include source reports.

Key Assertions:

- {'expected_value': {'source_reports': []}, 'actual_value': {'source_reports': []}}
- {'expected_value': {'is_aggregated': False}, 'actual_value': {'is_aggregated': True}}

Confidence: 80%

Tokens: 130 input + 116 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/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 handling of legacy and new fields in the `RunMeta` class.**Key Assertions:**

- The 'git_sha' field is correctly set to 'abc1234'.
- The 'git_dirty' field is correctly set to True.
- The 'repo_version' field is correctly set to '1.0.0'.
- The 'repo_git_sha' field is correctly set to 'abc1234'.
- The 'repo_git_dirty' field is correctly set to True.
- The 'plugin_git_sha' field is correctly set to 'def5678'.
- The 'plugin_git_dirty' field is correctly set to False.
- The 'config_hash' field is correctly set to 'def5678'.
- The length of the `source_reports` list is correctly set to 1.

Confidence: 80%**Tokens:** 483 input + 207 output = 690 total

COVERAGE

src/pytest_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: Test RunMeta to include run status fields.**Why Needed:** This test prevents a regression where the RunMeta object does not contain all necessary run status fields, potentially leading to incorrect analysis results.**Key Assertions:**

- The 'exit_code' field in the RunMeta object should be equal to 1.
- The 'interrupted' field in the RunMeta object should be True.
- The 'collect_only' field in the RunMeta object should be True.
- The 'collected_count' field in the RunMeta object should be equal to 10.
- The 'selected_count' field in the RunMeta object should be equal to 8.
- The 'deselected_count' field in the RunMeta object should be equal to 2.

Confidence: 80%**Tokens:** 285 input + 175 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/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: The schema version should be in semver format.

Key Assertions:

- {'name': "schema_version.split('.').should.have.length.equal.to(3)", 'message': 'Schema version should be in semver format.'}

Confidence: 80%

Tokens: 115 input + 83 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)

PASSED

tests/test_models.py::TestSchemaVersion::test_schema_version_in_repo_rt_root

1ms

3

AI ASSESSMENT

Scenario: tests/test_models.py::TestSchemaVersion::test_schema_version_in_report_root

Why Needed: The test is necessary to ensure that the ReportRoot object 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 + 109 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	54 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestSourceCoverageEntry::test_to_dict

1ms  3

AI ASSESSMENT

Scenario: CoverageEntry serialization test.**Why Needed:** This test prevents a potential bug where the coverage entry is not properly serialized to JSON.**Key Assertions:**

- The 'file_path' key in the dictionary should match the expected value.
- The 'line_ranges' key in the dictionary should match the expected format (e.g., '1-3, 5, 10-15').
- The 'line_count' key in the dictionary should match the expected value (10 in this case).

Confidence: 80%**Tokens:** 256 input + 119 output = 375 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	8 lines (ranges: 96-103)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestSourceReport::test_to_dict_minimal

1ms  3

AI ASSESSMENT

LLM error: Failed to parse LLM response as JSON

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	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: tests/test_models.py::TestSourceReport::test_to_dict_with_run_id**Why Needed:** To ensure that the SourceReport object's run_id attribute is correctly included in its dictionary representation.**Key Assertions:**

- {'name': "Expected value of 'run_id' key", 'value': 'run-1'}

Confidence: 80%**Tokens:** 134 input + 85 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/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 that the `CoverageEntry` class correctly serializes a coverage summary.

Why Needed: This test prevents regression in coverage reporting functionality.

Key Assertions:

- The 'file_path' key is set to the correct file path.
- The 'line_ranges' key is set to the expected line ranges.
- The 'line_count' key is set to the correct line count.

Confidence: 80%

Tokens: 254 input + 95 output = 349 total

COVERAGE

src/pytest_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: Test that a minimal result has the required fields.

Why Needed: This test prevents regression where a minimal result is not provided with all necessary information.

Key Assertions:

- The 'nodeid' field should match the node ID of the test.
- The 'outcome' field should be set to 'passed'.
- The 'duration' field should be set to 0.0 (or any other default value).
- The 'phase' field should be set to 'call'.

Confidence: 80%

Tokens: 244 input + 119 output = 363 total

COVERAGE

src/pytest_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: tests/test_models.py::TestTestCaseResult::test_result_with_coverage verifies that the test result includes a coverage list.

Why Needed: This test prevents regression by ensuring that the coverage report is always present and accurate.

Key Assertions:

- The length of the 'coverage' key in the result dictionary should be equal to 1.
- The value of the 'file_path' key within the 'coverage' list should match the provided file path.
- Each item in the 'coverage' list should have a 'file_path' attribute matching the provided file path.

Confidence: 80%

Tokens: 256 input + 131 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	24 lines (ranges: 65-68, 190, 194-199, 201, 203, 205, 207, 210-212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestTestCaseResult::test_result_with_llm_opt_out

1ms

3

AI ASSESSMENT

Scenario: tests/test_models.py::TestTestCaseResult::test_result_with_llm_opt_out

Why Needed: To ensure that the LLM opt-out flag is correctly set in the result.

Key Assertions:

- {'name': 'assert llm_opt_out is True', 'expected_value': True, 'message': '', 'type': 'assertion'}

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/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 case result should include rerun fields.**Key Assertions:**

- {'name': 'rerun_count', 'expected_value': 2, 'message': 'Rerun count is not equal to 2'}
- {'name': 'final_outcome', 'expected_value': 'passed', 'message': 'Final outcome is not passed'}

Confidence: 80%**Tokens:** 162 input + 118 output = 280 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 ensures that the `result` dictionary does not include 'rerun_count' and 'final_outcome' fields when a result without reruns is returned.

Key Assertions:

- {'field': 'final_outcome', 'expected_value': 'passed', 'actual_value': 'passed'}

Confidence: 80%

Tokens: 152 input + 148 output = 300 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	19 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

 tests/test_models_coverage.py

15 tests

PASSED

tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_all_optional_fields

1ms



4

AI ASSESSMENT

Scenario: Test to_dict includes all optional fields when set.**Why Needed:** Prevents bar regression in coverage report generation.**Key Assertions:**

- assert result['param_id'] == 'a-b-c',
- assert result['param_summary'] == 'a=1, b=2, c=3',
- assert result['captured_stdout'] == 'stdout content',
- assert result['captured_stderr'] == 'stderr content',
- assert result['requirements'] == ['REQ-100'],
- assert result['llm_opt_out'] is True,
- assert result['llm_context_override'] == 'complete',
- assert len(result['coverage']) == 1,
- assert result['llm_annotation']['scenario'] == 'Tests foo'

Confidence: 80%**Tokens:** 454 input + 182 output = 636 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	76 lines (ranges: 96-103, 241-243, 263-266, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526-540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_artifacts 1ms 3

AI ASSESSMENT

Scenario: Test to_dict includes artifacts when set.**Why Needed:** This test prevents a potential bug where the 'to_dict' method does not include all required artifacts in the report.**Key Assertions:**

- The length of 'artifacts' in the result dictionary is 2.
- The path of the first artifact entry in 'result' is 'report.html'.
- All required artifacts are included in the 'artifacts' list in 'result'.

Confidence: 80%**Tokens:** 264 input + 109 output = 373 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	59 lines (ranges: 263-266, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530-532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_collection_errors 1ms 3

AI ASSESSMENT

Scenario: Test to_dict includes collection_errors when set.**Why Needed:** Prevents a potential bug where the test fails due to missing or incorrect collection errors in the report.**Key Assertions:**

- The length of `result['collection_errors']` is 1.
- The value of `result['collection_errors'][0]['nodeid']` is 'broken_test.py'.
- The node id of the first collection error matches the expected node id.

Confidence: 80%**Tokens:** 243 input + 108 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/models.py	58 lines (ranges: 241-243, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526-528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_custom_metadata

1ms



3

AI ASSESSMENT

Scenario: Test to_dict includes custom_metadata when set.**Why Needed:** Prevents a potential bug where custom metadata is not included in the report dictionary even if it's set.**Key Assertions:**

- The 'custom_metadata' key should be present in the result dictionary with the correct values.
- The 'custom_metadata' key should contain the expected project, environment, and build_number values.
- The custom metadata should not be overridden by default settings (e.g., 'project', 'environment').
- Custom metadata should be included when the 'to_dict' method is called with a ReportRoot object that has it set.
- The custom metadata value should match the expected value for each key.
- If custom_metadata is not provided, the report dictionary should still contain default values (e.g., 'project': None, 'environment': None).
- Custom metadata should be included in the report even if it's empty or contains only default values.

Confidence: 80%**Tokens:** 264 input + 211 output = 475 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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



AI ASSESSMENT

Scenario:

tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_hmac_signature

Why Needed: HMAC signature is included in the report when it's set.

Key Assertions:

- {'expected': 'hmac_signature', 'actual': 'signature123'}

Confidence: 80%

Tokens: 128 input + 77 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/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: tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_sha256

Why Needed: The test is necessary to ensure that the ReportRoot class correctly converts its internal state into a dictionary, including any SHA-256 hashes.

Key Assertions:

- {'name': 'sha256 in report.to_dict()', 'expected': 'abcdef1234567890', 'actual': 'None'}

Confidence: 80%

Tokens: 131 input + 105 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/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:** This test prevents a regression where the coverage information is not included in the report.**Key Assertions:**

- The length of `source_coverage` should be 1.
- The file path of the first element in `source_coverage` should match 'src/mod.py'.
- The source_coverage entry for 'src/mod.py' should have a 'file_path' key matching 'src/mod.py'.

Confidence: 80%**Tokens:** 282 input + 111 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/models.py	63 lines (ranges: 96-103, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532-534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_warnings 1ms 4

AI ASSESSMENT

Scenario: tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_warnings

Why Needed: This test is needed to ensure that the `to_dict` method of `ReportRoot` includes warnings when set.

Key Assertions:

- {'name': "len(result['warnings']) == 1", 'expected_value': 1, 'message': "Expected len(result['warnings']) to be 1, but got {}", 'actual_value': '1'}
- {'name': "result['warnings'][0]['code'] == 'W001'", 'expected_value': 'W001', 'message': "Expected result['warnings'][0]['code'] to be 'W001', but got {}", 'actual_value': "'W001'"}

Confidence: 80%

Tokens: 151 input + 182 output = 333 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	55 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models_coverage.py::TestSummaryToDict::test_to_dict_with_coverage_total_percent 1ms 3

AI ASSESSMENT

Scenario: ...

Why Needed: ...

Key Assertions:

- {'assertion': "result['coverage_total_percent'] == 85.5", 'expected_result': 85.5}

Confidence: 80%

Tokens: 153 input + 57 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/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

COVERAGE

src/pytest_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 includes all optional fields when set.**Why Needed:** Prevents bar regression in coverage reporting.**Key Assertions:**

- assert result['param_id'] == 'a-b-c',
- assert result['param_summary'] == 'a=1, b=2, c=3',
- assert result['captured_stdout'] == 'stdout content',
- assert result['captured_stderr'] == 'stderr content',
- assert result['requirements'] == ['REQ-100'],
- assert result['llm_opt_out'] is True,

Confidence: 80%**Tokens:** 454 input + 136 output = 590 total

COVERAGE

src/pytest_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: to include captured stderr in the result dictionary when to_dict is called on a TestCaseResult object

Key Assertions:

- {'name': 'result', 'expected_value': {'captured_stderr': 'Error output here'}, 'actual_value': 'Error output here'}

Confidence: 80%

Tokens: 149 input + 98 output = 247 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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



AI ASSESSMENT

Scenario:

tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_captured_stdout

Why Needed: The `to_dict` method includes captured stdout when set.

Key Assertions:

- {'name': 'captured_stdout', 'expected_value': 'Debug output here'}

Confidence: 80%

Tokens: 149 input + 78 output = 227 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	20 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218-220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_param_summary

1ms



COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	21 lines (ranges: 190, 194-199, 201, 203-207, 210, 212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_requirements

1ms

3

AI ASSESSMENT

Scenario:

tests/test_models_coverage.py::TestTestCaseResultToDict::test_to_dict_with_requirements

Why Needed: To include requirements in the test case result dictionary when set.

Key Assertions:

- {'assertion_type': 'contains', 'expected_value': ['REQ-001', 'REQ-002'], 'actual_value': ['REQ-001', 'REQ-002']}

Confidence: 80%

Tokens: 151 input + 96 output = 247 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the LLM context.**Why Needed:** This test prevents a potential bug where the default exclude globs are not correctly set.**Key Assertions:**

- The function `defaults` contains the expected globs: `*.pyc`, `__pycache__/*`, and `*secret*`.
- The function `defaults` does not contain the expected glob `*password*`.

Confidence: 80%**Tokens:** 222 input + 103 output = 325 total

COVERAGE

src/pytest_llm_report/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: Test the default redact patterns of the Config class.**Why Needed:** Prevents a potential issue where sensitive information like API keys might be exposed if they are not properly redacted.**Key Assertions:**

- The `--password` and `--token` patterns should match any occurrences in the provided redact patterns.
- The `--api[_-]?key` pattern should match any occurrences in the provided redact patterns.

Confidence: 80%**Tokens:** 228 input + 103 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	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 test_default_values scenario.

Why Needed: This test prevents a regression where the default configuration settings are not properly initialized.

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'
- not cfg.is_llm_enabled() is True

Confidence: 80%

Tokens: 318 input + 184 output = 502 total

COVERAGE

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

PASSED

tests/test_options.py::TestConfig::test_get_default_config

1ms  3

AI ASSESSMENT

Scenario: tests/test_options.py

Why Needed: To ensure that the default configuration is correctly set to 'none'.

Key Assertions:

- {'name': 'cfg', 'expected_type': 'Config'}
- {'name': 'cfg.provider', 'expected_value': 'none'}

Confidence: 80%

Tokens: 104 input + 81 output = 185 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 is_llm_enabled check is enabled for different providers.

Why Needed: Prevents regression in LLM configuration when switching between providers.

Key Assertions:

- The function `is_llm_enabled` should return False for provider 'none'.
- The function `is_llm_enabled` should return True for provider 'ollama'.
- The function `is_llm_enabled` should return True for provider 'litellm'.
- The function `is_llm_enabled` should return True for provider 'gemini'.

Confidence: 80%

Tokens: 263 input + 128 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	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options.py::TestConfig::test_validate_invalid_aggregate_policy

1ms



3

AI ASSESSMENT

Scenario: test_validate_invalid_aggregate_policy**Why Needed:** To ensure that the `aggregate_policy` parameter is validated correctly and raises an error when it's invalid.**Key Assertions:**

- {'name': 'length of errors list', 'value': 1, 'expected_value': 1}
- {'name': 'error message in first error', 'value': "Invalid aggregate_policy 'random'", 'expected_value': "Invalid aggregate_policy 'random'")}

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/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



AI ASSESSMENT

Scenario: test_validate_invalid_context_mode**Why Needed:** To ensure that the `llm_context_mode` is valid and raises an error when it's invalid.**Key Assertions:**

- {'assertion_type': 'contains', 'value': "Invalid llm_context_mode 'mega_max'", 'expected_value': "Invalid llm_context_mode 'mega_max'"}

Confidence: 80%**Tokens:** 131 input + 92 output = 223 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 does not cause any issues during validation.**Key Assertions:**

- {'description': "The error message should indicate an invalid include phase 'lunch_break'.", 'expected_value': "Invalid include_phase 'lunch_break'", 'actual_value': "Invalid include_phase 'lunch_break'"}

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/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 ensure that the Config class correctly handles and reports invalid providers.**Key Assertions:**

- {'name': 'length of errors list', 'value': 1, 'expected_value': 1}
- {'name': 'error message in errors list', 'value': "Invalid provider 'invalid_provider'", 'expected_value': "Invalid provider 'invalid_provider'""}

Confidence: 80%**Tokens:** 122 input + 116 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	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:** Prevents regression due to invalid configuration values, ensuring correct behavior under different scenarios.**Key Assertions:**

- cfg.validate() should return at least 5 error messages.
- The 'llm_context_bytes' value must be at least 1000.
- The 'llm_max_tests' value must be 0 (no limit) or positive.
- The 'llm_requests_per_minute' value must be at least 1.
- The 'llm_timeout_seconds' value must be at least 1.
- The 'llm_max_retries' value must be 0 or positive.
- All error messages should contain the specified constraint strings.

Confidence: 80%**Tokens:** 329 input + 166 output = 495 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 configuration is properly validated and no errors are raised when a valid configuration is provided.**Key Assertions:**

- {'name': 'Configuration validation should be successful for a valid config', 'description': 'The function should return an empty list of errors when given a well-formed, non-empty configuration.'}

Confidence: 80%**Tokens:** 100 input + 99 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)

AI ASSESSMENT

Scenario: Test the `load_aggregation_options` function to ensure it correctly loads aggregation options.

Why Needed: This test prevents a bug where the aggregation policy is not being loaded correctly, potentially leading to incorrect results or errors in downstream processing.

Key Assertions:

- The `aggregate_dir` attribute of the configuration object should be set to 'aggr_dir'.
- The `aggregate_policy` attribute of the configuration object should be set to 'merge'.
- The `aggregate_run_id` attribute of the configuration object should be set to 'run-123'.
- The `aggregate_group_id` attribute of the configuration object should be set to 'group-abc'.

Confidence: 80%

Tokens: 295 input + 154 output = 449 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 batch flag is disabled by default and correctly handled when it's set to None.

Key Assertions:

- {'name': 'cfg.batch_parametrized_tests is True', 'expected_value': True, 'message': 'Expected cfg.batch_parametrized_tests to be True'}

Confidence: 80%

Tokens: 138 input + 100 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	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 potential bug where the LLM configuration is not properly loaded due to missing pyproject.toml file.**Key Assertions:**

- The `llm_max_retries` attribute in the config should be set to 10 by default.
- The `rootpath` attribute of the mock Pytest configuration should point to a temporary directory.
- The `llm_report_html`, etc. attributes in the config should not have any values assigned.
- The `llm_provider` and `llm_model` attributes should be set to their default values (e.g., 'default', 'default')
- The `llm_context_mode` attribute should be set to its default value ('strict')
- The `llm_context_compression` attribute should be set to its default value ('off')
- No other attributes in the config should have any values assigned.

Confidence: 80%**Tokens:** 413 input + 214 output = 627 total

COVERAGE

src/pytest_llm_report/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:**

- {'name': 'mock_pytest_config.option.llm_coverage_source', 'expected_value': 'cov_dir'}

Confidence: 80%**Tokens:** 126 input + 72 output = 198 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	86 lines (ranges: 123, 171, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607-608, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options.py::TestLoadConfig::test_load_defaults

4ms  3

AI ASSESSMENT

Scenario: tests/test_options.py::TestLoadConfig::test_load_defaults**Why Needed:** To test the functionality of the default configuration when no options are set.**Key Assertions:**

- {'name': "cfg.provider == 'none'", 'expected': 'None'}
- {'name': 'cfg.report_html is None', 'expected': '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: test_load_from_cli_overrides_pyproject

Why Needed: To test that CLI options override pyproject.toml options.

Key Assertions:

- {'name': 'CLI options override pyproject.toml options', 'expected_value': 'True'}

Confidence: 80%

Tokens: 134 input + 70 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	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 config from cli provider override

Why Needed: CLI provider option overrides pyproject.toml

Key Assertions:

- {'name': 'override pyproject.toml', 'expected_value': 'pyproject.toml'}

Confidence: 80%

Tokens: 130 input + 65 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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:

- {'name': 'mock_pytest_config.option.llm_max_retries', 'expected_value': 2, 'actual_value': 2}

Confidence: 80%

Tokens: 130 input + 85 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	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)

AI ASSESSMENT

Scenario: test_load_from_pyproject

Why Needed: To test the functionality of the `load_config` function in `tests/test_options.py`.

Key Assertions:

Confidence: 80%

Tokens: 119 input + 193 output = 312 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	134 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-336, 340, 342, 344, 348, 352, 356, 360-362, 364, 366, 368, 372, 374, 378, 380, 382-384, 386-388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options.py::TestLoadConfig::test_load_token_optimization_options 4ms 3

AI ASSESSMENT

Scenario: Test loading token optimization options from CLI.**Why Needed:** Prevents a bug where the 'llm_prompt_tier' option is set to 'minimal' and the 'batch_parametrized_tests' option is True, causing an error when running tests in parallel.**Key Assertions:**

- cfg.prompt_tier == 'minimal'
- cfg.batch_parametrized_tests is False
- cfg.context_compression == 'none'

Confidence: 80%**Tokens:** 264 input + 104 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/options.py	88 lines (ranges: 123, 171, 308, 311-312, 320-322, 460, 463, 466, 470-474, 476-477, 479, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

 tests/test_options_coverage.py

47 tests

PASSED

tests/test_options_coverage.py::TestCliOverrides::test_cli_dependency_snapshot

5ms



3

AI ASSESSMENT

Scenario: Verify that the `llm_dependency_snapshot` option is used to specify a dependency snapshot file.

Why Needed: This test prevents a potential regression where the `llm_dependency_snapshot` option is not used correctly, leading to incorrect or missing dependency snapshots in the report.

Key Assertions:

- The value of `llm_dependency_snapshot` is set to `deps.json` as expected.
- The configuration file `deps.json` is loaded and its contents are compared with the expected snapshot.
- The `report_dependency_snapshot` field in the configuration is updated correctly based on the specified snapshot file.
- No dependency snapshots are missing or incorrect when using the correct `llm_dependency_snapshot` option.
- Dependency snapshots are generated correctly for all dependencies when using the correct `llm_dependency_snapshot` option.

Confidence: 80%**Tokens:** 213 input + 180 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	92 lines (ranges: 123, 171, 308, 311-312, 320-322, 460-461, 463-464, 466-467, 470-474, 476-477, 479, 482, 484, 486, 488, 490-492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestCliOverrides::test_cli_evidence_bundle 6ms 3

AI ASSESSMENT

Scenario: Verify that the 'llm_evidence_bundle' option is correctly set to 'bundle.zip' when CLI override is enabled.

Why Needed: This test prevents a potential bug where the 'llm_evidence_bundle' option is not set to the correct value even though CLI override is enabled.

Key Assertions:

- The value of 'llm_evidence_bundle' in the configuration file should be 'bundle.zip'.
- The value of 'llm_evidence_bundle' in the configuration file should match the expected value when CLI override is enabled.
- The mock object's option.llm_evidence_bundle attribute should have been set to 'bundle.zip'.

Confidence: 80%

Tokens: 217 input + 151 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/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 5ms 3

AI ASSESSMENT

Scenario: Verify that the `test_cli_report_json` test sets the expected value for `report_json` in the configuration.

Why Needed: This test prevents a potential bug where the report JSON is not set correctly, potentially leading to incorrect or missing reports.

Key Assertions:

- The `llm_report_json` option is set to 'output.json' in the mock configuration.
- The `report_json` value in the loaded configuration matches 'output.json'.
- The expected value for `report_json` is not being overridden by any other options.
- The test ensures that the correct value is used for report JSON.
- No other values are overriding or setting the `report_json` option.
- The mock configuration does not contain any other override for `report_json`.
- The loaded configuration has a valid and expected value for `report_json`.

Confidence: 80%

Tokens: 212 input + 193 output = 405 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: Verify that the `test_cli_report_pdf` test function sets the `report_pdf` option to 'output.pdf' correctly.

Why Needed: This test prevents a potential bug where the `llm_report_pdf` option is not set to the expected value, causing the CLI report PDF to be generated incorrectly.

Key Assertions:

- The `report_pdf` option is set to 'output.pdf'.
- The `llm_report_pdf` option is set to 'output.pdf'.
- The output of `cfg.report_pdf` is 'output.pdf'.

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/options.py	92 lines (ranges: 123, 171, 308, 311-312, 320-322, 460-461, 463-464, 466-467, 470-474, 476-477, 479, 482, 484, 486-488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestConfigValidationCoverage::test_validate_invalid_token_output_format 1ms 3

AI ASSESSMENT

Scenario: test_validate_invalid_token_output_format

Why Needed: To ensure that the token output format is valid and does not cause coverage issues.

Key Assertions:

- {'assertion_type': 'contains', 'pattern': 'litellm_token_output_format', 'value': 'xml'}

Confidence: 80%

Tokens: 130 input + 78 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	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 current token refresh interval (30 seconds) is too short, which may lead to unexpected behavior and security vulnerabilities.

Key Assertions:

- {'description': 'The token refresh interval must be at least 60 seconds', 'expected_value': 60, 'actual_value': 30}

Confidence: 80%

Tokens: 146 input + 94 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	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 validation of valid LiteLLM config.**Why Needed:** To ensure the validation process works correctly with a valid LiteLLM configuration.**Key Assertions:**

- {'assertion': 'The validate method returns an empty list of errors.', 'expected_result': [], 'message': 'Expected the validate method to return an empty list of errors.'}

Confidence: 80%**Tokens:** 142 input + 90 output = 232 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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

1ms



3

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 loaded correctly and includes all necessary history.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': 'include_history=true', 'actual': 'include_history=true'}
- {'name': 'aggregate_include_history file path', 'expected': '/path/to/aggregate_include_history.py', 'actual': '/path/to/aggregate_include_history.py'}

Confidence: 80%**Tokens:** 118 input + 133 output = 251 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438-440, 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_value': 'pyproject.toml should exist and have an aggregate_policy section'}
- {'name': 'aggregate_policy is present in pyproject.toml', 'expected_value': 'aggregate_policy should be present in pyproject.toml'}

Confidence: 80%**Tokens:** 121 input + 137 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, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_all_config_keys_combined 2ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_all_config_keys_combined

Why Needed: To ensure that all configuration keys are loaded when loading the PyProject.

Key Assertions:

- {'name': 'All config keys should be present in pyproject.toml', 'expected': [...], 'actual': {'scenario': 'tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_all_config_keys_combined', 'why_needed': '...', 'key_assertions': [...]}}}

Confidence: 80%

Tokens: 120 input + 125 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	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



AI ASSESSMENT

Scenario: tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_cache_dir

Why Needed: To ensure the cache directory is loaded correctly from the PyProject file.

Key Assertions:

- {'name': 'pyproject.toml exists', 'expected': 'True'}
- {'name': 'cache_dir path exists in pyproject.toml', 'expected': '/path/to/cache/dir'}

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_cache_ttl_seconds

1ms



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_cache_ttl_seconds

Why Needed: To ensure that the cache TTL seconds are loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists and is not empty', 'value': 'True'}
- {'name': 'pyproject.toml contents match expected format', 'value': 'True'}

Confidence: 80%

Tokens: 116 input + 110 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	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 1ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_failed_output

Why Needed: The test is failing because the `capture_failed_output` key is not present in the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml', 'value': "Missing 'capture_failed_output' key in pyproject.toml"}

Confidence: 80%

Tokens: 116 input + 96 output = 212 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418-420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_output_max_chars 2ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_output_max_cha

Why Needed: To ensure that the `capture_output.max_chars` option is correctly loaded and applied when running tests.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'max_chars = 1024', 'actual': 'max_chars = 2048'}

Confidence: 80%

Tokens: 119 input + 101 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	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

1ms



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_bytes

Why Needed: To ensure that the `context_bytes` feature is properly loaded from the PyPI repository.**Key Assertions:**

- {'name': 'Context bytes are present in the PyPI repository', 'expected_value': 'True'}
- {'name': 'Context bytes are correctly extracted and returned', 'expected_value': 'The `context_bytes` feature is properly loaded from the PyPI repository.'}

Confidence: 80%**Tokens:** 113 input + 123 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_context_exclude_globs 2ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_exclude_globs

Why Needed: To ensure that the `context_exclude_globs` setting in pyproject.toml is properly loaded and excluded from the test coverage.

Key Assertions:

- {'name': 'pyproject.toml file exists and is readable', 'description': 'The pyproject.toml file should exist and be readable.'}
- {'name': 'context_exclude_globs setting is defined in pyproject.toml', 'description': 'The `context_exclude_globs` setting should be defined in the `pyproject.toml` file.'}

Confidence: 80%

Tokens: 119 input + 152 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	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368-369, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_file_limit

2ms

3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_file_limit

Why Needed: To ensure that the context file limit is correctly loaded from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists', 'expected': 'True'}
- {'name': 'pyproject.toml content', 'expected': 'The contents of pyproject.toml are correct.'}

Confidence: 80%

Tokens: 116 input + 111 output = 227 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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

1ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_include_globs

Why Needed: To ensure that the `context_include_globs` option is properly loaded from the PyProject.toml file.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': "[\"*.txt\", \"*.py\"]", '# Expected contents of pyproject.toml file\n": "", 'actual': "[\"*.txt\", \"*.py\"]\n"}
- {'name': 'context_include_globs option value', 'expected': '[\"*.txt\", \"*.py\"]', '# Expected value of context_include_globs option\n": "", 'actual': '[\"*.txt\", \"*.py\"]\n'}

Confidence: 80%**Tokens:** 119 input + 179 output = 298 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366-368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_hmac_key_file

1ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_hmac_key_file

Why Needed: To ensure that the hmac key file is loaded correctly and used for HMAC operations.**Key Assertions:**

- {'name': 'pyproject.toml exists', 'expected': 'pyproject.toml exists in the current working directory', 'actual': 'pyproject.toml was created at path /tmp/pyproject.toml'}
- {'name': 'pyproject.toml is not empty', 'expected': 'pyproject.toml is not empty', 'actual': 'pyproject.toml contents were read successfully'}

Confidence: 80%**Tokens:** 118 input + 153 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	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446-447, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_param_values 2ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_param_values

Why Needed: To ensure that the include_param_values function in options.py is correctly loading parameter values from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': '''\ntoml\ninclude_param_values =\n["path/to/file1", "path/to/file2"]\n''', 'actual': '''\ntoml\ninclude_param_values = []\n'''}

Confidence: 80%

Tokens: 116 input + 132 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	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372-374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_phase 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 exists and has an includes section', 'expected': 'True', 'actual': 'False'}
- {'name': 'includes section in pyproject.toml is not empty', 'expected': 'True', 'actual': ''}

Confidence: 80%**Tokens:** 113 input + 123 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, 412-413, 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 for `tests/test_options_coverage`**Why Needed:** To ensure that the `include_pytest_invocation` option is correctly loaded from `pyproject.toml`.**Key Assertions:**

- {'name': 'The `include_pytest_invocation` option is present in `pyproject.toml`', 'value': 'True'}
- {'name': 'The `include_pytest_invocation` option has the correct path', 'value': '/path/to/include_pytest_invocation'}

Confidence: 80%**Tokens:** 122 input + 126 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	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 1ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_invocation_redact_patterns

Why Needed: To ensure that the invocation_redact_patterns are correctly loaded from the pyproject.toml file and used during testing.

Key Assertions:

- {'assertion_type': 'file existence', 'expected_value': 'invocation_redact_patterns.toml', 'actual_value': 'pyproject.toml'}

Confidence: 80%

Tokens: 121 input + 105 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, 430-432, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_api_base 2ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_api_base

Why Needed: To ensure that the litellm_api_base is loaded correctly from the pyproject.toml file.

Confidence: 80%

Tokens: 122 input + 54 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	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': 'litellm_api_key', 'expected_value': '', 'actual_value': '<loaded_api_key>'}

Confidence: 80%

Tokens: 122 input + 92 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/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_llm_token_json_key

1ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_llm_token_json_key

Why Needed: To ensure that the llm token JSON key is correctly loaded from the pyproject.toml file.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': "llm_token_json_key = 'path/to/llm_token.json'", 'actual': 'pyproject.toml contents'}

Confidence: 80%**Tokens:** 125 input + 113 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	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

1ms



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 loaded correctly from the pyproject.toml file.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': "The contents of the pyproject.toml file should contain a section named 'llm.token.output_format'."}
- {'name': 'llm.token.output_format value', 'expected': "The value of the 'llm.token.output_format' setting in the pyproject.toml file should be 'llm_token_output_format'."}

Confidence: 80%**Tokens:** 125 input + 161 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	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 properly loaded from the `pyproject.toml` file.

Key Assertions:

- {'name': 'The `llm_token_refresh_command` should be present in the `pyproject.toml` file', 'expected_value': 'llm_token_refresh_command'}
- {'name': 'The `llm_token_refresh_command` should have the correct path to the token', 'expected_value': '/path/to/token'}

Confidence: 80%

Tokens: 125 input + 150 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/options.py	111 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342, 344-346, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_token_refresh_interval 2ms 3

AI ASSESSMENT

Scenario: Loading a specific configuration file from the .pyproject.toml file.**Why Needed:** To ensure that the correct token refresh interval is loaded for litellm.**Key Assertions:**

- {'name': 'The correct token refresh interval is loaded', 'description': "The value of 'litellm.token_refresh_interval' in the .pyproject.toml file should be equal to the expected value."}

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 1ms 3

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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` setting in `pyproject.toml` is correctly applied when loading dependencies.

Key Assertions:

- The `max_concurrency` value in `pyproject.toml` is read correctly from disk.
- The `max_concurrency` value in `pyproject.toml` is not overridden by any environment variables.

Confidence: 80%

Tokens: 116 input + 109 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/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

1ms



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 should contain a 'max_tests' setting", 'value': 'True'}
- {'name': "pyproject.toml should have a value for 'max_tests'", 'value': 10}

Confidence: 80%

Tokens: 113 input + 126 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	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

1ms



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_metadata_file

Why Needed: To ensure that the metadata file is loaded correctly from the pyproject.toml file.**Key Assertions:**

- {'name': 'pyproject.toml exists', 'expected': 'True'}
- {'name': 'metadata_file_path', 'expected': '/path/to/pyproject.toml'}

Confidence: 80%**Tokens:** 113 input + 104 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	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 1ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_ollama_host

Why Needed: To ensure that the ollama_host is loaded correctly from the pyproject.toml file.

Key Assertions:

- {'name': 'The ollama_host is present in the pyproject.toml file', 'value': 'True'}
- {'name': 'The ollama_host is a valid Python module', 'value': 'True'}

Confidence: 80%

Tokens: 119 input + 119 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	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, even when tests are omitted.**Key Assertions:**

- {'name': 'pyproject.toml file exists and is not empty', 'expected': {'path': '/tmp/pyproject.toml', 'status': 0}, 'actual': {'path': '/tmp/pyproject.toml', 'status': 0}}
- {'name': 'pyproject.toml contains omit_tests_from_coverage section', 'expected': {'key': 'omit_tests_from_coverage', 'value': 'True'}, 'actual': {'key': 'omit_tests_from_coverage', 'value': 'True'}}

Confidence: 80%**Tokens:** 121 input + 183 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	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 1ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_param_value_max_chars

Why Needed: To ensure that the `param_value_max_chars` option is correctly loaded and used in the build process.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'max_chars = 100', 'actual': 'max_chars = 100'}
- {'name': 'param_value_max_chars value is an integer', 'expected': 100, 'actual': 100}

Confidence: 80%

Tokens: 119 input + 132 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	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

1ms



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_report_collect_only

Why Needed: To ensure that the 'collect' option is loaded correctly from pyproject.toml.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': 'Collect keyword is present in pyproject.toml', 'actual': 'Collect keyword is not present in pyproject.toml'}

Confidence: 80%**Tokens:** 116 input + 104 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	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 1ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_timeout_seconds

Why Needed: To ensure that the timeout seconds setting in pyproject.toml is properly loaded and used when loading dependencies.

Key Assertions:

- {'name': 'pyproject.toml exists', 'value': 'True'}
- {'name': 'timeout_seconds setting exists', 'value': 'True'}

Confidence: 80%

Tokens: 113 input + 104 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	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384-386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_batch_max_tests

4ms



3

AI ASSESSMENT

Scenario: test_load_batch_max_tests

Why Needed: To ensure that the `batch_max_tests` optimization is correctly applied to batched tests in Pytest.

Key Assertions:

- {'name': 'pyproject.toml file exists and is not empty', 'expected_value': 'True'}
- {'name': 'pyproject.toml file has a correct `optimizations` section', 'expected_value': "{'batch_max_tests': True}"}

Confidence: 80%

Tokens: 117 input + 114 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	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



AI ASSESSMENT

Scenario: load_batch_parametrized_tests**Why Needed:** Optimize PyProject token optimization for parametrized tests.**Key Assertions:**

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

Confidence: 80%**Tokens:** 123 input + 66 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	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 correctly loaded and used in the project.

Key Assertions:

- {'name': 'pyproject.toml file exists', 'expected': 'True'}
- {'name': 'context_compression directory exists', 'expected': 'True'}

Confidence: 80%

Tokens: 117 input + 103 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	130 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402-404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_context_line_padding

5ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_context_line_padding

Why Needed: To ensure that the context line padding is correctly loaded and applied to the test files.

Key Assertions:

- {'name': 'Context line padding is applied correctly', 'expected_value': 'True'}
- {'name': 'Context line padding is not applied to empty lines', 'expected_value': 'False'}

Confidence: 80%

Tokens: 117 input + 110 output = 227 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	130 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404-405, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_prompt_tier

4ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_prompt_tier

Why Needed: To ensure that the 'prompt_tier' is loaded correctly from the pyproject.toml file.**Key Assertions:**

- {'name': 'pyproject.toml exists and has a prompt_tier section', 'expected_value': 'True'}
- {'name': "pyproject.toml has a 'prompt_tier' section with the correct type", 'expected_value': 'prompt_tier: string'}

Confidence: 80%**Tokens:** 117 input + 130 output = 247 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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: The test validation with batch_max_tests < 1 is necessary because it checks if the configuration can be validated without any tests.

Key Assertions:

- {'name': 'batch_max_tests must be at least 1', 'description': "The 'batch_max_tests' field in the configuration should be set to a value greater than or equal to 1."}

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/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: Negative context_line_padding is not allowed.

Key Assertions:

- {'description': 'The context_line_padding must be 0 or positive', 'expected_value': '0'}

Confidence: 80%

Tokens: 129 input + 77 output = 206 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 ensure that the test suite covers all possible error messages when validating with invalid context compression.

Key Assertions:

- {'value': 'Invalid context_compression'}

Confidence: 80%

Tokens: 124 input + 73 output = 197 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 function correctly identifies and reports invalid `prompt_tier` values.

Key Assertions:

- {'name': 'Test case 1: Invalid prompt_tier value', 'description': 'The test case checks if the validation function returns an error message for an invalid `prompt_tier` value.', 'expected_result': 'Invalid prompt_tier'}
- {'name': 'Test case 2: Multiple errors for multiple invalid prompt_tier values', 'description': 'The test case checks if the validation function correctly reports multiple errors for different invalid `prompt_tier` values.', 'expected_result': ['Invalid prompt_tier']}

Confidence: 80%

Tokens: 125 input + 179 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	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-261, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

 tests/test_plugin_integration.py

14 tests

PASSED

tests/test_plugin_integration.py::TestPluginConfigLoading::test_config_defaults

3ms



3

AI ASSESSMENT

Scenario: tests/test_plugin_integration.py::TestPluginConfigLoading::test_config_defaults**Why Needed:** To ensure that the plugin configuration has safe defaults.**Key Assertions:**

- {'name': 'cfg' is an instance of Config', 'expected_type': 'Config'}

Confidence: 80%**Tokens:** 119 input + 72 output = 191 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	124 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-337, 340, 342, 344, 348, 352, 356, 360-362, 364, 366, 368, 372, 374, 378-380, 382, 384-386, 388, 390, 392, 396, 400, 402, 404, 408-410, 412-413, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460-461, 463-464, 466-467, 470, 472-473, 476-477, 482-488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603-605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_plugin_integration.py::TestPluginConfigLoading::test_markers_exist_in_config

1ms



AI ASSESSMENT

Scenario: tests/test_plugin_integration.py

Why Needed: Because the `TestPluginConfigLoading` test relies on the plugin's configuration being available.

Key Assertions:

- pytestconfig is not None

Confidence: 80%

Tokens: 108 input + 55 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)

PASSED

tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_both_json_and_html_outputs

94ms



8

AI ASSESSMENT

Scenario: Test generates both JSON and HTML reports for a simple test.**Why Needed:** This test prevents regression in cases where the report generation is not compatible with different output formats.**Key Assertions:**

- The `report.json` file should exist at the specified path.
- The `report.html` file should exist at the specified path.
- Both `report.json` and `report.html` files should be present in the report directory.
- The `report.json` file should contain valid JSON data.
- The `report.html` file should not contain any syntax errors or warnings.
- The test function `test_simple()` should return a successful assertion result.

Confidence: 80%**Tokens:** 279 input + 151 output = 430 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 59ms 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': 'Expected pytest_collection_finish counts items to return the correct count', 'description': 'The test should run successfully and report the correct number of tests.', 'expected_result': 3, 'actual_result': 0}

Confidence: 80%

Tokens: 198 input + 107 output = 305 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 57ms 7

AI ASSESSMENT

Scenario: Test that output directories are created if missing.**Why Needed:** This test prevents a regression where the plugin does not create nested directories for reports.**Key Assertions:**

- The `nested` directory should be created in the report.json file.
- The `report.json` file should exist.
- The `nested` directory should have been created using `pytester.makepyfile()`.
- The `report.json` file should contain a nested directory structure.
- The `test_pass()` function should not be executed inside the nested directory.
- The `report.json` file should not be empty after running `pytester.runpytest()`.

Confidence: 80%**Tokens:** 247 input + 149 output = 396 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

62ms



7

AI ASSESSMENT

Scenario: Test that fixture errors are captured in report.**Why Needed:** Fixture errors should be reported and logged for debugging purposes.**Key Assertions:**

- The 'summary' key in the report contains an error code of 1.
- The 'error' value in the 'summary' key is set to True.
- The fixture name 'broken_fixture' is used in the test.
- The pytester.runpytest command captures and logs fixture errors.
- The captured error message is included in the report JSON file.

Confidence: 80%**Tokens:** 286 input + 122 output = 408 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

170ms



7

AI ASSESSMENT

Scenario: Test pytest_runtst_makereport captures all outcomes to ensure accurate reporting.

Why Needed: This test prevents a potential issue where the report may not capture all outcomes, leading to inaccurate reporting and debugging difficulties.

Key Assertions:

- The 'passed' outcome should be present in the report.
- The 'failed' outcome should be present in the report.
- The 'skipped' outcome should be present in the report.
- All three types of outcomes ('passed', 'failed', and 'skipped') should be included in the report.
- The report path is correctly set to capture the output of pytest_runtst_makereport.
- The report format (json) is used as expected.

Confidence: 80%

Tokens: 335 input + 164 output = 499 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 54ms 3

AI ASSESSMENT

Scenario:

tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_no_report_when_disabled

Why Needed: To ensure that the plugin correctly handles cases where no output is specified.

Key Assertions:

- {'assertion_type': 'not exists', 'expected_result': 'True'}

Confidence: 80%

Tokens: 150 input + 78 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	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-

PASSED

tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_pdf_option_enables_plugin

595ms



8

AI ASSESSMENT

Scenario: Test that --llm-pdf option enables the plugin.**Why Needed:** To prevent a regression where the plugin is disabled due to missing Playwright configuration.**Key Assertions:**

- The `pytester.makepyfile` function should be called with a valid test file.
- The `test_pass()` function should be defined and return True.
- The `result.ret` attribute of the `pytester.runpytest` function should be equal to 0 (success) or warning.
- The `assert result.ret == 0` statement should not throw an error if the plugin is disabled.
- If the plugin was enabled, it should exit with code 0 (success) or warning, but definitely run.
- The `pytester.makepyfile` function should be called with a valid test file that includes the `--llm-pdf=report.pdf` option.

Confidence: 80%**Tokens:** 435 input + 195 output = 630 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 59ms 7

AI ASSESSMENT

Scenario: Test that the `pytest_sessionstart` hook records start time correctly when running with pytester.

Why Needed: This test prevents a potential bug where the `pytest_sessionstart` hook does not record the start time of the session.

Key Assertions:

- The 'start_time' key is present in the run meta data.
- The value of the 'start_time' key matches the actual start time recorded by pytest.
- The 'start_time' key is correctly formatted as a timestamp.
- The test function `test_pass()` does not affect the start time recording.
- The report generated by pytester includes the correct start time data.
- The start time is reported in seconds since the epoch.
- The start time is recorded at the beginning of each session.

Confidence: 80%

Tokens: 276 input + 176 output = 452 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_text_marker

1ms  2

AI ASSESSMENT

LLM error: Failed after 10 retries. Last error: timed out

COVERAGE

src/pytest_llm_report/collector.py

14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)

src/pytest_llm_report/plugin.py

6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_plugin_integration.py::TestPluginIntegration::test_llm_opt_out_marker

1ms  2

COVERAGE

src/pytest_llm_report/collector.py

14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)

src/pytest_llm_report/plugin.py

6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_plugin_integration.py::TestPluginIntegration::test_requirement_marker

1ms



AI ASSESSMENT

Scenario: tests/test_plugin_integration.py::TestPluginIntegration::test_requirement_marker

Why Needed: Because the requirement marker is causing a `TypeError` when it's used as a function.

Key Assertions:

- `{'assertion_type': 'TypeError', 'message': 'requirement marker should not cause errors.'}`

Confidence: 80%

Tokens: 90 input + 80 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)

PASSED

tests/test_plugin_integration.py::TestReportGeneration::test_report_writer_integration 41ms 6

AI ASSESSMENT

Scenario: Test the integration of report writer with pytest_llm_report.**Why Needed:** This test prevents regression that may occur when using the report writer to generate reports.**Key Assertions:**

- Verify that the report writer writes a valid JSON file.
- Verify that the report writer writes a valid HTML file containing all test names.
- Check if there are any assertion errors in the report.
- Verify that the total number of tests is correct (2 in this case).
- Verify that only 'test_a.py' and 'test_b.py' are included in the HTML report.
- Ensure the report writer can handle test failures with error messages.

Confidence: 80%**Tokens:** 417 input + 149 output = 566 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	81 lines (ranges: 190, 194-199, 201-203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	136 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-

327, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

tests/test_plugin_maximal.py

26 tests

PASSED

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_disabled 1ms ⚡ 2

COVERAGE

src/pytest_llm_report/collector.py

14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)

src/pytest_llm_report/plugin.py

10 lines (ranges: 558-559, 562, 566-568, 579-580, 586-587)

PASSED

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_enabled 2ms ⚡ 2

AI ASSESSMENT

Scenario: TestPluginCollectReport

Why Needed: To test that the collectreport function calls the collector when enable is True.

Key Assertions:

- {'name': 'mock_collector.handle_collection_report was called once with mock_report as argument', 'expected': 1, 'actual': 0}

Confidence: 80%

Tokens: 204 input + 81 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/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: This test is needed because the `pytest_collectreport` function should not raise an exception when a session is not available.

Key Assertions:

- {'name': 'pytest_collectreport()' was called with no arguments', 'description': 'The pytest_collectreport() function was called with no arguments. This indicates that the function did not receive any arguments, which is unexpected behavior.'}

Confidence: 80%

Tokens: 138 input + 119 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/plugin.py	8 lines (ranges: 558-559, 562, 566-568, 579, 583)

PASSED

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_session_none 1ms 2

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_session_none

Why Needed: To ensure the plugin correctly handles a null session in Pytest.

Key Assertions:

- {'name': 'mock_report.session is None', 'expected': '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: Llama model is not compatible with PyTorch, and this warning is raised to inform the user.

Key Assertions:

- {'name': 'LLM model compatibility', 'description': 'The LLM model used by the plugin is not compatible with PyTorch. This raises a warning.'}

Confidence: 80%

Tokens: 143 input + 104 output = 247 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	136 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-325, 327-328, 332-336, 340, 342, 344, 348, 352, 356, 360-362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	30 lines (ranges: 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362-364, 366-367, 371-373, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_validation_errors 3ms 3

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_validation_errors

Why Needed: Validation errors are raised when the pytest configuration is invalid.

Key Assertions:

- {'name': 'pyproject.toml validation error', 'message': 'Invalid pyproject.toml file'}

Confidence: 80%

Tokens: 134 input + 80 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/options.py	135 lines (ranges: 123, 171, 199, 202-205, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	25 lines (ranges: 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-358, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_worker_skip

1ms

2

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_worker_skip

Why Needed: To ensure that the configure function skips on xdist workers as expected.

Key Assertions:

- {'assertion_type': 'mocking', 'expected_result': 'mock_config.addinivalue_line.called', 'actual_result': 'True'}

Confidence: 80%

Tokens: 170 input + 92 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	17 lines (ranges: 328-330, 332-334, 336-338, 342-343, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginConfigureFallback::test_pytest_configure_fallback_load

3ms



2

AI ASSESSMENT

Scenario: Test that fallback to load_config is triggered when Config.load is missing.

Why Needed: To prevent a crash due to missing Config.load method.

Key Assertions:

- The Config.validate() method should return an empty list.
- The load_config() method of pytest_llm_report.options.Config should be called once.
- The mock_load.assert_called_once() assertion should pass with the correct arguments.

Confidence: 80%

Tokens: 747 input + 97 output = 844 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	30 lines (ranges: 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362-364, 366-367, 371-373, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginLoadConfig::test_load_config
_cli_overrides_pyproject 2ms 3

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginLoadConfig::test_load_config_cli_overrides_pyproject

Why Needed: To test the functionality of CLI options overriding pyproject.toml options in plugin load configuration.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'pyproject.toml should contain some overrides'}
- {'name': 'load_config function call', 'expected': 'load_config function should be called with pyproject.toml as argument'}

Confidence: 80%

Tokens: 140 input + 124 output = 264 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 91ms 3

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginLoadConfig::test_load_config_from_pyproject

Why Needed: To ensure that the plugin can load configuration from a PyProject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists', 'expected': True, 'message': 'PyProject.toml should exist in the test directory'}

Confidence: 80%

Tokens: 136 input + 95 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	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



AI ASSESSMENT

Scenario: Test that terminal summary skips when plugin is disabled.**Why Needed:** This test prevents a regression where the plugin's terminal summary function does not skip when the plugin is disabled.**Key Assertions:**

- The `pytest_terminal_summary` function should return None when called with an invalid stash value (in this case, False).
- The `stash.get` method of the mock configuration object was called once with a False argument.
- The `enabled_key` variable in the mock configuration object is not set to True.

Confidence: 80%**Tokens:** 281 input + 121 output = 402 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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



COVERAGE

src/pytest_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: Verify that the `load_config` function correctly loads config options from pytest objects (CLI) into a mock configuration object.

Why Needed: This test prevents regression in the `load_config` function, which is used to load configuration options from pytest objects (CLI), potentially causing issues with plugin functionality.

Key Assertions:

- The `report_html` option of the loaded config is set to 'out.html'.
- The `llm_report_json` and `llm_report_pdf` options are not set in the loaded config.
- The `llm_evidence_bundle`, `llm_dependency_snapshot`, `llm_requests_per_minute`, `llm_aggregate_dir`, `llm_aggregate_policy`, `llm_aggregate_run_id`, `llm_aggregate_group_id`, and `llm_max_retries` options are not set in the loaded config.
- The `llm_coverage_source`, `llm_prompt_tier`, `llm_batch_parametrized`, `llm_context_compression`, `llm_context_bytes`, `llm_context_file_limit`, `llm_max_tests`, and `llm_max_concurrency` options are not set in the loaded config.
- The `llm_timeout_seconds` option is set to a non-zero value.
- The `llm_capture_failed` option is not set in the loaded config.
- The `llm_ollama_host`, `llm_llm_api_base`, and `llm_llm_token_refresh_command` options are not set in the loaded config.
- The `llm_llm_api_key` option is not set in the loaded config.
- The `llm_llm_token_output_format` option is not set in the loaded config.
- The `llm_llm_token_json_key` option is not set in the loaded config.
- The `llm_cache_dir` and `llm_cache_ttl` options are not set in the loaded config.
- The `llm_metadata_file` option is not set in the loaded config.
- The `llm_hmac_key_file` option is not set in the loaded config.
- The `include_params` option is not set in the loaded config.
- The `strip_docstrings` option is not set in the loaded config.

Confidence: 80%

Tokens: 639 input + 498 output = 1137 total

COVERAGE

src/pytest_llm_report/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 to ensure that makereport skips when disabled.

Key Assertions:

- {'name': 'mock_item.config.stash.get returns False', 'expected_value': False}
- {'name': 'gen.send returns StopIteration', 'expected_value': True}

Confidence: 80%

Tokens: 220 input + 105 output = 325 total

COVERAGE

src/pytest_llm_report/collector.py

14 lines (ranges: 90, 93, 96,
99, 110-112, 114-115, 124,
127, 140, 209-210)

src/pytest_llm_report/plugin.py

7 lines (ranges: 558-559,
562-563, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginRuntest::test_runtest_makereport_enabled

2ms



AI ASSESSMENT

Scenario: Test makereport calls collector when enabled.

Why Needed: Prevents a potential bug where the plugin does not call the collector even if makereport is enabled.

Key Assertions:

- The `pytest_runtest_makereport` function should be able to find and stash the `mock_collector` instance.
- The `stash_get` method of the mock item's config should return True when `_enabled_key` is present.
- The `stash_get` method of the mock item's config should return `mock_collector` when `_collector_key` is present.
- The `handle_runtest_logreport` method of the mock collector instance should be called with `mock_report` and `mock_item` as arguments.
- The `get_result` method of the mock report object should return an empty result if no outcome was yielded.
- No exceptions should be raised when calling `send` on the mock outcome.
- The mock item's stash get method should not raise an exception when `_enabled_key` is present.

Confidence: 80%

Tokens: 371 input + 230 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_collection_finish_disabled

1ms 2

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_collection_finish_disabled

Why Needed: The test is necessary to verify that the `pytest_collection_finish` hook skips collection when disabled.

Key Assertions:

- {'name': 'mock_session.config.stash.get.return_value', 'expected': {'False': {}}}

Confidence: 80%

Tokens: 149 input + 88 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	8 lines (ranges: 558-559, 562, 566-568, 602-603)

PASSED

tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_collection_finish_enabled

2ms 2

AI ASSESSMENT

Scenario: TestPluginSessionHooks

Why Needed: To ensure that the collector is called when collection_finish is enabled.

Key Assertions:

- {'name': 'mock_collector.handle_collection_finish was called once with mock_session.items', 'expected_call_count': 1}

Confidence: 80%

Tokens: 219 input + 73 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/plugin.py	10 lines (ranges: 558-559, 562, 566-568, 602, 606-608)

PASSED

tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_sessionstart_disabled

1ms

2

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_sessionstart_disabled

Why Needed: The test is currently disabled and needs to be enabled for it to run.

Key Assertions:

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

Confidence: 80%

Tokens: 157 input + 81 output = 238 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	8 lines (ranges: 558-559, 562, 566-568, 619-620)

PASSED

tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_sessionstart_enabled

1ms

3

AI ASSESSMENT

Scenario: Test that sessionstart initializes collector when enabled and stash is properly populated.

Why Needed: This test prevents a potential regression where the collector might not be created or initialized correctly when pytest_sessionstart is called with an enabled configuration.

Key Assertions:

- assert _collector_key in mock_stash
- assert _start_time_key in mock_stash

Confidence: 80%

Tokens: 335 input + 87 output = 422 total

COVERAGE

src/pytest_llm_report/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

2ms



2

AI ASSESSMENT

Scenario: Test pytest_addoption adds expected arguments to the parser.**Why Needed:** This test prevents a potential bug where pytest_addoption does not add all 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 + 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/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 2ms 2

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_pytest_addoption_no_ini

Why Needed: pytest_addoption no longer adds INI options in pytest 7.0 and later

Key Assertions:

- {'name': 'parser.addini was not called', 'expected_result': [], 'actual_result': 'parser.addini was called'}

Confidence: 80%

Tokens: 140 input + 95 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/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



AI ASSESSMENT

Scenario: Test coverage percentage calculation logic for `pytest_llm_report.plugin.pytest_terminal_summary`.

Why Needed: This test prevents regression in the coverage percentage calculation logic, ensuring that the plugin correctly calculates the terminal summary coverage.

Key Assertions:

- The `report_html` option is set to 'out.html' and the `CoverageMapper` class is instantiated with this value.
- The `Coverage` instance is created with a mock `report` method that returns a percentage of 85.5.
- The `load` method of the `Coverage` instance is called once, indicating successful loading of the coverage data.
- The `report` method of the `Coverage` instance is called once, indicating successful calculation of the coverage percentage.
- The `stash` dictionary passed to the `pytest_terminal_summary` function contains a key-value pair with `_enabled_key` and `_config_key` set to 'True' and 'cfg', respectively.
- The mock configuration object `mock_config` has its `workerinput` attribute deleted, ensuring that it is not used in the test.
- The mock stash dictionary passed to the `pytest_terminal_summary` function contains a key-value pair with `_enabled_key` set to 'True' and `_config_key` set to 'cfg', indicating correct configuration of the plugin.

Confidence: 80%

Tokens: 395 input + 285 output = 680 total

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 277, 285)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	53 lines (ranges: 399, 403, 407, 410, 429-430, 432, 434, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-466, 468, 470-473, 485-486, 491-492, 534-544, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_terminal_summary_llm_enabled

3ms



AI ASSESSMENT

Scenario: Test terminal summary with LLM enabled runs annotations.**Why Needed:** Prevents regression in plugin behavior when LLM is enabled.**Key Assertions:**

- Verify that the `pytest_terminal_summary_llm_enabled` test passes without any errors.
- Check if the `mock_annotate` function is called once with the correct configuration.
- Verify that the `stash` dictionary contains the expected keys and values.
- Confirm that the `mock_config.stash` dictionary matches the original stash.
- Test that the `pytest_terminal_summary` function does not raise any exceptions.
- Ensure that the `mock_writer_cls.return_value` is set to the mock writer instance.
- Verify that the `mock_provider.get_model_name.return_value` and `mock_get_provider.return_value` functions return the expected values.
- Check if the `pytest_terminal_summary` function does not raise any exceptions when called with a valid configuration.

Confidence: 80%**Tokens:** 477 input + 204 output = 681 total

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 277, 285)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	66 lines (ranges: 399, 403, 407, 410, 429-430, 432, 434, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485-486, 491-494, 497, 499, 502-504, 512-514, 516, 523-531, 534-544, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_terminal_summary_no_collector

2ms



3

AI ASSESSMENT

Scenario: Test terminal summary creates collector if missing.

Why Needed: This test prevents a potential regression where the plugin does not create a collector even when it is supposed to be present in the configuration.

Key Assertions:

- mock_terminalreporter.call_args_list[0][1] should return 'Collector' as expected.
- stash._enabled_key should be True.
- stash._config_key should have the correct value (pytest_llm_report.plugin.Config).
- mock_config.stash.mock_stash._enabled_key should be True.
- mock_config.stash.mock_stash._config_key should have the correct value (pytest_llm_report.plugin.Config).
- mock_terminalreporter.call_args_list[0][1] should return 'Collector' as expected.
- stash._enabled_key should be False after calling pytest_terminal_summary() with mock_config.

Confidence: 80%

Tokens: 391 input + 198 output = 589 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_termin
al_summary_with_aggregation

3ms



AI ASSESSMENT

Scenario: Test terminal summary with aggregation enabled.**Why Needed:** This test prevents regression in the case where aggregation is enabled and there are multiple terminals being reported.**Key Assertions:**

- The aggregated report should be available in both `out.html` and `out.json` files.
- The aggregation function should return a report object.
- The write_json method of the ReportWriter class should be called with the aggregated report.
- The write_html method of the ReportWriter class should be called with the aggregated report.
- The aggregation function should not raise an error if there are no terminals being reported.
- The write_json and write_html methods should only be called once each, even if multiple terminals are being reported.
- The stash object should contain both _enabled_key and _config_key keys with the correct values.
- The stash object should not have a workerinput key.
-

Confidence: 80%**Tokens:** 441 input + 200 output = 641 total

COVERAGE

src/pytest_llm_report/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 during terminal summary generation.**Why Needed:** This test prevents a regression where the coverage calculation fails due to an OSError caused by disk full.**Key Assertions:**

- The `pytest_terminal_summary` function should not raise a UserWarning when computing coverage percentage.
- The `coverage.Coverage.load()` method should be called before attempting to compute coverage percentage.
- The `coverage.Coverage` object should have been loaded successfully without raising an OSError.
- The `pytest_terminal_summary` function should not log the error message 'Failed to compute coverage percentage' as a warning.
- The `pytest_terminal_summary` function should correctly handle cases where coverage data is not available (e.g., due to disk full)
- The `pytest_terminal_summary` function should not raise an exception when computing coverage percentage for non-existent files or directories
- The `pytest_terminal_summary` function should log the error message 'Failed to compute coverage percentage' in a meaningful way, including the file path and line number where the error occurred.
- The `pytest_terminal_summary` function should correctly handle cases where the coverage data is not available for all lines (e.g., due to disk full)

Confidence: 80%**Tokens:** 389 input + 261 output = 650 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: Assembling a balanced context for test_a.py with utils.py dependency

Why Needed: This test prevents a potential bug where the ContextAssembler is unable to assemble a balanced context due to missing dependencies.

Key Assertions:

- The 'utils.py' file should be present in the assembled context.
- The 'def util()' function should be found in the 'utils.py' file within the assembled context.
- The assembly result should include a coverage entry for the 'utils.py' file with line ranges and count.
- The test result nodeid should match the one specified in the test file.
- The outcome of the test should be 'passed'.
- The assembler should be able to assemble a balanced context without any issues.
- The assembler should be able to find dependencies like 'utils.py' within the assembled context.

Confidence: 80%

Tokens: 331 input + 187 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/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 ensure that the ContextAssembler can assemble a complete context for a test file.

Key Assertions:

- {'assertion_type': 'contains', 'value': 'test_1'}

Confidence: 80%

Tokens: 176 input + 77 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	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



AI ASSESSMENT

Scenario: Test the ContextAssembler with minimal context mode and a test file.

Why Needed: This test prevents regression when using minimal context mode for tests that require minimal setup.

Key Assertions:

- The 'test_1' function is present in the source code of the test file.
- The context object returned by the ContextAssembler does not contain any information about the test file.
- The test result nodeid matches the expected location of the test function.
- The test file's source code contains an assert statement that passes when run with minimal context mode.

Confidence: 80%

Tokens: 267 input + 129 output = 396 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	30 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 116)

PASSED

tests/test_prompts.py::TestContextAssembler::test_balanced_context_limits 1ms 5

AI ASSESSMENT

Scenario: Test the ContextAssembler with balanced context limits to ensure it correctly truncates long content when exceeding the limit.

Why Needed: This test prevents regression by ensuring that the ContextAssembler does not silently truncate long code snippets without providing a clear indication of what was truncated.

Key Assertions:

- The 'f1.py' file in the context is truncated to 40 bytes or less, as expected.
- A message indicating truncation ('... truncated') is present in the 'f1.py' file.
- The length of the 'f1.py' file is within the allowed limit (20 bytes + truncation message).

Confidence: 80%

Tokens: 335 input + 143 output = 478 total

COVERAGE

src/pytest_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: Tests the ContextAssembler with complete mode and large context file to ensure it does not truncate files exceeding a certain size.

Why Needed: This test prevents regression where the ContextAssembler truncates files larger than the specified context limit in complete mode.

Key Assertions:

- The 'f1.py' node should be present in the assembled context.
- The content of 'f1.py' should not be truncated despite exceeding the 20 byte context limit.
- No 'truncated' key should be present in the context of 'f1.py'.

Confidence: 80%

Tokens: 361 input + 128 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/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	50 lines (ranges: 33, 49, 52, 55, 58, 60, 63, 65, 78-79, 82-84, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-193, 196-197, 201, 268-272, 284-285, 287)

PASSED

tests/test_prompts.py::TestContextAssembler::test_get_test_source_edge_cases

1ms



AI ASSESSMENT

Scenario: Test the ContextAssembler with edge cases, specifically non-existent files and nested test names with parameters.

Why Needed: This test prevents a potential bug where the ContextAssembler fails to extract test sources from non-existent or deeply nested test files.

Key Assertions:

- The method `'_get_test_source'` returns an empty string when given a file path that does not exist.
- The method `'_get_test_source'` correctly extracts the test name and parameters from a nested test file with multiple levels of nesting.

Confidence: 80%

Tokens: 275 input + 116 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)
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 being processed.

Why Needed: This test prevents a potential bug where some important files are inadvertently excluded during processing.

Key Assertions:

- The file 'test.pyc' is expected to be excluded because it has a .pyc extension.
- The file 'secret/key.txt' is expected to be excluded because of the presence of '*'.
- The file 'public/readme.md' should not be excluded as it does not contain any sensitive information.

Confidence: 80%

Tokens: 227 input + 122 output = 349 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	1 lines (ranges: 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	5 lines (ranges: 33, 284-287)

tests/test_prompts_coverage.py

12 tests

PASSED

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_assemble_minimal_mode 1ms 4

AI ASSESSMENT

Scenario: Test assemble minimal mode returns no context files.**Why Needed:** Prevents regression where assemble function does not generate any context files.**Key Assertions:**

- context_files is an empty list when config llm_context_mode is set to 'minimal'.
- test_source contains the expected function definition.
- Context files are generated only if test_foo.py has a __file__ attribute and it's not empty.
- The assemble function does not modify or create any context files in this case.

Confidence: 80%**Tokens:** 298 input + 117 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	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	29 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-109, 111-112, 116)

PASSED

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_assemble_with_context_override 1ms 5

AI ASSESSMENT

Scenario: Test assemble respects llm_context_override from test.**Why Needed:** This test prevents regression by ensuring that the ContextAssembler uses the correct LLM context mode even when an override is specified.**Key Assertions:**

- The assembler should use balanced mode due to the override.
- The module file path should be included in the context files.
- The coverage entry for the module file should have a line range of '1' and a line count of '1'.
- The test source code should be included in the context files.
- The context files should not contain any other test files or modules.
- The coverage entry for the module file should not include any other lines than specified (i.e., only '1').
- The LLM context override should be balanced, i.e., neither minimal nor default mode should be used.

Confidence: 80%**Tokens:** 362 input + 189 output = 551 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the test_balanced_context_excludes_patterns function to ensure it excludes files matching exclude patterns in a balanced context.

Why Needed: This test prevents regression where the LLM context mode is set to 'balanced' and the assembler does not exclude files that match the specified patterns.

Key Assertions:

- The file "secret_config.py" should be excluded from the balanced context.
- The file 'api_key =
- The line count of the excluded file should be 1.
- The line ranges for the excluded file should start at line 1 and end at line 1.
- The coverage entry for the excluded file should have a line range of '1' and a line count of 1.
- The test should fail if the LLM context mode is set to 'balanced' without excluding any files.
- The test should pass if the LLM context mode is set to 'balanced' with the specified exclude patterns in place.

Confidence: 80%

Tokens: 331 input + 213 output = 544 total

COVERAGE

src/pytest_llm_report/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 test the scenario where a balanced context file does not exist.

Key Assertions:

- {'message': 'Expected an empty context to be returned when a non-existent file is provided.', 'expected_result': {}, 'actual_result': {}}

Confidence: 80%

Tokens: 201 input + 93 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/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	16 lines (ranges: 33, 139, 142-145, 147-148, 152-153, 155-156, 159-161, 201)

PASSED

`tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_max_bytes_limit` 14ms 5

AI ASSESSMENT

Scenario: Test that balanced context respects max bytes limit.

Why Needed: This test prevents a potential memory leak or incorrect coverage due to an overly large source file being assembled into the context.

Key Assertions:

- The content of the source file is truncated to prevent excessive memory usage.
- A message indicating that the content was truncated is included in the context.
- The source file's length does not exceed the specified limit when assembling it into the context.
- The test verifies that the source file's content is truncated, preventing potential issues with coverage and memory consumption.
- The test ensures a clear indication of truncation in the context, facilitating debugging and maintenance efforts.

Confidence: 80%

Tokens: 405 input + 151 output = 556 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	34 lines (ranges: 33, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-194, 196-197, 201, 284-285, 287)

PASSED

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_no_coverage 1ms 4

AI ASSESSMENT

Scenario:

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_no_coverage

Why Needed: To ensure that the ContextAssembler can correctly assemble a balanced context with no coverage.

Key Assertions:

- {'assertion_type': 'assert context is empty', 'expected_result': '{}', 'actual_result': '{}'}

Confidence: 80%

Tokens: 162 input + 89 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	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



AI ASSESSMENT

Scenario: Test that loop exits when max bytes is reached before processing file.

Why Needed: This test prevents a potential bug where the ContextAssembler exceeds the maximum allowed bytes in a balanced context without encountering any files.

Key Assertions:

- context should have only one node (either 'file1.py' or 'file2.py')
- should not exceed the maximum allowed bytes of 5 for 'file1.py'
- should be empty if no files are processed
- should contain both 'content1 = 1' and 'content2 = 2' from both files
- should have only one line range (1) with a single line count (1)
- should not exceed the maximum allowed bytes of 5 for 'file2.py'
- should be empty if no files are processed

Confidence: 80%

Tokens: 409 input + 183 output = 592 total

COVERAGE

src/pytest_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: To ensure that the complete context delegates to balanced correctly.

Key Assertions:

- {'assertion_type': 'inclusion', 'expected_inclusion': ['module.py'], 'actual_inclusion': ['module.py']}

Confidence: 80%

Tokens: 211 input + 90 output = 301 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 cases where the input nodeid is empty.

Key Assertions:

- `{'name': 'assert result == "", 'expected_value': '', 'message': 'Expected _get_test_source with empty nodeid to return an empty string'}`

Confidence: 80%

Tokens: 148 input + 92 output = 240 total

COVERAGE

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



AI ASSESSMENT

Scenario: Tests that source extraction stops at next function definition.

Why Needed: To ensure correct source code is extracted and to prevent errors due to incomplete or incorrect source code.

Key Assertions:

- {'assertion_type': 'Source code extraction should stop at the next function definition', 'expected_result': 'The source code for the current test file should be extracted until the next function definition.', 'actual_result': 'The source code for the entire test file is extracted.'}

Confidence: 80%

Tokens: 129 input + 114 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	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/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_file_not_e

Why Needed: To test the _get_test_source method's behavior when a non-existent file is provided as an argument.

Key Assertions:

- {'name': 'assert result is empty', 'expected_result': '', 'message': 'Expected result to be an empty string'}

Confidence: 80%

Tokens: 142 input + 98 output = 240 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/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: The 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 add(a, b):', ' return a + b']}, 'actual': {'indentation': 2, 'expected_lines': ['def add(a, b):', ' return a + b']}}`

Confidence: 80%

Tokens: 118 input + 128 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)

 tests/test_ranges.py

13 tests

PASSED

tests/test_ranges.py::TestCompressRanges::test_consecutive_lines

1ms  3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_consecutive_lines

Why Needed: To ensure consecutive lines are compressed correctly.

Key Assertions:

- {'expected': '1-3', 'actual': '1-3'}

Confidence: 80%

Tokens: 106 input + 68 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	12 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65, 67)

PASSED

tests/test_ranges.py::TestCompressRanges::test_duplicates

1ms  3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_duplicates

Why Needed: To test the handling of duplicate ranges in the compress_ranges function.

Key Assertions:

- {'description': 'The output should contain a single range string.', 'expected_value': '1-3'}

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/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 current implementation of `compress_ranges` does not handle an empty input list correctly.**Key Assertions:**

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

Confidence: 80%**Tokens:** 92 input + 70 output = 162 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	2 lines (ranges: 29-30)

PASSED

tests/test_ranges.py::TestCompressRanges::test_mixed_ranges

1ms  3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_mixed_ranges**Why Needed:** To test the functionality of compressing mixed ranges (single numbers and overlapping ranges).**Key Assertions:**

- {'value': '1-3, 5, 10-12, 15'}

Confidence: 80%**Tokens:** 130 input + 79 output = 209 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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': '1, 3, 5', 'actual': '1, 3, 5'}

Confidence: 80%

Tokens: 113 input + 78 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/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: To ensure that the single line case does not use range notation.

Key Assertions:

- {'expected_value': '5', 'actual_value': '5'}

Confidence: 80%

Tokens: 96 input + 70 output = 166 total

COVERAGE

src/pytest_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



AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_two_consecutive**Why Needed:** To ensure that two consecutive lines are compressed to a single range.**Key Assertions:**

- {'expected': '1-2', 'actual': '1-2'}

Confidence: 80%**Tokens:** 103 input + 73 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/ranges.py	12 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65, 67)

PASSED

tests/test_ranges.py::TestCompressRanges::test_unsorted_input

1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_unsorted_input**Why Needed:** The test is necessary because the current implementation of `compress_ranges` only works with sorted input.**Key Assertions:**

- {'expected': '1-3, 5', 'actual': '1-3, 5'}

Confidence: 80%**Tokens:** 110 input + 85 output = 195 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	16 lines (ranges: 29, 33, 35-37, 39-40, 42, 45-47, 50, 52, 65-67)

PASSED

tests/test_ranges.py::TestExpandRanges::test_empty_string

1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_empty_string**Why Needed:** The function `expand_ranges` is expected to handle an empty input string correctly.**Key Assertions:**

- {'assertion': "expand_ranges('') == []", 'expected_result': [], 'actual_result': []}

Confidence: 80%**Tokens:** 90 input + 81 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/ranges.py	2 lines (ranges: 81-82)

PASSED

tests/test_ranges.py::TestExpandRanges::test_mixed

1ms  3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_mixed**Why Needed:** To handle mixed ranges and singles in the input string.**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 + 107 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/ranges.py	11 lines (ranges: 81, 84-91, 93, 95)

PASSED

tests/test_ranges.py::TestExpandRanges::test_range

1ms  3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_range**Why Needed:** The range function should expand to a list.**Key Assertions:**

- {'expected': [1, 2, 3], 'actual': ['1', '2', '3']}

Confidence: 80%**Tokens:** 99 input + 74 output = 173 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	10 lines (ranges: 81, 84-91, 95)

PASSED

tests/test_ranges.py::TestExpandRanges::test_roundtrip

1ms  3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_roundtrip**Why Needed:** To ensure that `compress_ranges` and `expand_ranges` are inverses.**Key Assertions:**

- {'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 + 114 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/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 return a list containing the input value when given a single number as an argument.**Key Assertions:**

- {'expected': [5], 'actual': ['5']}

Confidence: 80%**Tokens:** 95 input + 79 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	7 lines (ranges: 81, 84-87, 93, 95)

PASSED

tests/test_render.py::TestFormatDuration::test_milliseconds

1ms

3

AI ASSESSMENT

Scenario: Test the format_duration function with duration values less than 1s.**Why Needed:** This test prevents a regression where the function does not correctly format durations for values < 1s.**Key Assertions:**

- The function should return '500ms' when given a duration of 0.5 seconds.
- The function should return '1ms' when given a duration of 0.001 seconds.
- The function should return '0ms' when given a duration of 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:**

- {'expected': '1.23s', 'actual': '1.23s'}
- {'expected': '60.00s', 'actual': '60.00s'}

Confidence: 80%**Tokens:** 116 input + 97 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/render.py	2 lines (ranges: 65-66)

PASSED

tests/test_render.py::TestOutcomeToCssClass::test_all_outcomes

1ms



AI ASSESSMENT

Scenario: All outcomes should map to CSS classes.

Why Needed: This test prevents a potential CSS class mapping issue where 'xfailed' and 'xpassed' are mapped to different classes.

Key Assertions:

- outcome_to_css_class('passed') == 'outcome-passed'
- outcome_to_css_class('failed') == 'outcome-failed'
- outcome_to_css_class('skipped') == 'outcome-skipped'
- outcome_to_css_class('xfailed') == 'outcome-xfailed'
- outcome_to_css_class('xpassed') == 'outcome-xpassed'

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/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:** The test is necessary because the `outcome_to_css_class` function does not handle unknown outcomes correctly.**Key Assertions:**

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

Confidence: 80%**Tokens:** 102 input + 81 output = 183 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	8 lines (ranges: 79-85, 87)

PASSED

tests/test_render.py::TestRenderFallbackHtml::test_renders_basic_report

1ms



AI ASSESSMENT

Scenario: Test renders basic report with fallback HTML.

Why Needed: Prevents rendering of incomplete or malformed reports due to missing or incorrect plugin and repository metadata.

Key Assertions:

- The presence of the '' header in the rendered HTML document.
- The inclusion of 'Test Report' in the rendered HTML document.
- The presence of 'PASSED' and 'FAILED' text in the rendered HTML document.
- The correct display of plugin and repository metadata in the rendered HTML document.

Confidence: 80%

Tokens: 426 input + 119 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/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)

AI ASSESSMENT

Scenario: Test renders coverage for fallback HTML generation.

Why Needed: Prevents regression in rendering coverage information when fallback is used.

Key Assertions:

- The report includes the source file 'src/foo.py' as part of the rendered HTML.
- The assertion checks that at least 5 lines were included in the rendered HTML.
- The test verifies that the 'src/foo.py' file is present in the rendered HTML.
- The test ensures that exactly 5 lines are included in the rendered HTML.
- The report includes coverage information for the specified files.
- The assertion checks that the coverage information is accurate and complete.

Confidence: 80%

Tokens: 288 input + 144 output = 432 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 issues that could cause the 'Tests login flow' and 'Prevents auth bypass' scenarios to be missed.

Key Assertions:

- The report includes the specified LLM annotations with confidence scores.
- The 'Tests login flow' scenario is included in the rendered HTML.
- The 'Prevents auth bypass' scenario is included in the rendered HTML.
- The confidence score for each annotation is displayed correctly (85%).

Confidence: 80%

Tokens: 317 input + 132 output = 449 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: Verifies that the source coverage summary is included in the rendered HTML report.

Why Needed: This test prevents a regression where the source coverage summary was not displayed correctly due to missing or incomplete code coverage data.

Key Assertions:

- The 'Source Coverage' section should be present in the rendered HTML report.
- The 'src/foo.py' file path should be included in the 'Source Coverage' section.
- The percentage of covered source code (80.0%) should be displayed in the 'Source Coverage' section.
- The range of missed source code (5-10) should be included in the 'Missed Source Code' section.
- The ranges of covered and missed source code should match the corresponding coverage percentages.
- The 'covered_ranges' and 'missed_ranges' values should contain the expected ranges for each file.

Confidence: 80%

Tokens: 331 input + 189 output = 520 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	68 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-127, 131-132, 155-156, 159-167, 172-178, 180-186, 191, 206, 210-211, 224, 257-264, 267, 269, 271-277, 280-281, 285)

PASSED

tests/test_render.py::TestRenderFallbackHtml::test_renders_xpass_summary

1ms

 3

AI ASSESSMENT

Scenario: Test renders xpass summary with fallback HTML.**Why Needed:** Prevents regression when rendering summary without fallback.**Key Assertions:**

- The 'XFailed' and 'XPassed' tags should be present in the rendered HTML.
- Both 'XFailed' and 'XPassed' tags should contain the corresponding text.
- If either tag is missing, the test should fail with an error message indicating the missing tag.
- The 'xfailed' and 'xpassed' values should match the actual counts from the report.
- The 'xpass' value should be present but not exceed the total count.
- The 'xfail' value should be absent or equal to 0 if it's supposed to be missing.
- The HTML structure should maintain a consistent layout and formatting.

Confidence: 80%**Tokens:** 283 input + 181 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/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: Because different content should produce different hashes.

Key Assertions:

- {'name': 'Different Content', 'expected_result': {'hash1': '...', 'hash2': '...'}}}

Confidence: 80%

Tokens: 115 input + 77 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/report_writer.py	1 lines (ranges: 55)

PASSED

tests/test_report_writer.py::TestComputeSha256::test_empty_bytes

1ms 3

AI ASSESSMENT

Scenario: tests/test_report_writer.py::TestComputeSha256::test_empty_bytes

Why Needed: To ensure that an empty bytes object produces consistent hash.

Key Assertions:

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

Confidence: 80%

Tokens: 129 input + 63 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/report_writer.py	1 lines (ranges: 55)

AI ASSESSMENT

Scenario: Test that the build_run_meta method includes version info and other relevant metadata.

Why Needed: This test prevents regression where the report writer does not include version information or plugin versions in the run meta.

Key Assertions:

- assert meta.duration == 60.0
- assert meta.pytest_version
- assert meta.plugin_version == __version__
- assert meta.python_version

Confidence: 80%

Tokens: 318 input + 98 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	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 counts all outcome types correctly.

Why Needed: This test prevents a regression where the total count of outcomes is incorrect when there are multiple skipped tests.

Key Assertions:

- The total number of outcomes should be 6 (1 passed, 1 failed, 1 skipped, 1 xfailed, 1 xpassed, 1 error).
- The number of passed outcomes should be 1.
- The number of failed outcomes should be 1.
- The number of skipped outcomes should be 1.
- The number of xfailed and xpassed outcomes should be 1 each.

Confidence: 80%

Tokens: 336 input + 149 output = 485 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: The test verifies that the `total` attribute of a `Summary` object is correctly calculated by summing up all outcomes.

Why Needed: This test prevents regression where the total count of passed, failed and skipped tests might not match the actual number of tests run.

Key Assertions:

- The total count should be equal to the sum of passed, failed and skipped counts.
- The passed count should be equal to the number of tests with outcome 'passed'.
- The failed count should be equal to the number of tests with outcome 'failed'.
- The skipped count should be equal to the number of tests with outcome 'skipped'.

Confidence: 80%

Tokens: 283 input + 150 output = 433 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the `ReportWriter` class initializes correctly.

Why Needed: This test prevents a potential bug where the `ReportWriter` instance does not have access to its configuration and warnings.

Key Assertions:

- The `config` attribute of the `writer` should be set to the provided `Config` object.
- The `warnings` list of the `writer` should be empty.
- The `artifacts` list of the `writer` should be empty.

Confidence: 80%

Tokens: 199 input + 113 output = 312 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	3 lines (ranges: 156-158)

PASSED

tests/test_report_writer.py::TestReportWriter::test_write_report_assembles_tests

5ms



4

AI ASSESSMENT

Scenario: Test 'ReportWriter::test_write_report_assembles_tests' verifies that the ReportWriter class writes a report with all tests.

Why Needed: This test prevents regression in cases where the output paths of the tests are not specified, causing the report to be incomplete or missing.

Key Assertions:

- The length of the report.tests list should be equal to 2.
- The total number of tests in the summary should be 2.
- All tests should have a 'passed' outcome.
- Each test should have a unique nodeid.
- The ReportWriter class should not write any files if no output paths are specified.
- The report.summary.total property should return 2 even if there are less than 2 tests.

Confidence: 80%

Tokens: 255 input + 167 output = 422 total

COVERAGE

src/pytest_llm_report/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

6ms



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 its report.

Key Assertions:

- {'expected_value': 85.5, 'actual_value': 'report.summary.coverage_total_percent'}

Confidence: 80%

Tokens: 132 input + 89 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	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 5ms 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 bug where the report does not include source coverage information, which can make it difficult to understand the code's performance and identify areas for improvement.

Key Assertions:

- The length of the `source_coverage` list in the report should be equal to 1.
- The file path of the first element in the `source_coverage` list should match 'src/foo.py'.
- Each element in the `source_coverage` list should have the following attributes: `file_path`, `statements`, `missed`, `covered`, `coverage_percent`, `covered_ranges`, and `missed_ranges`.

Confidence: 80%

Tokens: 291 input + 166 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/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 5ms 4

AI ASSESSMENT

Scenario: Test ReportWriter::test_write_report_merges_coverage verifies that the test writes a merged coverage report.

Why Needed: This test prevents regression in case of multiple tests with different coverage, where only one test's coverage is being written to the report.

Key Assertions:

- The first test should have a single coverage entry.
- The file path of the first test's coverage entry should match 'src/foo.py'.
- Only one coverage entry should be included in the merged report.

Confidence: 80%

Tokens: 285 input + 116 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/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` 6ms 6

AI ASSESSMENT

Scenario: Test the fallback behavior of atomic write when it fails.**Why Needed:** To prevent a regression where an atomic write attempt fails and does not trigger a direct write.**Key Assertions:**

- The file 'report.json' should exist in the test directory.
- At least one warning message with code 'W203' should be present in the report.

Confidence: 80%**Tokens:** 276 input + 87 output = 363 total

COVERAGE

src/pytest_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::testCreatesDirectoryIfMissing 6ms 6

AI ASSESSMENT

Scenario: Test case: creates directory if missing

Why Needed: Because the test requires a report to be written, which will create a new directory if it doesn't exist.

Key Assertions:

- {'name': 'Directory should be created', 'description': 'The directory should be created at the expected path.'}
- {'name': 'Report JSON file should be present', 'description': 'A report.json file should be present in the expected location.'}

Confidence: 80%

Tokens: 171 input + 115 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/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	81 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

src/pytest_llm_report/report_writer.py

128 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-484, 502, 504, 506-508, 510, 513)

PASSED

tests/test_report_writer.py::TestReportWriterWithFiles::test_ensure_dir_failure 1ms 4

AI ASSESSMENT

Scenario: Test ensures that directory creation fails and captures warnings.

Why Needed: This test prevents a bug where the report writer does not handle directory creation failures correctly, potentially leading to silent errors or incorrect reporting.

Key Assertions:

- The function `writer._ensure_dir(json_path)` raises an exception with code 'W201' when directory creation fails.
- Any warnings raised during directory creation are captured and stored in the `writer.warnings` list.
- The `writer.warnings` list contains at least one warning with code 'W201'.
- The function does not silently ignore errors or return an error code, but instead raises an exception.
- The test verifies that directory creation fails and captures warnings to ensure the report writer behaves as expected.
- The test uses a mock `mkdir` function to simulate a permission denied error, which is raised by the `writer._ensure_dir(json_path)` method.
- The test ensures that the `writer.warnings` list contains at least one warning with code 'W201' for each directory creation attempt.

Confidence: 80%

Tokens: 278 input + 239 output = 517 total

COVERAGE

src/pytest_llm_report/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



AI ASSESSMENT

Scenario: Test 'test_git_info_failure' verifies that the `get_git_info` function handles git command failures gracefully by returning `None` for both SHA and dirty flag values.

Why Needed: This test prevents a potential regression where the `get_git_info` function fails to return expected results when running git commands with errors.

Key Assertions:

- The `get_git_info` function should not raise an exception when it encounters a git command failure.
- The `sha` variable should be set to `None` in this case.
- The `dirty` variable should also be set to `None` in this case.
- When running subprocess.check_output with an exception, the test should still pass without raising an error.
- The function's return values should not be affected by external git command failures.
- The function should handle git commands that fail silently (e.g., due to permissions issues) correctly.
- The `get_git_info` function should not raise a `SubprocessError` exception when it encounters a git command failure.

Confidence: 80%

Tokens: 231 input + 229 output = 460 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	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 40ms 6

AI ASSESSMENT

Scenario: Test verifies that the report writer creates an HTML file and includes expected content.

Why Needed: This test prevents a bug where the report writer does not create an HTML file or does not include expected content in the report.

Key Assertions:

- The report.html file should exist at the specified path.
- The report.html file should contain the expected content (test1, test2, PASSED, FAILED, Skipped, XFailed, XPassed, Errors).
- All nodes with nodeid 'test1' and 'test2' should be found in the HTML content.
- Node 'Skipped' should not be found in the HTML content.
- Node 'XFailed' should not be found in the HTML content.
- Node 'XPassted' should not be found in the HTML content.
- All error messages should be included in the HTML content (AssertionError).

Confidence: 80%

Tokens: 366 input + 200 output = 566 total

COVERAGE

src/pytest_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 39ms 6

AI ASSESSMENT

Scenario: The test verifies that the report includes xfail outcomes in the HTML summary.

Why Needed: This test prevents regression by ensuring that xfail outcomes are included in the HTML summary.

Key Assertions:

- Asserts that 'XFAILED' and 'XFailed' keywords are present in the HTML string.
- Checks if 'XPASSED' and 'XPassed' keywords are also present in the HTML string.
- Verifies that all xfail outcomes have a corresponding xpased outcome in the report.
- Ensures that the report includes both xfailed and xpased outcomes for each test.
- Guarantees that the report does not exclude any tests from being included in the summary.

Confidence: 80%

Tokens: 308 input + 159 output = 467 total

COVERAGE

src/pytest_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 6ms 5

AI ASSESSMENT

Scenario: Test verifies that a JSON file is created with the report.**Why Needed:** Prevents regression where test reports are not written to files.**Key Assertions:**

- The `report.json` file should be created in the specified path.
- The artifact should be tracked for this test.
- The number of artifacts should be at least one.
- The file should exist as expected.
- The report writer should create a new JSON file with each test.

Confidence: 80%**Tokens:** 265 input + 110 output = 375 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 42ms 6

AI ASSESSMENT

Scenario: Test writes PDF file when Playwright is available.**Why Needed:** Prevents regression where the test fails due to missing or corrupted PDF files.**Key Assertions:**

- The `write_pdf` method of the `ReportWriter` class should create a new PDF file with the specified path.
- The `report.pdf` attribute of the `Config` object should be set to the created PDF file path.
- All artifacts in the report should have paths that match the created PDF file path.
- The `write_report` method of the `ReportWriter` class should create a new PDF file with the specified path and add it to the report artifacts.

Confidence: 80%**Tokens:** 478 input + 149 output = 627 total

COVERAGE

src/pytest_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_p
df_missing_playwright.warns` 7ms 4

AI ASSESSMENT

Scenario: Test that a warning is raised when Playwright is missing for PDF output.

Why Needed: To prevent the test from failing and providing incorrect results, this test verifies that it correctly handles the absence of Playwright for PDF output.

Key Assertions:

- The `pdf_path` does not exist after the test is run.
- Any warnings raised are of type WarningCode.W204_PDF_PLAYWRIGHT_MISSING.
- The warning message includes the string 'Playwright is missing'.

Confidence: 80%

Tokens: 311 input + 115 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	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	103 lines (ranges: 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226, 230-231, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 408-412, 415)

 [tests/test_report_writer_coverage.py](#)

10 tests

PASSED

tests/test_report_writer_coverage.py::TestGetGitInfo::test_git_info_from_nonexistent_path 3ms 3

AI ASSESSMENT

Scenario:

tests/test_report_writer_coverage.py::TestGetGitInfo::test_git_info_from_nonexistent_path

Why Needed: To test that the function correctly returns None when given a non-git directory.

Key Assertions:

- {'expected_value': 'None', 'actual_value': 'sha'}
- {'expected_value': 'None', 'actual_value': 'dirty'}

Confidence: 80%

Tokens: 123 input + 98 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)
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 6ms 3

AI ASSESSMENT

Scenario: tests/test_report_writer_coverage.py::TestGetGitInfo::test_git_info_from_valid_repo

Why Needed: To ensure that the test_get_git_info function correctly returns git info from a valid git repository.

Key Assertions:

- {'name': 'assert sha is None or isinstance(sha, str)', 'expected_result': {'type': 'NoneType', 'message': 'Test should return None if not in git repo'}, 'actual_result': {'type': 'str', 'message': 'Expected to get a string from the test_get_git_info function'}}}

Confidence: 80%

Tokens: 159 input + 135 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/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: Test falls back to git runtime when `_git_info` import fails.

Why Needed: Prevents a potential bug where the test fails due to an import error in `_git_info`.

Key Assertions:

- The function `'get_plugin_git_info()'` should return either `None` or a string.
- The function `'get_plugin_git_info()'` should still work via git runtime fallback even when `'_git_info'` is not available.
- The value of `'sha'` in the returned tuple should be `None` if `'_git_info'` import fails, otherwise it should be a string.
- The type of `'sha'` in the returned tuple should be either `None` or `str`.
- If `'_git_info'` import fails, the test should still pass without raising an `AssertionError`.

Confidence: 80%

Tokens: 253 input + 169 output = 422 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	3 lines (ranges: 127-128, 130)

PASSED

tests/test_report_writer_coverage.py::TestGetPluginGitInfo::test_plugin_git_info_returns_values

1ms

3

AI ASSESSMENT

Scenario: tests/test_report_writer_coverage.py::TestGetPluginGitInfo

Why Needed: To ensure the plugin git info function returns some values.

Key Assertions:

- {'name': 'assert sha is None or isinstance(sha, str)', 'expected_result': ['None', 'str'], 'actual_result': [1, 2]}

Confidence: 80%

Tokens: 141 input + 84 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	3 lines (ranges: 127-128, 130)

PASSED

tests/test_report_writer_coverage.py::TestReportWriterAtomicWrite::test_atomic_write_fallback

7ms



AI ASSESSMENT

Scenario: Test atomic write falls back to direct write on error.**Why Needed:** To ensure the report writer can handle cases where an atomic write operation fails and needs to fall back to a non-atomic write method.**Key Assertions:**

- {'assertion_type': 'File existence', 'expected_result': 'report.json exists'}

Confidence: 80%**Tokens:** 181 input + 85 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/models.py	80 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	122 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED

tests/test_report_writer_coverage.py::TestReportWriterPDF::test_pdf_playwright_exception

117ms



6

AI ASSESSMENT

Scenario: Test PDF generation when playwright raises exception.**Why Needed:** Prevents regression where playwright raises an exception and report generation fails without a clear error message.**Key Assertions:**

- The `writer.write_pdf` method should raise a `FileNotFoundError` if the PDF file cannot be written to the specified path.
- The `writer.warnings` list should contain warnings with code 'W201' when an exception is raised during report generation.
- The `report.pdf` attribute of the `writer` object should not be set to a non-existent file path.
- Any exceptions raised by `writer.write_pdf` or `writer.warnings` should be caught and handled correctly.
- The `config.report_pdf` parameter should be set to a valid file path for the report PDF.
- The `report.pdf` attribute of the `ReportWriter` object should not be set to an empty string.
- Any errors that occur during report generation should be logged or reported in some way.

Confidence: 80%**Tokens:** 356 input + 218 output = 574 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	26 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65-67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	28 lines (ranges: 156-158, 408, 417, 419, 421-423, 431-436, 439, 441-442, 455, 460, 462, 465-469, 477-478)

PASSED

tests/test_report_writer_coverage.py::TestReportWriterPDF::test_pdf_playwright_not_installed 1ms 4

AI ASSESSMENT

Scenario: Test PDF generation when playwright is not installed.**Why Needed:** Prevents a potential bug where the report writer fails to create a PDF file without the required playwright installation.**Key Assertions:**

- The 'W204' warning should be raised indicating that playwright is missing.
- The PDF file 'report.pdf' should not exist.
- Any other warnings or errors related to playwright should be suppressed.

Confidence: 80%**Tokens:** 293 input + 97 output = 390 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 57ms 6

AI ASSESSMENT

Scenario: Test _resolve_pdf_html_source creates temp file when no HTML source is provided.**Why Needed:** Prevents a potential bug where the test fails if no HTML source is provided, causing the report writer to create an empty temporary file.**Key Assertions:**

- The path returned by _resolve_pdf_html_source should exist and have a suffix of '.html'.
- The path returned by _resolve_pdf_html_source should not be empty.
- The suffix of the returned path should be '.html'.

Confidence: 80%**Tokens:** 265 input + 118 output = 383 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	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

34ms



AI ASSESSMENT

Scenario: Verify that the test resolves an HTML source when a non-existent HTML file exists.

Why Needed: Prevents a bug where the report writer falls back to using a temporary PDF file instead of the actual HTML source.

Key Assertions:

- The path to the resolved HTML file exists.
- The path to the resolved HTML file is different from the original non-existent HTML file.
- The report writer does not fall back to using a temporary PDF file when an HTML source is missing.
- The test passes even if the actual HTML source is not found but the configuration specifies it should exist.
- The test ensures that the path to the resolved HTML file is correct and unique.
- The test verifies that the report writer does not use the original non-existent HTML file as a fallback.
- The test checks for any potential issues with the temporary PDF file created during the resolution process.

Confidence: 80%

Tokens: 270 input + 196 output = 466 total

COVERAGE

src/pytest_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:** This test prevents a regression where the report writer does not use an existing HTML file.**Key Assertions:**

- The path of the resolved PDF file is set to the existing HTML file.
- The temporary flag is set to False, indicating that the report was generated from a non-temporary source.
- The report writer uses the existing HTML file as its source.
- The configuration string contains the path to the existing HTML file.
- The resolved PDF file has the same name as the original HTML file.
- The resolved PDF file is not temporary, indicating that it was generated from a non-temporary source.

Confidence: 80%**Tokens:** 266 input + 157 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	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 `from_dict` can handle a full dictionary with all required fields.

Why Needed: Prevents regression in case of missing or invalid data.

Key Assertions:

- asserts the correct value for `scenario` field
- asserts the correct value for `why_needed` field
- asserts the correct values for `key_assertions` field
- sets the expected confidence level

Confidence: 80%

Tokens: 276 input + 100 output = 376 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/schemas.py	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: Test that the `to_dict` method correctly converts the schema to a dictionary with all required fields.

Why Needed: This test prevents regression bugs where the `to_dict` method is not correctly converting the schema to a dictionary.

Key Assertions:

- assert data['scenario'] == 'Verify login'
- assert data['why_needed'] == 'Catch auth bugs'
- assert data['key_assertions'] == ['assert 200', 'assert token']
- # Correctly assert all required fields

Confidence: 80%

Tokens: 273 input + 119 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/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 93ms 8

AI ASSESSMENT

Scenario: The test verifies that an HTML report is created when the `--llm-report` option is used.

Why Needed: This test prevents a regression where the HTML report is not generated correctly due to changes in the PyTest configuration.

Key Assertions:

- The file path of the report should exist after running the test with the `--llm-report` option.
- The content of the report should contain the string `''`.
- The string 'test_simple' should be present in the report's content.

Confidence: 80%**Tokens:** 264 input + 122 output = 386 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 131ms 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 some statuses are missing from the HTML summary.

Key Assertions:

- The function `assert_summary(labels: list[str], expected: int)` should be able to correctly identify and report on each status in the list.
- For the 'Passed' label, it should match the expected count of 1.
- For the 'Failed' label, it should also match the expected count of 1.
- For the 'Skipped' label, it should not raise an assertion error but instead continue to the next test.
- For the 'XFailed' and 'XPassed' labels, they should also match their respective expected counts.
- The function `assert_summary(labels: list[str], expected: int)` should be able to handle cases where there are multiple statuses in a single line (e.g., 'Errors', 'Error').
- It should raise an assertion error if the number of labels does not match the expected count.
- For each label, it should check that the count is equal to the expected value and return without raising an assertion error if this condition is met.

Confidence: 80%

Tokens: 621 input + 265 output = 886 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)

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 66ms 7

AI ASSESSMENT

Scenario: The JSON report is created and contains the expected schema version, summary statistics, and test counts.

Why Needed: This test prevents a regression where the report generation process fails to create a valid JSON file with the required metadata.

Key Assertions:

- The `schema_version` key in the report data should be equal to '1.0'.
- The `summary` key in the report data should contain an object with keys 'total', 'passed', and 'failed'.
- The `summary` object's `total` value should be equal to 2.
- The `summary` object's `passed` value should be equal to 1.
- The `summary` object's `failed` value should be equal to 1.
- The `test_pass()` function is called with no assertions.
- The `test_fail()` function is called with a non-zero assertion count.

Confidence: 80%

Tokens: 295 input + 202 output = 497 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 58ms 14

AI ASSESSMENT

Scenario: Verify that LLM annotations are included in the report when a provider is enabled.**Why Needed:** Prevents regressions by ensuring LLM annotations are present in reports.**Key Assertions:**

- The function `test_pass()` returns True.
- The value of `litellm.completion` is set to `mock_completion` before the test runs.
- The `pytest_configure` function imports `litellm` and sets its completion function to `mock_completion`.
- The `pyproject.toml` file includes `[tool.pytest_llm_report]` configuration with `litellm.completion = mock_completion`.
- The `makefile` creates a `pyproject.toml` file with the specified configuration.
- The `pytester.makeconftest` function patches `litellm.completion` before importing it in the test.
- The `mock_completion` function returns a `SimpleNamespace` with the expected annotations.
- The `asserts True` assertion passes when the `test_pass()` function is called.

Confidence: 80%**Tokens:** 385 input + 233 output = 618 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 94ms 14

AI ASSESSMENT

Scenario: Test that LLM errors are surfaced in HTML output.**Why Needed:** This test prevents regression where LLM errors are not reported correctly.**Key Assertions:**

- Verify that the error is reported in the HTML output.
- Check if the error message is displayed correctly.
- Ensure the error is included in the report.
- Verify that the error is not silently ignored.
- Test that the error is surfaced even when it's a litellm.completion error.
- Check for any additional error messages or details.
- Verify that the error is reported with the correct severity level.

Confidence: 80%**Tokens:** 313 input + 138 output = 451 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: The test verifies that the LLM opt-out marker is correctly recorded in the report.

Why Needed: This test prevents a regression where the LLM opt-out marker might not be properly recorded in the report.

Key Assertions:

- The 'llm_opt_out' marker should be present in the test results.
- The 'llm_opt_out' marker should have a value of True for this test.
- There should only be one test result with the 'llm_opt_out' marker.
- The 'llm_opt_out' marker should not be False for any other tests.

Confidence: 80%

Tokens: 290 input + 137 output = 427 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 to ensure it records the correct requirements.

Why Needed: This test prevents a potential bug where the requirement marker is not recorded correctly, causing tests to fail with an incorrect list of requirements.

Key Assertions:

- The `@pytest.mark.requirement` decorator should be applied to functions or modules that require specific requirements.
- The `requirement` parameter in `@pytest.mark.requirement` should be set to a valid string (e.g., 'REQ-001', 'REQ-002')
- The `requirements` key in the test function's metadata should contain a list of strings (e.g., ['REQ-001', 'REQ-002'])
- The `tests` key in the JSON report file should contain exactly one object with a single string value (the requirements)
- Each requirement in the requirements list should be present in the report
- The `reqs` variable should contain a list of strings that includes both 'REQ-001' and 'REQ-002'
- The test function should have been run successfully without any errors

Confidence: 80%

Tokens: 307 input + 239 output = 546 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 62ms 7

AI ASSESSMENT

Scenario: The test verifies that multiple xfailed tests are recorded in the report.**Why Needed:** This test prevents regression by ensuring that all xfailed tests are properly reported and counted.**Key Assertions:**

- The number of xfailed tests is correctly reported as 2.
- All xfailed tests are included in the 'xfailed' key in the report.
- Each xfailed test has a corresponding outcome ('xfailed') in the list of outcomes.
- The test does not fail due to an assertion error or other non-xfailed test.

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

56ms  7

AI ASSESSMENT

Scenario: Test that skipping tests prevents the 'skip' marker from being recorded in the report.

Why Needed: This test ensures that skipping tests is properly handled and reported by pytester, preventing incorrect or misleading results.

Key Assertions:

- The number of skipped tests should be exactly 1 according to the report.
- The 'skip' marker should not be present in the report.
- The report path should contain a JSON file with the correct data.
- The 'summary' key in the report data should have a value of 'skipped'.
- The 'skipped' count in the report data should be 1.
- The test skipped function should not be executed during the run.
- The pytester.runpytest command should produce an error message indicating that skipping tests was prevented.

Confidence: 80%

Tokens: 264 input + 180 output = 444 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

61ms  7

AI ASSESSMENT

Scenario: Verifies that the 'xfailed' counter is incremented when an xfailed test is run.

Why Needed: Prevents regression in the test outcomes report, where the 'xfailed' counter might not be updated correctly if a new test with the same name fails.

Key Assertions:

- The 'summary' key in the JSON report should contain a single integer value equal to 1 for each xfailed test.
- Each xfailed test should have a corresponding entry in the 'summary' JSON report with a 'xfailed' counter value of 1.
- If a new test with the same name fails, the 'xfailed' counter should be incremented correctly and reflected in the report.

Confidence: 80%

Tokens: 264 input + 158 output = 422 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

60ms



7

AI ASSESSMENT

Scenario: Test parameterized tests are recorded separately.

Why Needed: This test prevents regression in parametrized testing, where the same test is run multiple times with different inputs.

Key Assertions:

- The function `test_param` should be called three times with input values [1, 2, 3].
- The total number of tests executed should be 3 (three runs of the test).
- At least one test should pass for each input value.
- No additional tests should be run beyond what is expected from `test_param`.

Confidence: 80%

Tokens: 290 input + 130 output = 420 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

51ms



3

AI ASSESSMENT

Scenario: tests/test_smoke_pytester.py::TestPluginRegistration::test_help_contains_examples**Why Needed:** This test is necessary to ensure that the help text for the CLI tool includes usage examples.**Key Assertions:**

- {'assertion': 'The help text should include at least one example.', 'expected_result': ['Example:'], 'actual_result': ['*Example:*--llm-report*', '*Example:*--model-serve']}

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	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 45ms 3

AI ASSESSMENT

Scenario: test markers registered

Why Needed: Test that LLM markers are correctly registered.

Key Assertions:

- {'name': 'markers are found', 'value': ['llm_opt_out', 'llm_context', 'requirement']}
- {'name': 'expected stdout lines', 'value': ['*llm_opt_out*', '*llm_context*', '*requirement*']}

Confidence: 80%

Tokens: 142 input + 93 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	89 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)

src/pytest_llm_report/plugin.py

240 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 558-559, 562, 566-568)

PASSED

tests/test_smoke_pytester.py::TestPluginRegistration::test_plugin_registered 51ms 3

AI ASSESSMENT

Scenario: tests/test_smoke_pytester.py::TestPluginRegistration::test_plugin_registered

Why Needed: The plugin is not registered, which means it will not be available for use during test execution.

Key Assertions:

- {'name': 'plugin is registered', 'description': 'The plugin should be registered before any tests are run.'}

Confidence: 80%

Tokens: 118 input + 87 output = 205 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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::TestSpecialCharacters::test_special_chars_in_nodeid 94ms 8

AI ASSESSMENT

Scenario: Test that special characters in nodeid are handled correctly by pytester.

Why Needed: This test prevents a potential regression where special characters in node IDs could cause issues with the LLM report generation.

Key Assertions:

- The function `test_special_chars_in_nodeid` should not crash when given special characters in node IDs.
- The HTML file generated by pytester should contain the '
- The test assertion `assert s` should pass for all cases where a special character is present in the node ID.
- The function `pytester.makepyfile()` should create an HTML report that contains the '
- The file path of the generated report should exist and be valid.
- The content of the report should contain the '

Confidence: 80%

Tokens: 288 input + 178 output = 466 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)

PASSED

tests/test_time.py::TestFormatDuration::test_boundary_one_minute

1ms



3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_boundary_one_minute**Why Needed:** To ensure that the `format_duration` function correctly formats a duration of exactly one minute.**Key Assertions:**

- {'expected': '1m 0.0s', 'actual': '1m 0.0s'}

Confidence: 80%**Tokens:** 106 input + 85 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/time.py	6 lines (ranges: 39, 41, 43, 46-48)

PASSED

tests/test_time.py::TestFormatDuration::test_microseconds_format

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_microseconds_format

Why Needed: To ensure that the `format_duration` function correctly formats sub-millisecond durations as microseconds.

Key Assertions:

- {'name': "assert 'μs' in result", 'expected': '500μs'}

Confidence: 80%

Tokens: 121 input + 81 output = 202 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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:

- {'expected': '500.0ms', 'actual': '500.0ms'}

Confidence: 80%

Tokens: 119 input + 78 output = 197 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	3 lines (ranges: 39, 41-42)

PASSED

tests/test_time.py::TestFormatDuration::test_minutes_format

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_minutes_format**Why Needed:** To ensure the `format_duration` function correctly formats durations over a minute, including minutes and seconds.**Key Assertions:**

- {'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:**

- The output of `format_duration(185.0)` is '3m 5.0s'.

Confidence: 80%**Tokens:** 112 input + 75 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/time.py	6 lines (ranges: 39, 41, 43, 46-48)

PASSED

tests/test_time.py::TestFormatDuration::test_one_second

1ms



AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_one_second**Why Needed:** To ensure that the `format_duration` function correctly formats a duration of exactly one second as '1.00s'.**Key Assertions:**

- {'expected_value': '1.00s', 'actual_value': '1.0'}

Confidence: 80%**Tokens:** 101 input + 86 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/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 the function `format_duration` correctly formats seconds under a minute.**Key Assertions:**

- {'description': "The result should contain 's' (seconds) as part of the assertion.", 'value': 's'}
- {'description': "The result should be equal to '5.50s'.", 'value': '5.50s'}

Confidence: 80%**Tokens:** 110 input + 115 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/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 that the `format_duration` function correctly formats small millisecond durations.

Key Assertions:

- {'expected': '1.0ms', 'actual': '1.0ms'}

Confidence: 80%

Tokens: 111 input + 77 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/time.py	3 lines (ranges: 39, 41-42)

PASSED

tests/test_time.py::TestFormatDuration::test_very_small_microseconds 1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_very_small_microseconds

Why Needed: To ensure that the `format_duration` function can correctly format very small durations as microseconds.

Key Assertions:

- {'name': 'result', 'expected': '1μs'}

Confidence: 80%

Tokens: 116 input + 77 output = 193 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	2 lines (ranges: 39-40)

PASSED

tests/test_time.py::TestIsoFormat::test_formats_datetime_with_utc

1ms



AI ASSESSMENT

Scenario: Tests time-related functionality with UTC timezone**Why Needed:** To ensure that datetime objects can be correctly formatted with the UTC timezone.**Key Assertions:**

- {'assertion': "The result of iso_format(dt) is equal to '2024-01-15T10:30:45+00:00'", 'expected_result': '2024-01-15T10:30:45+00:00'}

Confidence: 80%**Tokens:** 143 input + 106 output = 249 total

COVERAGE

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



AI ASSESSMENT

Scenario: Testing naive datetime formats

Why Needed: To ensure that the `iso_format` function correctly handles naive datetime objects without a timezone.

Key Assertions:

- {'name': 'Result format', 'expected': '2024-06-20T14:00:00', 'actual': '2024-06-20T14:00:00', 'message': 'Expected result format, got actual'}

Confidence: 80%**Tokens:** 136 input + 106 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/time.py	1 lines (ranges: 27)

PASSED

tests/test_time.py::TestIsoFormat::test_formats_with_microseconds

1ms



AI ASSESSMENT

Scenario: Tests time module with microseconds

Why Needed: To test the format of datetime objects with microseconds

Key Assertions:

- {'name': 'Result contains 123456', 'expected_value': '123456', 'actual_value': ' returns a string containing the ISO-formatted time in seconds and microseconds, which is then converted to a datetime object.'}

Confidence: 80%

Tokens: 133 input + 98 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/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: This test ensures that the `utc_now()` function returns a datetime object with an associated UTC timezone.

Key Assertions:

- {'name': 'assert result.tzinfo is not None', 'description': 'The returned datetime object has a valid timezone info.'}
- {'name': 'assert result.tzinfo == UTC', 'description': "The returned datetime object's timezone info is set to UTC."}

Confidence: 80%**Tokens:** 109 input + 124 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/util/time.py	1 lines (ranges: 15)

PASSED

tests/test_time.py::TestUtcNow::test_is_current_time

1ms  3

AI ASSESSMENT

Scenario: Test that the function returns a valid UTC now.

Why Needed: The test is necessary to ensure the function can return a current time within a reasonable tolerance.

Key Assertions:

- {'name': 'Before and After Time Tolerance', 'description': 'The result should be within 1 second of the before and after times.'}

Confidence: 80%

Tokens: 116 input + 87 output = 203 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	1 lines (ranges: 15)

PASSED

tests/test_time.py::TestUtcNow::test_returns_datetime

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestUtcNow::test_returns_datetime

Why Needed: This test ensures that the `utc_now` function returns a datetime object.

Key Assertions:

- {'name': 'result is an instance of datetime', 'expected_result': 'datetime'}

Confidence: 80%

Tokens: 94 input + 75 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/time.py	1 lines (ranges: 15)

PASSED

tests/test_token_refresh.py::TestTokenRefresher::test_command_failure

1ms



3

AI ASSESSMENT

Scenario: Test TokenRefresher raises error on command failure when get-token command fails.

Why Needed: This test prevents a potential bug where the TokenRefresher class does not handle command failures properly, potentially leading to unexpected behavior or errors.

Key Assertions:

- The 'run' function of subprocess is called with an incorrect returncode (1) when the get-token command fails.
- The error message returned by the 'run' function contains the string 'Authentication failed'
- The test asserts that the error message includes the string 'exit 1', which indicates a failure in the process.

Confidence: 80%

Tokens: 310 input + 136 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/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 when it encounters an empty output.

Key Assertions:

- the output of the get_token() method is an empty string
- the error message returned by the TokenRefreshError exception includes the phrase 'empty output'
- the error message is in lowercase to ensure correct comparison with the expected string

Confidence: 80%

Tokens: 297 input + 117 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/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: Verify that `TokenRefresher` forces a cache refresh when `force=True` and the refresh interval is set to 3600 seconds.

Why Needed: This test prevents a potential bug where the token refresh does not occur immediately after setting `force=True` due to the cached tokens taking too long to expire.

Key Assertions:

- The function `get_token()` returns the expected token value for both `token1` and `token2` when `force=True`.
- The function `get_token(force=True)` calls the `fake_run` function with the correct arguments, resulting in a completed process object with an `stdout` attribute containing the expected token value.
- The `call_count` variable is incremented correctly to 2 after calling `get_token()` twice with `force=True`.
- The `token1` and `token2` variables are assigned the expected values based on their indices in the list of tokens returned by `get_token()`.

Confidence: 80%

Tokens: 346 input + 214 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/token_refresh.py	25 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh.py::TestTokenRefresher::test_get_token_json _custom_key

1ms



3

AI ASSESSMENT

Scenario: Verify that the `TokenRefresher` class uses a custom JSON key for token refresh.**Why Needed:** This test prevents a potential bug where the default JSON key used by the `TokenRefresher` class is not compatible with the expected custom key.**Key Assertions:**

- The `json_key` parameter passed to the `get_token()` method of the `TokenRefresher` class matches the custom JSON key provided in the test.
- The output of the `subprocess.run()` function is a JSON object with the correct access token.
- The `access_token` field in the JSON response is set to the specified custom key.
- The `token` variable is assigned the expected custom key value.
- The `assert` statement checks that the `token` variable matches the expected custom key value.
- The `subprocess.run()` function returns a CompletedProcess object with a returncode of 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



3

AI ASSESSMENT

Scenario: The `TokenRefresher` class extracts the correct JSON format for the obtained token.

Why Needed: This test prevents a potential bug where the token is not in the expected JSON format.

Key Assertions:

- The output of the `get-token` command is a JSON object with 'token' and 'expires_in' keys.
- The 'token' key contains the value 'json-token-value'.
- The 'expires_in' key contains the value 3600 (one hour).
- The 'command' attribute of the `TokenRefresher` instance matches the command used to run the `get-token` command.
- The `refresh_interval` attribute is set to 3600 seconds.
- The `output_format` attribute is set to 'json'.

Confidence: 80%

Tokens: 308 input + 178 output = 486 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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 `TokenRefresher` class extracts the correct token from the text output when using the 'text' format.

Why Needed: This test prevents a bug where the token is not extracted correctly if the output format is set to 'text'.

Key Assertions:

- token == 'my-secret-token'
- output_format == 'text'
- subprocess.run.returncode == 0
- # expected return code of 0

Confidence: 80%

Tokens: 298 input + 106 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/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: The test verifies that the TokenRefresher raises a TokenRefreshError when given invalid JSON.

Why Needed: This test prevents a bug where the TokenRefresher incorrectly handles invalid JSON input.

Key Assertions:

- The output of the get_token() method should contain 'json' in its string representation.
- The error message should include the word 'json'.
- The error message should not be empty.
- The error message should not contain any other characters besides 'json'.

Confidence: 80%

Tokens: 299 input + 118 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/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 a refresh.

Why Needed: This test prevents potential regression where the TokenRefresher.invalidate() method does not clear the cache after a successful refresh.

Key Assertions:

- The token returned by get_token() should be different from the initial token.
- The call count of the fake_run function should increase by 1 after calling invalidate() on the TokenRefresher instance.
- The output of get_token() should contain the token number in the format 'token-X', where X is the call count.
- The output of invalidate() should be a CompletedProcessResult with returncode=0, stdout='token-Y', and stderr=''
- The value of call_count after calling invalidate() on the TokenRefresher instance should be 1.
- The token returned by get_token() should not be equal to the initial token.
- The output of get_token() should contain the token number in the format 'token-1' or 'token-2'.
- The output of invalidate() should contain the expected error message.

Confidence: 80%

Tokens: 340 input + 243 output = 583 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	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 a JSON key is missing.

Why Needed: To prevent the test from passing and to ensure that the function correctly handles cases where a required JSON key is not present.

Key Assertions:

- The `token` key should be present in the response.
- A message indicating that the token was not found should be included in the error message.
- The error message should include the word 'not found'.
- The function should raise a `TokenRefreshError` exception with a meaningful error message.
- The function should handle cases where the required JSON key is missing without raising an exception or returning an incorrect result.

Confidence: 80%

Tokens: 325 input + 153 output = 478 total

COVERAGE

src/pytest_llm_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: Verify that TokenRefresher is thread-safe by ensuring all threads acquire the lock before getting a token.

Why Needed: This test prevents a potential deadlock or starvation of threads when multiple threads are trying to get a token simultaneously.

Key Assertions:

- all threads should acquire the lock before getting a token (token-1)
- each thread should get the same token (token-0, token-1, ..., token-4)
- no thread should be left without acquiring the lock
- the order of tokens obtained by each thread is consistent (token-1, token-2, ..., token-4)
- all threads should finish getting a token within 5 seconds or less

Confidence: 80%

Tokens: 427 input + 158 output = 585 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 TokenRefresher handles command timeouts correctly by raising a TokenRefreshError when the 'get-token' command times out.

Why Needed: This test prevents a potential bug where the TokenRefresher does not raise an error when the 'get-token' command takes too long to complete, potentially leading to unexpected behavior or errors in downstream code.

Key Assertions:

- The function `fake_run` raises a `TimeoutExpired` exception with the message 'timed out'.
- The test asserts that the error message contains the word 'timed out'.
- The test asserts that the error message is case-insensitive (i.e., it matches 'Timed Out' or 'TIMEOUT').

Confidence: 80%

Tokens: 279 input + 162 output = 441 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	16 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 113-114)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh.py::TestTokenRefresher::test_token_caching

1ms



AI ASSESSMENT

Scenario: Test that TokenRefresher caches tokens and doesn't call command again.

Why Needed: This test prevents a potential issue where the `get-token` command is called multiple times due to caching.

Key Assertions:

- The function `token1` should be equal to `

Confidence: 80%

Tokens: 353 input + 74 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/llm/token_refresh.py	28 lines (ranges: 59-60, 63-66, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

[tests/test_token_refresh_coverage.py](#)

9 tests

PASSED

tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_command_failure_no_stderr

1ms



3

AI ASSESSMENT

Scenario: Test the test_command_failure_no_stderr function to verify that it correctly handles a command failure with no stderr output.

Why Needed: This test prevents a potential bug where the TokenRefresher class does not handle command failures properly, leading to unexpected behavior or errors.

Key Assertions:

- The function `get_token()` in the `TokenRefresher` class should raise a `TokenRefreshError` with an error message indicating that no stderr output was produced.
- The error message should include the string 'exit 1' to indicate that the command failed without producing any stderr output.
- The function should also check for the presence of the string 'No error output' in the error message to ensure it is not silently ignoring this information.

Confidence: 80%

Tokens: 322 input + 165 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/llm/token_refresh.py	20 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101-104, 113, 115)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_empty_command_string

1ms 3

AI ASSESSMENT

Scenario: Test handling of empty command string.

Why Needed: To ensure the TokenRefresher class handles empty command strings correctly and raises a TokenRefreshError with the correct message.

Key Assertions:

- {'assertion': 'The function `get_token()` should raise a `TokenRefreshError` when given an empty command string.', 'expected': 'TokenRefreshError', 'actual': 'Exception'}

Confidence: 80%

Tokens: 151 input + 100 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/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 potential bug where the `TokenRefresher` class incorrectly raises a `TokenRefreshError` for an invalid command string.

Key Assertions:

- The `'get_token()'` method of the `'TokenRefresher'` instance should raise a `'TokenRefreshError'` with a message indicating that the command string is invalid.
- The error message should include the phrase 'Invalid command string'.
- When an invalid command string is passed to the `'get_token()'` method, it should not return any token data.
- The error message should be raised as a `'pytest.raises(TokenRefreshError)'` exception.
- The `'exc_info.value'` attribute of the raised exception should contain a `'TokenRefreshError'` object with the specified error message.
- When an invalid command string is passed to the `'get_token()'` method, it should not raise any other exceptions (e.g. `'ValueError'`, `'TypeError'`).

Confidence: 80%

Tokens: 251 input + 226 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/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 verifying that a TokenRefresher raises an error when the output is not a dictionary.

Why Needed: This test prevents a potential regression where the TokenRefresher incorrectly handles non-dict JSON outputs.

Key Assertions:

- The `get_token` method of the `TokenRefresher` instance should raise a `TokenRefreshError` with an error message indicating that the output is not a dictionary.
- The error message should include the string 'list' to ensure it's not just a generic JSON parsing error.
- The test should verify that the error message includes the expected string, allowing for future changes in JSON formats without breaking the test.
- The `json_key` parameter should be set to `

Confidence: 80%

Tokens: 328 input + 162 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/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 bug where the TokenRefresher incorrectly handles empty or non-string token values.**Key Assertions:**

- The function `get_token()` of the `TokenRefresher` class should raise a `TokenRefreshError` with the message 'empty or not a string' when given an empty or non-string token value.
- The error message returned by `get_token()` should include the string 'empty or not a string' to indicate that the input is invalid.
- The function should return a `TokenRefreshError` exception instead of raising it internally, allowing for proper error handling and logging.
- The test should verify that the error message includes the specific phrase 'empty or not a string', which indicates the type of input being validated.
- The test should also verify that the error is not raised when an empty string is passed as a token value in JSON format, but instead returns an empty or non-string result.
- The `json.dumps()` function used to create the expected output should produce a string with only whitespace characters, indicating an empty or invalid input.

Confidence: 80%**Tokens:** 324 input + 251 output = 575 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 that the test_json_token_not_string scenario verifies when token value is not a string, preventing TokenRefreshError.

Why Needed: This test prevents the TokenRefreshError by ensuring that the token value is always a string before attempting to refresh it.

Key Assertions:

- The `json.dumps` function is called with an integer instead of a string.
- An error message indicating 'empty or not a string' is raised when trying to get the token.
- The `TokenRefreshError` exception is raised with a meaningful error message.

Confidence: 80%

Tokens: 326 input + 127 output = 453 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	30 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101, 107-108, 111, 113, 115, 132-135, 139, 143-146, 149)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_oserror_on_execution

1ms



3

AI ASSESSMENT

Scenario: Test verifies that a TokenRefresher raises a TokenRefreshError when executing a command that does not exist.

Why Needed: This test prevents a potential bug where the TokenRefresher incorrectly handles commands that are not found during execution.

Key Assertions:

- The 'get_token' method of the refresher should raise a TokenRefreshError with a message indicating that the command was not found.
- The error message should include 'Failed to execute'.
- The error message should be raised as an exception, not as a string.
- The error message should contain the text 'Command not found'.
- The error message should not be empty.

Confidence: 80%

Tokens: 280 input + 149 output = 429 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	19 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 113, 115-118)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_text_only_whitespace_lines

1ms



3

AI ASSESSMENT

Scenario: Test handling when text output has only whitespace lines after initial strip.

Why Needed: Prevents TokenRefreshError due to incorrect parsing of output with only blank lines after the initial strip.

Key Assertions:

- The `_parse_output` method should return a `TokenRefreshError` if no non-empty lines are found in the output.
- The output wrapper should contain whitespace content lines but not any non-whitespace lines.
- The `parse_output` function should raise a `TokenRefreshError` when given text with only blank lines after the initial strip.
- The test should fail when the `_parse_output` method is called directly with text that has only whitespace lines.
- The test should pass if the output wrapper contains non-empty lines and only whitespace content lines.

Confidence: 80%

Tokens: 376 input + 166 output = 542 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	4 lines (ranges: 132, 153-155)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_whitespace_only_command

1ms

3

AI ASSESSMENT

Scenario: Test the test_whitespace_only_command to verify it handles whitespace-only command strings correctly.

Why Needed: This test prevents a potential bug where TokenRefresher incorrectly handles empty commands and raises an exception instead of returning a meaningful error message.

Key Assertions:

- The function `get_token()` should raise a `TokenRefreshError` with the message 'empty' when given an empty command string.
- The function `get_token()` should return None or handle the case where the command is not empty correctly.
- The error message returned by `get_token()` should be 'empty'.
- The test should fail if `get_token()` returns a non-empty value instead of raising an exception.
- The test should pass if `get_token()` raises a `TokenRefreshError` with the correct message.
- The function `get_token()` should handle whitespace-only command strings without any issues or exceptions.
- The function `get_token()` should not raise any other types of exceptions for empty commands.

Confidence: 80%

Tokens: 236 input + 219 output = 455 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: Verify token usage aggregation for multiple test cases**Why Needed:** Prevent regression in token usage reporting when aggregating results from multiple tests**Key Assertions:**

- The total input tokens should be 30 (10 + 20)
- The total output tokens should be 15 (5 + 10)
- The number of annotations should be 2 (LLM annotation for test1 and test3)
- The total tokens should be 45 (15 + 30)

Confidence: 80%**Tokens:** 775 input + 115 output = 890 total

COVERAGE

src/pytest_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)
---------------------------------	---