

Test Report

Run ID: 21201087766-py3.12 • Generated: 2026-01-21 07:36:48 • Duration: 119.30s

Plugin: v0.2.1 (a03dbe622cdc018f89b74731aed91adf1a582867) [dirty]

Repo: v0.2.0 (b1f3c49618f01bb762e8fa4ff37082e2ac4ba601)

LLM: ollama / llama3.2:1b (minimal context, 621 annotated, 1 errors)

Token Usage: 135999 input, 73657 output (Total: 209656)

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 il/ranges.py	33	0	33	100.0%	12, 15, 29-30, 33, 35-37, 39-40, 42, 45-47, 50, 52, 55, 65-67, 70, 81-82, 84-91, 93, 95	-
------------------------------------------	----	---	----	--------	--------------------------------------------------------------------------------------------------------	---

src/pytest_llm_report/ut il/time.py	16	0	16	100.0%	4, 6, 9, 15, 18, 27, 30, 39-44, 46-48	-
----------------------------------------	----	---	----	--------	---------------------------------------------	---

Per Test Details

[tests/test_adaptive_prompts.py](#) 9 tests

PASSED	tests/test_adaptive_prompts.py::TestComplexityEstimation::test_compl ex_test_high_complexity	1ms	5
--------	-------------------------------------------------------------------------------------------------	-----	---

AI ASSESSMENT

Scenario: ...

Why Needed: ...

Key Assertions:

- ...

Confidence: 80%

Tokens: 118 input + 25 output = 143 total

COVERAGE

src/pytest_llm_report/collector.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:

- {'name': 'assert provider._estimate_test_complexity returns 0 for an empty string', 'expected': 0, 'got': 0}
- {'name': 'assert provider._estimate_test_complexity returns 0 for None', 'expected': 0, 'got': 0}

Confidence: 80%

Tokens: 136 input + 139 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/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_low_complexity 2ms 5

AI ASSESSMENT

Scenario:

tests/test_adaptive_prompts.py::TestComplexityEstimation::test_simple_low_complexity

Why Needed: The test is necessary to ensure that the complexity estimation algorithm can accurately handle simple tests with low complexity.

Key Assertions:

- {'name': 'complexity score', 'value': 'low'}

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/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: tests/test_adaptive_prompts.py::TestConfigValidation::test_invalid_prompt_tier

Why Needed: To ensure that the prompt tier is valid and raises an error when it's invalid.

Key Assertions:

- {'message': "The prompt tier should be one of [None, 'standard', 'advanced']", 'expected_value': 'invalid'}

Confidence: 80%

Tokens: 126 input + 92 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	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 should pass validation

Why Needed: To ensure that the `prompt_tier` field in the configuration is correctly validated and does not cause any errors.

Key Assertions:

- {'name': 'Expected error count for minimal tier', 'expected_value': 0, 'actual_value': 1}
- {'name': 'Expected error count for standard tier', 'expected_value': 0, 'actual_value': 1}
- {'name': 'Expected error count for auto tier', 'expected_value': 0, 'actual_value': 1}

Confidence: 80%

Tokens: 142 input + 149 output = 291 total

COVERAGE

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

Why Needed: ...

Key Assertions:

- ...

Confidence: 80%

Tokens: 122 input + 25 output = 147 total

COVERAGE

src/pytest_llm_report/collector.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-tier feature can use minimal prompts for simple tests, which reduces the overhead of prompt generation and improves test performance.

Key Assertions:

- {'name': 'selected_prompt_type', 'expected_value': 'MINIMAL_SYSTEM_PROMPT'}
- {'name': 'minimal_system_prompt', 'expected_value': {'prompt_type': 'minimal', 'prompt_text': 'def test_simple(): assert 1 == 1'}}}

Confidence: 80%**Tokens:** 155 input + 136 output = 291 total

COVERAGE

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

PASSED

tests/test_adaptive_prompts.py::TestPromptTierSelection::test_minimal_tier_override

1ms



AI ASSESSMENT

Scenario:

tests/test_adaptive_prompts.py::TestPromptTierSelection::test_minimal_tier_override

Why Needed: To ensure that the minimal prompt is always used when possible, even in cases where a more advanced tier is required.**Key Assertions:**

- {'name': 'config', 'value': 'Config override to minimal should always use minimal prompt.'}

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/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: Because the config override is not necessary in this case, as the standard prompt is already used.

Key Assertions:

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

Confidence: 80%

Tokens: 148 input + 77 output = 225 total

COVERAGE

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

tests/test_aggregation.py

10 tests

AI ASSESSMENT

Scenario: Verify that the aggregate function correctly handles all policy for multiple test cases.

Why Needed: This test prevents regression when using 'all' aggregation policy, which may cause duplicate tests to be included in the report.

Key Assertions:

- The aggregated report should contain both retained tests.
- The number of tests in the aggregated report should match the expected number (2).
- Each retained test should have a unique ID.
- No duplicate tests should be included in the aggregated report.
- All test cases should be included in the aggregated report, even if they are not retained.
- The aggregate function should correctly handle all policy for multiple test cases.
- The aggregate function should not include any redundant or duplicate data.

Confidence: 80%

Tokens: 364 input + 164 output = 528 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

3ms



AI ASSESSMENT

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

Why Needed: To test that the aggregate function correctly handles a directory that does not exist.

Key Assertions:

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

Confidence: 80%

Tokens: 104 input + 77 output = 181 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

3ms



3

AI ASSESSMENT

Scenario: Test that the latest policy is selected when aggregating reports with different times.**Why Needed:** This test prevents a regression where the latest policy might not be chosen for aggregated reports with different times.**Key Assertions:**

- The outcome of the aggregate should match the outcome of the report with the latest timestamp.
- There should only be one test in the result.
- The outcome of the test should be 'passed'.
- The run meta should indicate that the aggregation was performed on multiple runs.
- The number of passed tests should equal the total number of runs.
- The summary should contain exactly one passed test.

Confidence: 80%**Tokens:** 477 input + 144 output = 621 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

Why Needed: The test ensures that an aggregator can be created without a specified directory configuration.

Key Assertions:

- {'name': 'agg is None after aggregate call', 'expected': 'None'}

Confidence: 80%

Tokens: 110 input + 67 output = 177 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 no reports exist and there are no files to aggregate.

Why Needed: Prevents a potential bug where the function `aggregate` throws an error or raises an exception due to missing reports or empty file paths.

Key Assertions:

- The `aggregate()` method should return `None` when called with an aggregator that has no reports and no files to aggregate.
- The `aggregate()` method should not throw any errors or raise exceptions in this scenario.
- The `aggregate()` method should behave as expected without throwing any errors or raising exceptions due to missing reports or empty file paths.

Confidence: 80%

Tokens: 201 input + 144 output = 345 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

Key Assertions:

- coverage: Verify coverage was properly deserialized from JSON.
- llm_annotation: Verify LLM annotation was properly deserialized from JSON.
- token_usage: Verify token usage serialization is correct (this was the CI bug fix).

Confidence: 80%

Tokens: 1002 input + 98 output = 1100 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: Source coverage summary should be deserialized.

Why Needed: This test prevents a bug where the source coverage is not correctly deserialized from the aggregated report.

Key Assertions:

- The 'source_coverage' key in the aggregated report contains a list of SourceCoverageEntry objects.
- Each SourceCoverageEntry object has the required attributes: 'file_path', 'statements', 'missed', 'covered', 'coverage_percent', 'covered_ranges', and 'missed_ranges'.
- The 'file_path' attribute is set to the correct file path in the report.
- The 'statements', 'missed', 'covered', 'coverage_percent', 'covered_ranges', and 'missed_ranges' attributes are correctly populated with values from the report.
- The 'coverage_percent' value is a valid percentage between 0 and 100.
- The 'covered_ranges' attribute contains ranges in the correct format (e.g., '1-5, 7-11').
- The 'missed_ranges' attribute also contains ranges in the correct format (e.g., '6, 12').

Confidence: 80%

Tokens: 395 input + 243 output = 638 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: This test prevents a bug where the aggregator fails to load coverage data when the `llm_coverage_source` option is not provided.

Key Assertions:

- The `'_load_coverage_from_source()'` method should return `None` when `llm_coverage_source` is `None`.
- The `'_load_coverage_from_source()'` method should raise a `UserWarning` when trying to load coverage data from an invalid source file.
- The `'_load_coverage_from_source()'` method should mock the `coverage.Coverage` class and its methods to return the expected values.
- The `'_load_coverage_from_source()'` method should verify that it calls the correct functions with the expected arguments.
- The `'_load_coverage_from_source()'` method should verify that the coverage data is correctly loaded into the aggregator's internal state.
- The `'_load_coverage_from_source()'` method should return a valid `SourceCoverageEntry` object when successful loading occurs.
- The `SourceCoverageEntry` class should be mocked to return the correct values for its attributes.

Confidence: 80%

Tokens: 584 input + 242 output = 826 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 function correctly updates the latest summary with test results.

Why Needed: This test prevents regression where the recalculate_summary function fails to update the latest summary even after adding new tests or removing skipped tests.

Key Assertions:

- The total count of passed, failed, and skipped tests should match the actual counts in the latest summary.
- The number of xfailed and xpassed tests should be equal to their actual counts in the latest summary.
- The error status should remain unchanged even after adding new tests or removing skipped tests.
- The coverage percentage should be preserved from the latest summary.
- The total duration of all tests should increase by 5 seconds ($1.0 + 4.0 = 5.0$) to reflect the updated count in the latest summary.

Confidence: 80%

Tokens: 473 input + 181 output = 654 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)

AI ASSESSMENT

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

Why Needed: This test verifies that the `aggregate` function correctly handles cases where the input is not a valid JSON report.

Key Assertions:

- The `aggregate` function should skip the 'invalid.json' file and only count the 'valid.json' file in the aggregation result.
- The `aggregate` function should raise a warning when it encounters an invalid JSON file, indicating that it's being skipped.
- The `aggregate` function should not include any reports from files with missing fields (e.g., 'missing_fields.json') in its final count of valid reports.
- The test should be able to reproduce the issue by creating a temporary directory and writing both an invalid JSON file and a valid JSON file inside it.
- The `aggregate` function should correctly handle cases where the input is not a valid JSON report, without raising any errors or unexpected behavior.

Confidence: 80%

Tokens: 352 input + 212 output = 564 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)

PASSED

tests/test_aggregation_maximal.py::TestAggregationMaximal::test_recalculate_summary_coverage

1ms



4

AI ASSESSMENT

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

Why Needed: This test prevents regression in coverage calculation when new tests with different durations are added to the aggregation process.

Key Assertions:

- The total duration of all passed tests should be equal to the latest summary's total duration.
- The number of passed tests should match the latest summary's passed count.
- The total duration of failed tests should remain unchanged.
- The coverage total percent should still reflect the latest summary's coverage.
- The total duration of all tests should increase by 1 second (3.0 seconds) compared to the previous test result.

Confidence: 80%

Tokens: 299 input + 157 output = 456 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 generation and verification.

Key Assertions:

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

Confidence: 80%

Tokens: 112 input + 116 output = 228 total

COVERAGE

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

PASSED

tests/test_annotator.py::TestAnnotateTests::test_cached_progress_reporting

1ms



6

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_cached_progress_reporting**Why Needed:** To ensure that the progress reporting is cached correctly and not lost in case of a provider or assembler failure.**Key Assertions:**

- {'name': 'mock_cache.get_cached_progress_reporting_result().should_return_json', 'description': 'Mocked cache should return the expected JSON result when the annotator has completed the task.'}
- {'name': 'mock_provider.get_progress_reporting_status().should_return_success', 'description': 'Mocked provider should return a successful status when the annotator completes the task.'}

Confidence: 80%**Tokens:** 101 input + 147 output = 248 total

COVERAGE

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

PASSED

tests/test_annotator.py::TestAnnotateTests::test_cached_tests_are_skipped

2ms



6

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_cached_tests_are_skipped**Why Needed:** To ensure that cached tests are skipped correctly and not executed unnecessarily.**Key Assertions:**

- {'name': 'mock_provider', 'expected_value': 'Mocked provider is used', 'actual_value': 'Mocked provider is used'}
- {'name': 'mock_cache', 'expected_value': 'Mocked cache is used', 'actual_value': 'Mocked cache is used'}
- {'name': 'mock_assembler', 'expected_value': 'Mocked assembler is used', 'actual_value': 'Mocked assembler is used'}

Confidence: 80%**Tokens:** 102 input + 165 output = 267 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**Why Needed:** To ensure that annotators can be annotated concurrently without causing any issues.**Key Assertions:**

- {'name': 'Mock provider should not raise an exception when called concurrently', 'expected_result': 'None'}
- {'name': 'Mock cache should not raise an exception when accessed concurrently', 'expected_result': 'None'}
- {'name': 'Mock assembler should not raise an exception when called concurrently', 'expected_result': 'None'}

Confidence: 80%**Tokens:** 98 input + 128 output = 226 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: To test that concurrent annotation handles failures correctly.

Key Assertions:

- {'assertion_type': 'pytest-capsys', 'expected_output': 'Capture the output of the annotation process and verify it contains an error message.', 'actual_output': 'Capture the output of the annotation process and verify it contains an error message.'}

Confidence: 80%

Tokens: 116 input + 111 output = 227 total

COVERAGE

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

Key Assertions:

- {'name': 'Mock provider should be called with a valid task ID', 'expected_result': 1, 'actual_result': 0}
- {'name': 'Mock cache should not raise an exception when getting a valid task ID', 'expected_result': 1, 'actual_result': 0}

Confidence: 80%

Tokens: 96 input + 129 output = 225 total

COVERAGE

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

1ms



6

AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_reports_progress_messages**Why Needed:** To ensure that the annotator correctly displays progress messages during reporting.**Key Assertions:**

- {'name': 'Mock provider is called with correct arguments', 'expected_calls': [], 'asserted_calls': [], 'message': 'Expected mock provider to be called with correct arguments, but got unexpected ones.'}

Confidence: 80%**Tokens:** 101 input + 106 output = 207 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:** This test ensures that the annotator respects the opt-out and limit settings when performing annotations.**Key Assertions:**

- {'assertion_type': 'mocking', 'expected_mocking': ['mock_provider', 'mock_cache', 'mock_assembler']}

Confidence: 80%**Tokens:** 104 input + 95 output = 199 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:** Respects rate limit for annotating test data.**Key Assertions:**

- {'name': 'mock_provider.get_rate_limit() returns a valid rate limit value', 'expected_value': 10, 'actual_value': 5}
- {'name': 'mock_cache.get_result() does not raise an exception when the result is already cached', 'expected_exception': 'Cache miss', 'actual_exception': 'None'}

Confidence: 80%**Tokens:** 112 input + 133 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, 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



AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_sequential_annotation**Why Needed:** To ensure that sequential annotation works correctly and produces the expected output.**Key Assertions:**

- {'name': 'expected_output', 'description': 'The expected output of the sequential annotation process.'}
- {'name': 'output_length', 'description': 'The length of the output string after sequential annotation.'}

Confidence: 80%**Tokens:** 98 input + 107 output = 205 total

COVERAGE

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

PASSED

tests/test_annotator.py::TestAnnotateTests::test_sequential_annotation_error_tracking

24.00s



6

AI ASSESSMENT

Scenario:

tests/test_annotator.py::TestAnnotateTests::test_sequential_annotation_error_tracking

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

- {'name': 'Mocking the provider', 'expected_output': 'Mocked provider object'}
- {'name': 'Mocking the cache', 'expected_output': 'Mocked cache object'}
- {'name': 'Mocking the assembler', 'expected_output': 'Mocked assembler object'}

Confidence: 80%**Tokens:** 105 input + 135 output = 240 total

COVERAGE

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

PASSED

tests/test_annotator.py::TestAnnotateTests::test_skips_if_disabled

1ms



AI ASSESSMENT

Scenario: tests/test_annotator.py::TestAnnotateTests::test_skips_if_disabled**Why Needed:** The test should be skipped when the LLM (Language Model) is not enabled.**Key Assertions:**

- {'expected_type': 'bool', 'actual_type': 'str'}

Confidence: 80%**Tokens:** 108 input + 79 output = 187 total

COVERAGE

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

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/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 handles malformed JSON correctly and raises a meaningful error.**Key Assertions:**

- {'assertion_type': 'JSONDecodeError', 'expected_error_message': 'Failed to parse LLM response as JSON'}

Confidence: 80%**Tokens:** 152 input + 85 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/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 handles non-string fields correctly, specifically verifying if it correctly identifies the expected key assertion.

Why Needed: This test prevents a potential bug where the function incorrectly assumes all fields are strings and fails to identify the correct list of keys.

Key Assertions:

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

Confidence: 80%**Tokens:** 269 input + 117 output = 386 total

COVERAGE

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

 tests/test_base_maximal.py

9 tests

PASSED

tests/test_base_maximal.py::TestGetProvider::test_get_gemini_provider

1ms



5

AI ASSESSMENT

Scenario: tests/test_base_maximal.py::TestGetProvider::test_get_gemini_provider**Why Needed:** To ensure that the `get_gemini_provider` function returns an instance of `GeminiProvider` when a Gemini provider is requested.**Key Assertions:**

- {'name': 'provider_type', 'expected_value': 'GeminiProvider'}
- {'name': 'provider_class', 'expected_value': 'GeminiProvider'}

Confidence: 80%**Tokens:** 104 input + 111 output = 215 total

COVERAGE

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

PASSED

tests/test_base_maximal.py::TestGetProvider::test_get_invalid_provider

2ms



AI ASSESSMENT

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

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

Key Assertions:

- {'name': 'Expected exception message', 'value': 'Unknown LLM provider: invalid'}

Confidence: 80%

Tokens: 106 input + 80 output = 186 total

COVERAGE

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

PASSED

tests/test_base_maximal.py::TestGetProvider::test_get_litellm_provider

1ms

5

AI ASSESSMENT

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

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

Key Assertions:

- {'name': 'provider_type', 'expected': 'LiteLLMProvider'}

Confidence: 80%

Tokens: 109 input + 86 output = 195 total

COVERAGE

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

AI ASSESSMENT

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

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

Key Assertions:

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

Confidence: 80%

Tokens: 104 input + 80 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	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 Ollama provider can be retrieved and used correctly.

Key Assertions:

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

Confidence: 80%

Tokens: 108 input + 85 output = 193 total

COVERAGE

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

PASSED

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

AI ASSESSMENT

Scenario: Test that the `is_available()` method returns a boolean indicating availability.**Why Needed:** This test prevents a potential regression where the `is_available()` method does not return a boolean value when it should.**Key Assertions:**

- The `is_available()` method of the provider is called and its result is checked to be 'True'.
- The `is_available()` method of the provider is called again with the same arguments and its result is checked to be 'True'.
- The value of `provider.checks` after calling `_check_availability()` is checked to be 1.

Confidence: 80%**Tokens:** 280 input + 137 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	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 model name defaults to the configuration if no custom value is provided.

Key Assertions:

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

Confidence: 80%

Tokens: 114 input + 81 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, 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 LLM provider defaults to using rate limits when they are not explicitly set.

Key Assertions:

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

Confidence: 80%

Tokens: 108 input + 75 output = 183 total

COVERAGE

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

PASSED

tests/test_base_maximal.py::TestLlmProviderDefaults::test_is_local_defaults_to_false

1ms

4

AI ASSESSMENT

Scenario: tests/test_base_maximal.py

Why Needed: To ensure that the LLM default settings are correctly set to false when using a non-local configuration.

Key Assertions:

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

Confidence: 80%

Tokens: 105 input + 74 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/llm/base.py	3 lines (ranges: 65-66, 174)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

tests/test_batching.py

17 tests

PASSED

tests/test_batching.py::TestBuildBatchPrompt::test_context_files_included

1ms



4

AI ASSESSMENT

Scenario: Verify that context files are included in the batch prompt for a given function.**Why Needed:** This test prevents regression where context files are not added to the prompt, potentially causing unexpected behavior or errors.**Key Assertions:**

- The 'src/module.py' file is present in the prompt.
- The 'def helper()' function is included in the prompt.
- Context files should be added to the prompt for a given function.

Confidence: 80%**Tokens:** 261 input + 103 output = 364 total

COVERAGE

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

PASSED

tests/test_batching.py::TestBuildBatchPrompt::test_parametrized_batch_prompt

1ms



3

AI ASSESSMENT

Scenario: Test the parametrized batch prompt functionality.**Why Needed:** This test prevents regression by ensuring that all variants of a test are included in a parametrized batch prompt.**Key Assertions:**

- The 'Test Group: test.py::test_add[*]' assertion should be present in the prompt.
- The 'Parameterizations (2 variants)' assertion should be present in the prompt.
- The '[1+1=2]' assertion should be present in the prompt.
- The '[0+0=0]' assertion should be present in the prompt.
- The 'ONE annotation' assertion should be present in the prompt.

Confidence: 80%**Tokens:** 330 input + 145 output = 475 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/batching.py	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 single_test_prompt test verifies that a normal batched request can be generated.

Why Needed: This test prevents a potential bug where the test prompt is not correctly formatted or does not include necessary information.

Key Assertions:

- Test: test.py::test_foo should appear in the prompt.
- ```python should appear in the prompt.
- source should be included in the prompt.
- Parameterizations should not appear in the prompt.

Confidence: 80%

Tokens: 269 input + 107 output = 376 total

COVERAGE

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

AI ASSESSMENT

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

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

Key Assertions:

- The `source` variable is assigned a string containing test function code.
- Two calls to `_compute_source_hash(source)` return the same hash value.
- The length of the returned hash value is 32 bytes (as expected).

Confidence: 80%

Tokens: 220 input + 110 output = 330 total

COVERAGE

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

PASSED

tests/test_batching.py::TestComputeSourceHash::test_different_source_different_hash

1ms

3

AI ASSESSMENT

Scenario:

tests/test_batching.py::TestComputeSourceHash::test_different_source_different_hash

Why Needed: To ensure that different sources produce different hashes, which is a requirement for batching to work correctly.

Key Assertions:

- {'name': 'hash1 != hash2', 'description': 'The first computed source hash should be different from the second one'}

Confidence: 80%

Tokens: 127 input + 92 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/llm/batching.py	2 lines (ranges: 67, 70)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_batching.py::TestComputeSourceHash::test_empty_source

1ms



AI ASSESSMENT

Scenario: tests/test_batching.py::TestComputeSourceHash::test_empty_source**Why Needed:** The current implementation of compute_source_hash() does not handle an empty source correctly.**Key Assertions:**

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

Confidence: 80%**Tokens:** 94 input + 90 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/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



AI ASSESSMENT

Scenario: tests/test_batching.py::TestConfigValidation::test_batch_max_tests_minimum**Why Needed:** This test is needed because the `batch_max_tests` configuration option must be at least 1 to ensure that the batch size can be set correctly.**Key Assertions:**

- {'name': 'config.validate() returns an error', 'description': 'The `validate()` method of the config object should return a list of errors.'}
- {'name': "any('batch_max_tests' in e for e in errors)", 'description': "At least one 'batch_max_tests' key-value pair should be present in the error messages."}

Confidence: 80%**Tokens:** 126 input + 152 output = 278 total

COVERAGE

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

- {'assertion_type': 'contains', 'expected_value': 'context_line_padding', 'actual_value': -1}

Confidence: 80%

Tokens: 126 input + 79 output = 205 total

COVERAGE

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

PASSED

tests/test_batching.py::TestConfigValidation::test_invalid_context_compression

1ms



AI ASSESSMENT

Scenario: TestConfigValidation test_invalid_context_compression**Why Needed:** To ensure that the `context_compression` parameter is validated correctly and raises an error when it's invalid.**Key Assertions:**

- {'name': 'config validation errors', 'description': 'The `validate()` method should return a list of error messages for the given configuration.'}
- {'name': 'invalid context compression mode', 'description': 'The `context_compression` parameter should be an invalid value and raise an error.'}

Confidence: 80%**Tokens:** 122 input + 125 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	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: To ensure that valid context compression modes pass validation.

Key Assertions:

- {'description': "Context compression mode should be None or 'lines'", 'expected_value': "None|'lines'"}

Confidence: 80%

Tokens: 133 input + 67 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	26 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_batching.py::TestGetBaseNodeid::test_nested_params

1ms  3

AI ASSESSMENT

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

- {'name': 'striped base node id', 'expected': 'test.py::test', 'actual': '_get_base_nodeid()}'}

Confidence: 80%**Tokens:** 109 input + 90 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/batching.py	2 lines (ranges: 53-54)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_batching.py::TestGetBaseNodeid::test_parametrized_nodeid

1ms  3

AI ASSESSMENT

Scenario: tests/test_batching.py::TestGetBaseNodeid::test_parametrized_nodeid**Why Needed:** To ensure that the `parametrized_nodeid` function correctly strips parameters from node IDs.**Key Assertions:**

- {'name': 'node_id assertion', 'expected_value': 'tests/test_foo.py::test_add', 'actual_value': '_get_base_nodeid()}'}

Confidence: 80%**Tokens:** 133 input + 101 output = 234 total

COVERAGE

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

PASSED

tests/test_batching.py::TestGetBaseNodeid::test_simple_nodeid

1ms



AI ASSESSMENT

Scenario: tests/test_batching.py::TestGetBaseNodeid::test_simple_nodeid**Why Needed:** This test is needed because the `'_get_base_nodeid` function does not handle nodeids without parameters correctly.**Key Assertions:**

- {'name': 'nodeid is unchanged', 'expected_value': 'tests/test_foo.py::test_bar', 'actual_value': '_get_base_nodeid(''')}

Confidence: 80%**Tokens:** 123 input + 103 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/batching.py	2 lines (ranges: 53, 55)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_batching.py::TestGroupTestsForBatching::test_batch_max_size_respected

1ms



4

AI ASSESSMENT

Scenario: Large groups of tests should be split into batches with a maximum size.

Why Needed: This test prevents regression when the batch size is set to a value that would result in too many tests being processed at once.

Key Assertions:

- The length of each batch should match the expected number of tests (2, 2, 1).
- Each batch should contain exactly two tests.
- A batch with only one test is not considered a valid batch.
- The total number of tests across all batches should be equal to the specified maximum size (3).

Confidence: 80%

Tokens: 364 input + 133 output = 497 total

COVERAGE

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



AI ASSESSMENT

Scenario: Test Group Tests For Batching

Why Needed: This test is needed because the batching parameterized tests are disabled.

Key Assertions:

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

Confidence: 80%

Tokens: 170 input + 79 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/batching.py	6 lines (ranges: 92-93, 95, 97-99)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

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

1ms



4

AI ASSESSMENT

Scenario: Test parametrized tests should be grouped together.

Why Needed: This test prevents regression due to the lack of grouping of parametrized tests, which can lead to unexpected behavior when running tests in parallel.

Key Assertions:

- The number of batches is equal to 1.
- Each batch contains exactly 3 tests.
- Each batch is a parameterized test.
- 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 + 124 output = 470 total

COVERAGE

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

PASSED

tests/test_batching.py::TestGroupTestsForBatching::test_single_tests_no_grouping

1ms



4

AI ASSESSMENT

Scenario: Test 'test_single_tests_no_grouping' verifies that single tests are batched individually without grouping.

Why Needed: This test prevents regression where multiple tests are grouped together and their individual tests are not batched separately.

Key Assertions:

- Each test is assigned to a separate batch with no grouping.
- There are two batches in total, each containing one test.
- The first batch contains only one test ('test_a') and the second batch also contains only one test ('test_b').

Confidence: 80%

Tokens: 278 input + 118 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/llm/batching.py	18 lines (ranges: 53, 55, 92-93, 95, 103-106, 108-110, 112-116, 136)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

tests/test_cache.py

7 tests

PASSED

tests/test_cache.py::TestHashSource::test_consistent_hash

1ms  3

AI ASSESSMENT

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

Why Needed: To ensure that the cache is consistent across multiple tests.

Key Assertions:

- {'name': 'same source produces same hash', 'expected': 'hash_source(source)', 'actual': 'hash_source(source)'}

Confidence: 80%

Tokens: 107 input + 81 output = 188 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 cache is working correctly when using different source code.

Key Assertions:

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

Confidence: 80%

Tokens: 108 input + 80 output = 188 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 length is correct and consistent across different inputs.**Key Assertions:**

- {'expected_value': 16, 'actual_value': {'hash': '486d8f5e4b2ae6c99a9c43d3a1f29f44'}}}

Confidence: 80%**Tokens:** 100 input + 96 output = 196 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 the `clear` method of LlmCache to ensure it removes all existing entries.

Why Needed: This test prevents a potential bug where the cache might not be cleared properly, leading to unexpected behavior or incorrect results in subsequent tests.

Key Assertions:

- Verify that the `clear` method correctly removes all cache entries.
- Ensure that after clearing the cache, no existing entries are still present.
- Test that the `get` method returns `None` for any non-existent keys.
- Verify that the cache size is updated correctly before and after clearing.
- Check if the test raises an error when trying to access a non-existent key.

Confidence: 80%

Tokens: 283 input + 150 output = 433 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**Why Needed:** To ensure that LLMCache does not cache errors in annotations.**Key Assertions:**

- {'name': 'cache should be empty when annotation has error', 'expected_value': [], 'actual_value': ['abc123']}

Confidence: 80%**Tokens:** 157 input + 74 output = 231 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



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:**

- {'expected_value': 'None', 'actual_value': 'result'}

Confidence: 80%**Tokens:** 128 input + 72 output = 200 total

COVERAGE

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

PASSED

tests/test_cache.py::TestLlmCache::test_set_and_get

1ms  4

AI ASSESSMENT

Scenario: Test that annotations can be set and retrieved from the cache.**Why Needed:** Prevents bypass attacks by ensuring that LLMCache stores and retrieves annotations in a consistent manner.**Key Assertions:**

- Check that the annotation is stored correctly in the cache.
- Verify that the annotation's confidence value is preserved during retrieval.
- Ensure that the retrieved annotation matches the original annotation.

Confidence: 80%**Tokens:** 286 input + 93 output = 379 total

COVERAGE

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

 tests/test_collector.py

11 tests

PASSED

tests/test_collector.py::TestCollectorCollectionErrors::test_collection_error_structure

1ms 2

AI ASSESSMENT

Scenario:

tests/test_collector.py::TestCollectorCollectionErrors::test_collection_error_structure

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

Key Assertions:

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

Confidence: 80%

Tokens: 124 input + 94 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::TestCollectorCollectionErrors::test_get_collection_errors_initially_empty 1ms 3

AI ASSESSMENT

Scenario:

tests/test_collector.py::TestCollectorCollectionErrors::test_get_collection_errors_initially_empty

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

Key Assertions:

- {'name': 'assert get_collection_errors is a list', 'expected_value': [], 'actual_value': 'get_collection_errors'}

Confidence: 80%

Tokens: 114 input + 94 output = 208 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', 'expected_value': 'None'}

Confidence: 80%

Tokens: 136 input + 66 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)

PASSED

tests/test_collector.py::TestCollectorMarkerExtraction::test_llm_opt_out_default_false

1ms



2

AI ASSESSMENT

Scenario:

tests/test_collector.py::TestCollectorMarkerExtraction::test_llm_opt_out_default_false

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

Key Assertions:

- {'name': 'llm_opt_out is not equal to False', 'expected_value': False, 'actual_value': 'None'}

Confidence: 80%

Tokens: 136 input + 90 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: The output capture feature is not disabled by default.

Key Assertions:

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

Confidence: 80%

Tokens: 104 input + 79 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/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 in the test results.

Key Assertions:

- {'name': 'expected failure', 'value': 'xfail'}

Confidence: 80%

Tokens: 206 input + 80 output = 286 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:

- {'assertion_name': "result.outcome == 'xpassed'", 'expected_value': 'xpassed'}

Confidence: 80%

Tokens: 205 input + 81 output = 286 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 that the `TestCollector` class initializes correctly and returns default values for results, collection errors, and collected count.

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

Key Assertions:

- The `results` attribute of the `collector` object is set to an empty dictionary.
- The `collection_errors` attribute of the `collector` object is an empty list.
- The `collected_count` attribute of the `collector` object is set to 0.

Confidence: 80%

Tokens: 205 input + 131 output = 336 total

COVERAGE

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

AI ASSESSMENT

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

Why Needed: To ensure that the results are sorted by nodeid.

Key Assertions:

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

Confidence: 80%

Tokens: 227 input + 94 output = 321 total

COVERAGE

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

PASSED

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

AI ASSESSMENT

Scenario: Verify that the `handle_collection_finish` method correctly tracks collected and deselected items.

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 + 110 output = 366 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 3

AI ASSESSMENT

Scenario:

tests/test_collector_internals.py::TestCollectorInternals::test_capture_output_disabled_via_handle_report

Why Needed: To ensure that the test does not capture output when `config.capture_failed_output=False` and the integration is via handle_runttest_logreport.

Key Assertions:

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

Confidence: 80%

Tokens: 211 input + 95 output = 306 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: Tests for Collector Internals

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

Key Assertions:

- {'name': 'collector._capture_output', 'expected_result': 'Some error'}

Confidence: 80%

Tokens: 157 input + 62 output = 219 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_stdout

1ms



AI ASSESSMENT

Scenario: TestCollectorInternals::test_capture_output_stdout

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

Key Assertions:

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

Confidence: 80%

Tokens: 157 input + 64 output = 221 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)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_collector_maximal.py::TestCollectorInternals::test_capture_output_truncated

1ms



3

COVERAGE

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

PASSED

tests/test_collector_maximal.py::TestCollectorInternals::test_create_result_with_item_markers

2ms



3

AI ASSESSMENT

Scenario: Test creates a result with item markers.**Why Needed:** Prevents regression in case of item markers being used as requirements or context.**Key Assertions:**

- item.callspec.id = 'param1' is called and its value is set to 'param1'.
- get_closest_marker('llm_opt_out') returns a mock object with the expected behavior.
- get_closest_marker('llm_context') returns a mock object with the correct arguments.
- get_closest_marker('requirement') returns a mock object with the expected requirements.
- item.get_closest_marker('llm_opt_out') is called and its value is set to True.
- item.get_closest_marker('llm_context_override') is called and its value is set to 'complete'.
- item.get_closest_marker('requirement') returns a mock object with the expected requirements.
- result.param_id matches the expected value of 'param1'.

Confidence: 80%**Tokens:** 382 input + 213 output = 595 total

COVERAGE

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

PASSED

tests/test_collector_maximal.py::TestCollectorInternals::test_extract_error_repr_crash

1ms



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



AI ASSESSMENT

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

Why Needed: To ensure the `extract_error` method returns a string that can be used to reconstruct the original error message.

Key Assertions:

- {'name': 'assert extract error is correct', 'description': 'The extracted error string should match the expected value'}

Confidence: 80%

Tokens: 130 input + 93 output = 223 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: test_collector_maximal

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

Key Assertions:

- {'name': 'assert _extract_skip_reason returns None for None longrepr', 'expected_value': 'None'}

Confidence: 80%

Tokens: 130 input + 76 output = 206 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 that 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 + 86 output = 219 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 collection report that fails, it should record this failure.

Why Needed: This test prevents a potential bug where a collection report fails and does not trigger any error messages or warnings.

Key Assertions:

- The `collection_errors` list should contain exactly one item.
- The first item in the `collection_errors` list should have a 'nodeid' of 'test_broken.py'.
- The first item in the `collection_errors` list should have a 'message' that matches 'SyntaxError'.

Confidence: 80%

Tokens: 273 input + 133 output = 406 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 the TestCollector's handle_runttest_rerun method to ensure it correctly handles reruns and updates the report accordingly.

Why Needed: This test prevents a potential regression where the Test Collector does not update the report when a rerun is requested.

Key Assertions:

- The 'rerun' attribute of the report should be set to 1 after handling a runtest_rerun event.
- The final outcome of the runtest_rerun event should be updated to 'failed'.
- The 'rerun_count' and 'final_outcome' attributes of the results dictionary should contain the expected values.

Confidence: 80%

Tokens: 281 input + 143 output = 424 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 is executed, it should record an error setup in the report.

Why Needed: This test prevents a potential regression where the collector might not correctly handle setup failures and instead incorrectly reports a success or no action.

Key Assertions:

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

Confidence: 80%

Tokens: 300 input + 98 output = 398 total

COVERAGE

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

PASSED

tests/test_collector_maximal.py::TestCollectorReportHandling::test_handle_runtest_teardown_failure

1ms

3

AI ASSESSMENT

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

Why Needed: To prevent regression in case of teardown failure, where the collector logs an error instead of a success result.

Key Assertions:

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

Confidence: 80%

Tokens: 391 input + 85 output = 476 total

COVERAGE

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

tests/test_context_compression.py

12 tests

PASSED

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

AI ASSESSMENT

Scenario: Test invalid context compression mode

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

Key Assertions:

- {'message': "context_compression must be one of 'none', 'gzip', or 'lz4'", 'expected': ['context_compression', 'invalid']}

Confidence: 80%

Tokens: 124 input + 86 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	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 this configuration provider.', 'expected_value': -1}

Confidence: 80%

Tokens: 121 input + 75 output = 196 total

COVERAGE

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

PASSED

tests/test_context_compression.py::TestConfigValidation::test_valid_compression_modes

1ms



AI ASSESSMENT

Scenario: TestConfigValidation

Why Needed: To ensure that valid compression modes are correctly validated and do not raise any errors.

Key Assertions:

- {'message': "Context compression should be None or 'lines'", 'expected_value': 'None'}

Confidence: 80%

Tokens: 135 input + 69 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	26 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_compression.py::TestConfigValidation::test_zero_padding_valid

1ms



AI ASSESSMENT

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

Why Needed: Zero padding is a valid configuration option.

Key Assertions:

- {'name': 'config.validate() returns no errors', 'expected_result': 'None'}

Confidence: 80%

Tokens: 122 input + 71 output = 193 total

COVERAGE

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

PASSED

tests/test_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: The context compression should be enabled by default.

Key Assertions:

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

Confidence: 80%

Tokens: 119 input + 71 output = 190 total

COVERAGE

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

PASSED

tests/test_context_compression.py::TestContextCompression::test_compression_mode_lines

1ms

3

AI ASSESSMENT

Scenario:

tests/test_context_compression.py::TestContextCompression::test_compression_mode_lines

Why Needed: Lines compression mode is needed to ensure that the test suite can run without any issues.

Key Assertions:

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

Confidence: 80%

Tokens: 113 input + 79 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/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 to a default value of 2, which is the expected behavior according to the documentation.

Key Assertions:

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

Confidence: 80%

Tokens: 106 input + 94 output = 200 total

COVERAGE

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

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 are marked with a gap indicator.

Key Assertions:

- The count of '#' characters should be zero for contiguous lines.
- # L3:
- # L4:
- # L5:

Confidence: 80%

Tokens: 293 input + 77 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/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	23 lines (ranges: 33, 216, 219-220, 223-228, 231-232, 235-237, 239-240, 242, 244-247, 249)

PASSED

tests/test_context_compression.py::TestExtractCoveredLines::test_empty_coverage

1ms

4

AI ASSESSMENT

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

Why Needed: Empty coverage should return empty string.

Key Assertions:

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

Confidence: 80%

Tokens: 130 input + 70 output = 200 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/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 result contains the expected gap indicator between the two covered lines (# ...).
- The result contains the expected range indicator for line L3: # L3:...
- The result contains the expected range indicator for line L15: # L15:...

Confidence: 80%

Tokens: 274 input + 111 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/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	24 lines (ranges: 33, 216, 219-220, 223-228, 231-232, 235-237, 239-240, 242-247, 249)

PASSED

tests/test_context_compression.py::TestExtractCoveredLines::test_extract_single_line

1ms



AI ASSESSMENT

Scenario: The test verifies that a single covered line is extracted with the correct number of lines padded.

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

Key Assertions:

- The result should contain '# L2:' and '# L3:' and '# L4:'
- The result should include lines 2, 3, and 4 with the correct number of lines padded (1)
- The line numbers in the result should match the original covered lines

Confidence: 80%

Tokens: 302 input + 120 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/prompts.py	23 lines (ranges: 33, 216, 219-220, 223-228, 231-232, 235-237, 239-240, 242, 244-247, 249)

PASSED

tests/test_context_compression.py::TestExtractCoveredLines::test_pading_boundary

1ms

4

AI ASSESSMENT

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

Why Needed: This test prevents a potential bug where padding exceeds the file boundary, potentially causing incorrect results or errors.

Key Assertions:

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

Confidence: 80%

Tokens: 288 input + 86 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/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	23 lines (ranges: 33, 216, 219-220, 223-228, 231-232, 235-237, 239-240, 242, 244-247, 249)

 tests/test_context_limits.py

4 tests

PASSED

tests/test_context_limits.py::test_no_truncation_needed

1ms  4

COVERAGE

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

PASSED

tests/test_context_limits.py::test_smart_distribution

2ms



AI ASSESSMENT

Scenario: tests/test_context_limits.py::test_smart_distribution verifies that the smart distribution algorithm does not waste tokens when F1 and F2 have different needs.

Why Needed: This test prevents regression in the smart distribution logic, where F1 gets more budget than F2 if their needs are unequal.

Key Assertions:

- F1 should be full with ~536 chars (134 tokens).
- F2 got the extra budget of ~400 chars (480 tokens) instead of ~100 tokens (110 tokens).
- F2's content is truncated to ~800 chars (720 tokens) instead of ~440 tokens (110 tokens).

Confidence: 80%

Tokens: 773 input + 145 output = 918 total

COVERAGE

src/pytest_llm_report/collector.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: The test verifies that the splitting logic correctly truncates strings and meets budget requirements.

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

Key Assertions:

- f1 contains 'truncated' in its prompt
- f2 contains 'truncated' in its prompt
- prompt includes 'Present'
- prompt is within budget (~200 tokens total)
- prompt has overhead small (<80 tokens per file)

Confidence: 80%

Tokens: 317 input + 118 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	24 lines (ranges: 243, 245, 264, 266, 270-272, 274, 277, 279-280, 283, 286, 290-291, 294-295, 298-299, 305, 307, 310, 312, 314)
src/pytest_llm_report/llm/utils.py	30 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 90-91, 93-94, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_limits.py::test_truncation_logic

1ms  4

AI ASSESSMENT

Scenario: Test truncation logic when creating large context files.**Why Needed:** Prevents a potential bug where the test passes even though the context is too long, causing unnecessary computation and potentially incorrect results.**Key Assertions:**

- The prompt should be truncated to fit within 100 tokens minus system prompt overhead (~40-50 tokens) minus header overhead (~20 tokens)
- Context should be very small or empty
- Prompt contains '[... truncated]' or 'Relevant context' indicating truncation

Confidence: 80%**Tokens:** 397 input + 116 output = 513 total

COVERAGE

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

 tests/test_context_util.py

28 tests

PASSED

tests/test_context_util.py::TestCollapseEmptyLines::test_collapse_three_empty_lines

1ms



AI ASSESSMENT

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

Why Needed: Because the current implementation does not handle three or more empty lines correctly.

Key Assertions:

- {'assertion': 'The result of collapsing three+ empty lines is 2!', 'expected_result': 'line1\n\nline2'}

Confidence: 80%

Tokens: 128 input + 90 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/context_util.py	1 lines (ranges: 108)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestCollapseEmptyLines::test_many_empty_lines

1ms



AI ASSESSMENT

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

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

Key Assertions:

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

Confidence: 80%

Tokens: 127 input + 83 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/context_util.py	1 lines (ranges: 108)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestCollapseEmptyLines::test_preserve_two_empty_lines

1ms

3

AI ASSESSMENT

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

Why Needed: Preserves up to 2 consecutive newlines.

Key Assertions:

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

Confidence: 80%

Tokens: 125 input + 81 output = 206 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	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: Preserve single newlines in collapsed context.

Key Assertions:

- {'name': 'result', 'expected_value': 'line1\nline2\nline3'}

Confidence: 80%

Tokens: 121 input + 74 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	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: The test is necessary because the current implementation does not correctly handle empty lines in the input source.

Key Assertions:

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

Confidence: 80%

Tokens: 137 input + 90 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/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 works as expected and does not introduce any unexpected behavior.**Key Assertions:**

- {'name': 'expected Optimizations to be applied', 'value': 'All optimizations should be applied'}
- {'name': 'Optimization Order', 'value': 'The combined optimization process should apply all optimizations in the expected order.'}

Confidence: 80%**Tokens:** 96 input + 115 output = 211 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestOptimizeContext::test_default_strips_docs_only

1ms

3

AI ASSESSMENT

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

Why Needed: The default behavior of `optimize_context` is to strip all code, including docstrings and comments. However, this can be problematic when working with complex codebases where docstrings provide important context.

Key Assertions:

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

Confidence: 80%

Tokens: 100 input + 107 output = 207 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestOptimizeContext::test_empty_source

1ms



AI ASSESSMENT

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

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

Confidence: 80%**Tokens:** 95 input + 71 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/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



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 code with only whitespace correctly.**Key Assertions:**

- The output of the optimize_context function should be a string containing only whitespace characters.
- The input to the optimize_context function should be a string containing only whitespace characters.

Confidence: 80%**Tokens:** 115 input + 95 output = 210 total

COVERAGE

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

- {'assertion': 'source: def foo():\\n# ...', 'expected': 'def foo():\\n# ...'}

Confidence: 80%**Tokens:** 95 input + 80 output = 175 total

COVERAGE

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



AI ASSESSMENT

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

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

Key Assertions:

- {'assertion_type': 'string', 'expected_value': 'def foo():\n # This is a comment\n pass', 'actual_value': {'scenario': 'tests/test_context_util.py::TestOptimizeContext::test_strip_comments_only', 'why_needed': 'To optimize the context by removing unnecessary comments.', 'key_assertions': ['def foo():\n # This is a comment\n pass']}, 'reason': "The docstring 'This is a comment' should be removed."}

Confidence: 80%

Tokens: 95 input + 161 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/context_util.py	14 lines (ranges: 81-82, 86, 88-90, 93, 108, 124, 126, 129-130, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

AI ASSESSMENT

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

Why Needed: The test is failing because the optimizer strips out unnecessary code.

Key Assertions:

- {'name': 'source_code', 'expected': 'def foo():\n pass', 'actual': 'def foo():\n # This line will be stripped by the optimizer'}

Confidence: 80%

Tokens: 94 input + 97 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	6 lines (ranges: 108, 124, 126, 129, 133, 135)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestStripComments::test_comment_after_string_with_hash

1ms



AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripComments::test_comment_after_string_with_hash

Why Needed: To ensure that comments are stripped from strings containing hash symbols (#) after they appear within string literals.

Key Assertions:

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

Confidence: 80%

Tokens: 134 input + 100 output = 234 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestStripComments::test_escaped_quotes

1ms



AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripComments::test_escaped_quotes**Why Needed:** To handle escaped quotes in strings correctly.**Key Assertions:**

- {'description': 'The function should return the original string without any escaped quotes.', 'expected_result': 's = "escaped \\\"}'}
- actual_result": {u's': u'\u201c'} # The actual result is a Unicode escape sequence for a double quote character, not just a literal double quote.

Confidence: 80%**Tokens:** 133 input + 120 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/context_util.py	7 lines (ranges: 81-82, 86, 88-90, 93)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestStripComments::test_mixed_quotes

1ms  3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripComments::test_mixed_quotes**Why Needed:** To strip quotes from a string containing both single and double quotes.**Key Assertions:**

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

Confidence: 80%**Tokens:** 101 input + 77 output = 178 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestStripComments::test_no_comments

1ms  3

AI ASSESSMENT

Scenario: tests/test_context_util.py::TestStripComments::test_no_comments**Why Needed:** To strip comments from the source code to ensure correct context.**Key Assertions:**

- {'assertion_type': 'source_code', 'expected_result': 'def foo():\n# This is a comment\npass'}

Confidence: 80%**Tokens:** 91 input + 84 output = 175 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestStripComments::test_preserve_hash_in_double_quoted_string

1ms

3

AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripComments::test_preserve_hash_in_double_quoted_string

Why Needed: To preserve '#' inside double-quoted strings in the context of strip_comments() function.

Key Assertions:

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

Confidence: 80%

Tokens: 135 input + 99 output = 234 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestStripComments::test_preserve_hash_in_single_quoted_string

1ms

3

AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripComments::test_preserve_hash_in_single_quoted_string

Why Needed: To preserve the hash (#) inside single-quoted strings when stripping comments.

Key Assertions:

- {'expected': '# is preserved in single-quoted string', 'actual': "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_strip_simple_comment

1ms



AI ASSESSMENT

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

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

Key Assertions:

- {'expected_result': 'x = 1', 'actual_result': 'x = 1'}

Confidence: 80%

Tokens: 119 input + 76 output = 195 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestStripComments::test_strip_standalone_comment

1ms



AI ASSESSMENT

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

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

Key Assertions:

- {'expected': {'line': 1, 'column': 4}, 'actual': {'line': 2, 'column': 0}}

Confidence: 80%

Tokens: 99 input + 85 output = 184 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestStripDocstrings::test_handles_syntax_error_gracefully

1ms

3

AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripDocstrings::test_handles_syntax_error_gracefully

Why Needed: The test is checking if the function correctly handles syntax errors in the input source code.

Key Assertions:

- {'assertion': 'strip_docstrings() returns the original source on syntax error.', 'expected_result': 'def foo(unclosed paren', 'actual_result': 'def foo(unclosed paren'}

Confidence: 80%

Tokens: 119 input + 107 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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:** This test is needed because the current implementation of context_util.strip_docstrings() does not handle multiple docstrings correctly.**Key Assertions:**

- {'assertion': 'The function should remove all docstrings from the source code.', 'expected_output': 'No docstring remains in the source code.'}
- {'assertion': 'The function should return an empty string if there are no docstrings to strip.', 'expected_output': ''}

Confidence: 80%**Tokens:** 95 input + 131 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	30 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-59, 61-62, 64, 66-69)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestStripDocstrings::test_preserves_multiline_data_strings

1ms

3

AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripDocstrings::test_preserves_multiline_data_strings

Why Needed: Preserve multiline data strings in docstrings.

Key Assertions:

- {'name': 'docstring triple quotes are preserved', 'expected_value': 'def foo():\n """\n This is a triple quoted string.\n """', 'actual_value': 'def foo():\n This is a triple quoted string.', 'message': 'Expected docstring triple quotes to be preserved, but were not.'}

Confidence: 80%

Tokens: 103 input + 133 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/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



AI ASSESSMENT

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

Why Needed: Preserve regular strings in test output.

Key Assertions:

- {'name': 'strip_docstrings', 'expected_value': 'x = "hello world"'}

Confidence: 80%

Tokens: 102 input + 74 output = 176 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestStripDocstrings::test_preserves_strings_in_structures

1ms



AI ASSESSMENT

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

Why Needed: Preserving strings in structures is a crucial aspect of the context utility.

Key Assertions:

- {'assertion': "The string 'string 1' is preserved in the output.", 'expected_output': '\"string 1\"', 'actual_output': "'''string 1'''"}
- {'assertion': "The string ''"}

Confidence: 80%

Tokens: 146 input + 117 output = 263 total

COVERAGE

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



AI ASSESSMENT

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

Why Needed: This test ensures that the `strip_multiline_docstring` function correctly removes multiline docstrings from Python code.

Key Assertions:

- {'name': 'docstring removal', 'expected': 'The function should remove all docstrings, including those with multiple lines.'}

Confidence: 80%

Tokens: 97 input + 95 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/context_util.py	29 lines (ranges: 27, 29, 33, 35-36, 38-45, 47-49, 51-52, 55-56, 58-59, 61-62, 64, 66-69)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_context_util.py::TestStripDocstrings::test_strip_triple_double_quoted_docstring

1ms

3

AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripDocstrings::test_strip_triple_double_quoted_docstring

Why Needed: To ensure that context managers are properly stripped of triple double-quoted docstrings.

Key Assertions:

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

Confidence: 80%

Tokens: 106 input + 82 output = 188 total

COVERAGE

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

PASSED

tests/test_context_util.py::TestStripDocstrings::test_strip_triple_single_quoted_docstring

1ms



AI ASSESSMENT

Scenario:

tests/test_context_util.py::TestStripDocstrings::test_strip_triple_single_quoted_docstring

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

Key Assertions:

- {'assertion': 'striped source code contains a triple single-quoted docstring', 'expected_result': "'''\\ndef foo():\\n>>> foo()'''\\'"}
• actual_result

Confidence: 80%

Tokens: 106 input + 151 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/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 GeminiProvider's _parse_preferred_models method with edge cases.**Why Needed:** This test prevents a potential bug where the method returns incorrect results when given an empty or invalid model configuration.**Key Assertions:**

- The function should return an empty list when the 'model' parameter is None.
- The function should return an empty list when the 'model' parameter is set to 'All'.
- The function should not return any models when the 'model' parameter is an invalid string (e.g., 'abc').

Confidence: 80%**Tokens:** 273 input + 126 output = 399 total

COVERAGE

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

PASSED

tests/test_coverage_boosters.py::TestCoverageBoosters::test_gemini_rate_limiter_edge_math

1ms



AI ASSESSMENT

Scenario: Verify that the rate limiter correctly handles edge cases where there are more tokens than available.

Why Needed: This test prevents a potential bug where the rate limiter fails to handle situations with excessive token usage, leading to incorrect behavior or errors.

Key Assertions:

- The `next_available_in` method should return an error when there are more tokens than available.
- The `next_available_in` method should return 0 when both limits have been exceeded.
- The rate limiter should not attempt to process requests beyond the available token limit.

Confidence: 80%

Tokens: 273 input + 127 output = 400 total

COVERAGE

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

PASSED

tests/test_coverage_boosters.py::TestCoverageBoosters::test_models_to_dict_variants

1ms



3

AI ASSESSMENT

Scenario: Test that the `models_to_dict` method returns accurate coverage percentages for SourceCoverageEntry objects.

Why Needed: This test prevents regression in the `models_to_dict` method by ensuring it accurately calculates coverage percentages for SourceCoverageEntry objects.

Key Assertions:

- The 'coverage_percent' key of the dictionary returned by `to_dict()` contains the correct value.
- The 'error' key of the dictionary returned by `to_dict()` contains the expected error message.
- The 'duration' key of the dictionary returned by `to_dict()` contains the correct duration value.
- The 'start_time', 'end_time', and 'duration' keys are present in the dictionary with their respective values.
- The coverage percentage is calculated correctly based on the number of statements, covered, missed, and coverage percent.
- The LLM annotation error message is correctly extracted from the annotation object.

Confidence: 80%**Tokens:** 318 input + 198 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/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)

PASSED

tests/test_coverage_map.py::TestCoverageMapper::test_create_mapper

1ms



AI ASSESSMENT

Scenario: tests/test_coverage_map.py::TestCoverageMapper::test_create_mapper**Why Needed:** Mapper initialization should be tested to ensure it creates a new instance with the correct configuration.**Key Assertions:**

- {'name': 'mapper is an instance of CoverageMapper', 'expected_type': 'CoverageMapper', 'actual_type': 'type'}
- {'name': 'config is initialized correctly', 'expected_value': 'Config', 'actual_value': 'type'}

Confidence: 80%**Tokens:** 109 input + 118 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/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': 'list'}

Confidence: 80%**Tokens:** 110 input + 78 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/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 present.

Why Needed: Prevents a potential bug where the test fails due to missing coverage data.

Key Assertions:

- The `map_coverage()` method should return an empty dictionary when no coverage file exists.
- The `map_coverage()` method should not have any warnings when no coverage file is present.
- The `map_coverage()` method should correctly handle the absence of a coverage file by returning an empty dictionary.

Confidence: 80%

Tokens: 277 input + 119 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/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` extracts all phases when the `include_phase` parameter is set to 'all'.

Why Needed: This test prevents a regression where the coverage map does not include all phases when `include_phase=all`.

Key Assertions:

- The function `_extract_nodeid` of the `CoverageMapper` class correctly extracts node IDs from the given path.
- The function `_extract_nodeid` of the `CoverageMapper` class returns the expected node ID for each phase.
- The function `_extract_nodeid` of the `CoverageMapper` class handles cases where the input path is empty or contains only one phase.
- The function `_extract_nodeid` of the `CoverageMapper` class correctly handles cases where the input path starts with a phase name (e.g., 'test.py::test_foo|')
- The function `_extract_nodeid` of the `CoverageMapper` class does not modify any node IDs.
- The function `_extract_nodeid` of the `CoverageMapper` class raises an `AssertionError` if the input path is invalid or malformed.

Confidence: 80%

Tokens: 279 input + 243 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/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, and return None as per the expected result.

Key Assertions:

- {'expected_result': 'None', 'actual_result': 'None'}

Confidence: 80%

Tokens: 128 input + 83 output = 211 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 filter out setup phase when include_phase=run.

Key Assertions:

- {'name': 'nodeid is None', 'expected_value': '', 'actual_value': 'None'}

Confidence: 80%

Tokens: 139 input + 82 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/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 run phase context in coverage reports.

Key Assertions:

- {'assertion_type': 'node ID extraction', 'expected_value': 'test.py::test_foo', 'actual_value': 'test.py::test_foo'}

Confidence: 80%

Tokens: 145 input + 99 output = 244 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the coverage mapper's behavior when encountering an exception while extracting contexts by line number.

Why Needed: Prevents a regression where the test fails due to an unexpected exception being raised during context extraction.

Key Assertions:

- mock_data.contexts_by_lineno.side_effect is set to contexts_side_effect with the correct side effect.
- call_count[0] is incremented correctly before raising the exception.
- the exception is not raised within the first call to contexts_by_lineno
- the exception raises an Exception object with the correct message and context number
- the exception does not cause any other tests to fail

Confidence: 80%

Tokens: 332 input + 141 output = 473 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	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 the case where coverage data has no measured files.

Key Assertions:

- {'name': 'result is an empty dictionary', 'expected_result': '{}'}

Confidence: 80%

Tokens: 136 input + 75 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/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



AI ASSESSMENT

Scenario: Test Extract Contexts**Why Needed:** To skip non-Python files from coverage report.**Key Assertions:**

- {'assertion': "mocked mock_data.measured_files.return_value is equal to ['file.txt', 'data.json']", 'expected_result': ['file.txt', 'data.json']}
- {'assertion': 'mocked mock_data.contexts_by_lineno.return_value is empty', 'expected_result': {}}

Confidence: 80%**Tokens:** 154 input + 111 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/coverage_map.py	14 lines (ranges: 44-45, 118, 121-122, 127, 131-135, 144-146)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_coverage_map_coverage.py::TestLoadCoverageData::test_coverage_not_installed

1ms



AI ASSESSMENT

Scenario: TestLoadCoverageData

Why Needed: To test that coverage.py is installed and properly loaded into the environment.

Key Assertions:

- {'name': 'coverage.py is imported correctly', 'expected_result': 'coverage.py should be imported from the Python path.'}
- {'name': 'CoverageMapper is created successfully', 'expected_result': 'CoverageMapper should be created with a Config object.'}

Confidence: 80%

Tokens: 166 input + 104 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/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 that the function returns None when no .coverage file exists.

Key Assertions:

- {'description': 'The result of _load_coverage_data() is not None.', 'expected_value': 'None'}
- {'description': 'Any warnings are present in the mapper.warnings list.', 'expected_value': ['W001']}

Confidence: 80%

Tokens: 153 input + 100 output = 253 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	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 that analysis exception handling prevents adding an empty coverage report.**Why Needed:** To prevent adding an empty coverage report when analysis2 raises an exception.**Key Assertions:**

- The test verifies that the `map_source_coverage` function returns an empty list of warnings.
- The test verifies that any warning messages contain 'COVERAGE_ANALYSIS_FAILED'.
- The test verifies that the `map_source_coverage` function sets the expected warnings to be empty.

Confidence: 80%**Tokens:** 286 input + 108 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/coverage_map.py	22 lines (ranges: 44-45, 243-244, 246-248, 250, 252-254, 259, 261, 263-268, 271, 299-300)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED

tests/test_coverage_map_coverage.py::TestMapSourceCoverage::test_empty_statements

1ms



AI ASSESSMENT

Scenario: TestMapSourceCoverage test_empty_statements**Why Needed:** To ensure that the function correctly handles a case where there are no statements in the file.**Key Assertions:**

- {'name': 'mock_cov.get_data.return_value is empty list', 'expected': [], 'actual': []}
- {'name': 'mock_cov.analysis2.return_value is correct', 'expected': ['empty.py', [], [], [], []], 'actual': ['empty.py', [], [], [], []]}

Confidence: 80%**Tokens:** 178 input + 123 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/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 the coverage map does not include all test files when omitting tests from coverage.

Key Assertions:

- The `covered` count for each file should be greater than or equal to 1.
- The `missed` count for each file should be less than or equal to 0.
- All test files (`/project/tests/test_foo.py`) should be included in the coverage map.

Confidence: 80%

Tokens: 322 input + 122 output = 444 total

COVERAGE

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



4

AI ASSESSMENT

Scenario: Test Map Source Coverage**Why Needed:** This test is needed to ensure that non-Python files are skipped from coverage reports.**Key Assertions:**

- {'assertion': 'mock_data.measured_files.return_value should be an empty list', 'expected_result': [], 'message': 'Expected mock_data.measured_files.return_value to be an empty list'}
- {'assertion': 'mock_data.contexts_by_lineno.return_value should be an empty dictionary', 'expected_result': {}, 'message': 'Expected mock_data.contexts_by_lineno.return_value to be an empty dictionary'}

Confidence: 80%**Tokens:** 154 input + 143 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/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:**

- {'name': 'Test case 1: Test file should be skipped', 'description': 'The test file should not be included in the coverage report.', 'expected_result': [], 'actual_result': []}
- {'name': 'Test case 2: Test file is included in the coverage report', 'description': 'The test file should be included in the coverage report.', 'expected_result': ['Expected result'], 'actual_result': ['Actual result']}

Confidence: 80%**Tokens:** 182 input + 149 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/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 coverage mapping for 'all' phase configuration.**Key Assertions:**

- The mapper should return the same nodeid for any phase.
- The mapper should match the expected nodeids for each phase.
- No unexpected nodeids are returned for phases other than 'setup', 'run', and 'teardown'.

Confidence: 80%**Tokens:** 305 input + 97 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/coverage_map.py	11 lines (ranges: 44-45, 216, 220, 224-225, 228-229, 231, 233, 236)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_empty_string

1ms

4

AI ASSESSMENT

Scenario:

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_empty_string

Why Needed: The test extracts node ID from an empty string, which should return None.

Key Assertions:

- {'expected_value': 'None', 'actual_value': 'assert mapper._extract_nodeid("") is None'}

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/coverage_map.py	4 lines (ranges: 44-45, 216-217)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_none

1ms



AI ASSESSMENT

Scenario:

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_none

Why Needed: None input returns None.

Key Assertions:

- {'expected_value': 'None', 'actual_value': 'None'}

Confidence: 80%

Tokens: 114 input + 66 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/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 regression where `run` phase is not included in coverage.**Key Assertions:**

- The function `extract_nodeid` should return the node ID when the phase matches.
- The function `extract_nodeid` should return None when the phase does not match.
- The function `extract_nodeid` should not throw an error or raise an exception when the phase is setup or teardown.

Confidence: 80%**Tokens:** 297 input + 110 output = 407 total

COVERAGE

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

PASSED

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_setup_phase_config

1ms



4

AI ASSESSMENT

Scenario: Test that setup phase is correctly filtered when configured.**Why Needed:** Prevents a potential bug where the test would incorrectly filter out nodes in the setup phase due to missing configuration.**Key Assertions:**

- The function _extract_nodeid should return the nodeid of the specified module and phase.
- If the phase does not match 'setup', the function should return None for that phase.
- If the phase is neither 'setup' nor 'run' (or 'teardown'), the function should return None for those phases as well.

Confidence: 80%**Tokens:** 293 input + 126 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/coverage_map.py	12 lines (ranges: 44-45, 216, 220, 224-225, 228-229, 231-233, 236)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_teardown_phase_config

1ms



4

AI ASSESSMENT

Scenario: Test that teardown phase is correctly filtered when configured.**Why Needed:** This test prevents a potential bug where the teardown phase configuration does not filter out nodeids for certain phases.**Key Assertions:**

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

Confidence: 80%**Tokens:** 296 input + 125 output = 421 total

COVERAGE

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

PASSED

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_without_pipe

1ms

4

AI ASSESSMENT

Scenario:

tests/test_coverage_map_coverage.py::TestPhaseFiltering::test_extract_nodeid_without_pipe

Why Needed: To ensure that the node id is extracted correctly when there are no phase delimiters in the import statement.

Key Assertions:

- {'expected': 'test_foo.py::test_bar', 'actual': 'test_foo.py::test_bar'}

Confidence: 80%

Tokens: 136 input + 93 output = 229 total

COVERAGE

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

 tests/test_coverage_map_maximal.py

9 tests

PASSED

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

AI ASSESSMENT

Scenario: Test that the test_extract_contexts_full_logic function exercises all paths in _extract_contexts.

Why Needed: This test prevents regression where a file's coverage is not fully included in the extracted contexts.

Key Assertions:

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

Confidence: 80%

Tokens: 413 input + 128 output = 541 total

COVERAGE

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

PASSED

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

COVERAGE

src/pytest_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 coverage mapper's ability to extract node IDs for different phases and contexts.

Why Needed: This test prevents regression by ensuring that the coverage mapper correctly identifies missing lines in the code.

Key Assertions:

- The function `_extract_nodeid` should return '`test.py::test`' when called with a valid context (e.g., '`test.py::test|setup`').
- The function `_extract_nodeid` should return `None` when called with an invalid context (e.g., '`test.py::test_no_phase`').

Confidence: 80%

Tokens: 323 input + 122 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	14 lines (ranges: 44-45, 216, 220, 224-225, 228-229, 231-234, 236, 239)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_load_coverage_data_no_files

1ms



5

AI ASSESSMENT

Scenario: Test the case where no coverage files exist.**Why Needed:** This test prevents a potential bug where the function does not raise an exception when no coverage files are found, but instead returns None or other unexpected values.**Key Assertions:**

- The function `mapper._load_coverage_data()` should return `None` when there are no .coverage files.
- The function `mapper.warnings` should contain exactly one warning with code 'W001'.
- The current implementation of the test does not raise an exception when no coverage files exist, but instead returns None or other unexpected values.

Confidence: 80%**Tokens:** 276 input + 135 output = 411 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/coverage_map.py	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 that the `CoverageMapper` raises an error when trying to load corrupted coverage data.

Why Needed: To prevent a regression where the test fails due to a corrupt coverage file being loaded, which would cause the `CoverageMapper` to incorrectly report no coverage data.

Key Assertions:

- The function `load_coverage_data()` should raise an exception when it encounters a corrupted coverage file.
- Any warnings generated by the `CoverageMapper` should contain the message 'Failed to read coverage data'.
- The test should fail if any warnings are generated, indicating that the coverage data is corrupt.

Confidence: 80%

Tokens: 343 input + 137 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	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, which can lead to incorrect coverage data being reported if not properly updated.

Key Assertions:

- The mock instances of `CoverageData` should have been called with at least two times.
- The `update` method of the mock instances should be called.
- The number of calls to `update` for each mock instance should be greater than or equal to 1.
- Each mock instance's `update` method should be called before any other method is called on it.
- The `update` method should not be called multiple times with the same arguments if it was already called once.
- If a mock instance's `update` method is called, it should only be called once per test.
- If a mock instance's `update` method is called multiple times, it should raise an AssertionError.

Confidence: 80%

Tokens: 378 input + 217 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/coverage_map.py	15 lines (ranges: 44-45, 72-73, 83, 86, 88, 92, 94, 98, 101-104, 106)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_map_coverage_no_data 1ms 4

AI ASSESSMENT

Scenario: Test that the test_map_coverage_no_data function returns an empty dictionary when _load_coverage_data returns None.

Why Needed: Prevents a potential bug where the mapper does not handle the case of no coverage data being loaded.

Key Assertions:

- The mapper should return an empty dictionary when _load_coverage_data returns None.
- _load_coverage_data() was called with argument None
- No coverage data is available for mapping
- mapper._load_coverage_data() did not return a non-None value
- mapper.map_coverage() should have been called with an empty dictionary

Confidence: 80%

Tokens: 228 input + 135 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/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: The test verifies that the `map_source_coverage` method skips files with errors during analysis.

Why Needed: This test prevents a regression where an error in analysis results in incorrect skipping of source code files.

Key Assertions:

- mock_cov.analysis2.assert_called_once_with(mock_data)
- mock_cov.get_data.return_value.measured_files.return_value == ['app.py']
- entries == []

Confidence: 80%

Tokens: 274 input + 98 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/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: Test should exercise all paths in map_source_coverage to ensure comprehensive coverage of source files.

Why Needed: This test prevents regression by ensuring that the CoverageMapper is correctly analyzing all possible paths from the map_source_coverage configuration.

Key Assertions:

- The function mapper.map_source_coverage() returns a list containing one entry with file_path='app.py', statements=3, covered=2, missed=1, and coverage_percent=66.67.
- The entries in the returned list should include all possible paths from map_source_coverage configuration.
- All files in the source directory should be included in the analysis.
- Statements should be counted correctly based on their actual values.
- Covered statements should match the number of statements in the file.
- Missed statements should be zero or less, not greater than the total number of statements.
- Coverage percentage should be between 0 and 100, inclusive.

Confidence: 80%

Tokens: 345 input + 202 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/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: Prevents a potential bug where an invalid or unknown code is passed to the `make_warning` function, resulting in unexpected behavior or errors.

Key Assertions:

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

Confidence: 80%

Tokens: 236 input + 139 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/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: Prevents a potential bug where the warning code for W001_NO_COVERAGE is set to an incorrect value, potentially leading to unexpected behavior or errors in the application.

Key Assertions:

- {'message': 'assert WarningCode.W001_NO_COVERAGE.value == "W001"', 'description': "Checks that the assertion correctly sets the warning code to 'W001'"}
- {'message': 'assert WarningCode.W101_LLM_ENABLED.value == "W101"', 'description': "Checks that the assertion correctly sets the warning code to 'W101'"}
- {'message': 'assert WarningCode.W201_OUTPUT_PATH_INVALID.value == "W201"', 'description': "Checks that the assertion correctly sets the warning code to 'W201'"}
- {'message': 'assert WarningCode.W301_INVALID_CONFIG.value == "W301"', 'description': "Checks that the assertion correctly sets the warning code to 'W301'"}
- {'message': 'assert WarningCode.W401_AGGREGATE_DIR_MISSING.value == "W401"', 'description': "Checks that the assertion correctly sets the warning code to 'W401'"}

Confidence: 80%

Tokens: 240 input + 256 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_errors.py::test_warning_to_dict

1ms 3

AI ASSESSMENT

Scenario: Test ReportWarning.to_dict() method to ensure it returns the correct warning code and detail.

Why Needed: This test prevents a potential bug where the 'to_dict()' method of ReportWarning class does not return the expected warning details for certain warnings.

Key Assertions:

- The function should return a dictionary with the correct warning code and detail.
- The function should return the same warning details even when no detail is provided.
- The function should handle WarningCode.W001_NO_COVERAGE correctly.
- The function should handle WarningCode.W101_LLM_ENABLED correctly.
- The function should not raise any errors for valid warnings with missing detail.

Confidence: 80%

Tokens: 276 input + 148 output = 424 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 Verify creation of warning with known code.

Why Needed: Prevents regression in case unknown code is used.

Key Assertions:

- The function make_warning() returns a Warning object with the correct code and message.
- The detail attribute of the Warning object is None, indicating no additional information about the warning.
- The assertion that w.code == WarningCode.W101_LLM_ENABLED checks if the created warning has the correct code.
- The assertions that w.message == WARNING_MESSAGES[WarningCode.W101_LLM_ENABLED] and w.detail is None check if the created warning has the expected message and no additional information.
- If unknown code was used, make_warning() should raise an exception or return a Warning object with incorrect attributes.

Confidence: 80%

Tokens: 222 input + 167 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/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_unknown_code

1ms



AI ASSESSMENT

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

Why Needed: To test the behavior of `make_warning` when it encounters an unknown warning code.

Key Assertions:

- {'name': 'missing_message_for_valid_code', 'expected': 'Unknown warning.', 'actual': 'Unknown warning.'}
- {'name': 'restoring_original_message', 'expected': 'WARNING_MESSAGES[missing_code] = old_message', 'actual': 'WARNING_MESSAGES[missing_code] = old_message'}

Confidence: 80%

Tokens: 202 input + 132 output = 334 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_with_detail

1ms



AI ASSESSMENT

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

Why Needed: The test is needed to ensure that the `make_warning` function creates a warning with detail.

Key Assertions:

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

Confidence: 80%

Tokens: 127 input + 91 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/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



AI ASSESSMENT

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

Why Needed: Because the `code.value` attribute is expected to be a string.

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 + 102 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)

PASSED

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

AI ASSESSMENT

Scenario: Tests for errors maximal

Why Needed: To test the warning_to_dict method of ReportWarning class without including detail in the dictionary.

Key Assertions:

- {'assertion': "data == {'code': 'W001', 'message': 'No coverage'}", 'expected_result': {'code': 'W001', 'message': 'No coverage'}}}

Confidence: 80%

Tokens: 147 input + 92 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/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

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

COVERAGE

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

 tests/test_fs.py

12 tests

PASSED

tests/test_fs.py::TestIsPythonFile::test_non_python_file

1ms



AI ASSESSMENT

Scenario: tests/test_fs.py::TestIsPythonFile::test_non_python_file**Why Needed:** The test should return False for non-.py files.**Key Assertions:**

- {'name': 'is_python_file() should return False for non-.py files', 'expected_result': False, 'actual_result': 'foo/bar.txt'}
- {'name': 'is_python_file() should return False for non-.py files (2)', 'expected_result': False, 'actual_result': 'foo/bar.pyc'}

Confidence: 80%**Tokens:** 115 input + 130 output = 245 total

COVERAGE

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

PASSED

tests/test_fs.py::TestIsPythonFile::test_python_file

1ms  3

AI ASSESSMENT

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

- {'name': 'type', 'expected': 'str', 'actual': 'bool'}
- {'name': 'extension', 'expected': '.py', 'actual': 'True'}

Confidence: 80%**Tokens:** 98 input + 103 output = 201 total

COVERAGE

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

PASSED

tests/test_fs.py::TestMakeRelative::test_makes_path_relative

1ms  3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestMakeRelative::test_makes_path_relative**Why Needed:** To ensure that the `make_relative` function correctly makes a path relative to the test directory.**Key Assertions:**

- {'path': '/tmp/test_dir/subdir/file.py', 'expected_result': '/tmp/test_dir/file.py'}

Confidence: 80%**Tokens:** 144 input + 86 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/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', 'message': "Expected result to be 'foo/bar'")}

Confidence: 80%

Tokens: 107 input + 91 output = 198 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	7 lines (ranges: 30, 33, 36, 39, 42, 55-56)

PASSED

tests/test_fs.py::TestNormalizePath::test_already_normalized

1ms  3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestNormalizePath::test_already_normalized

Why Needed: The function should be able to handle already-normalized paths without raising an error.

Key Assertions:

- {'expected': 'foo/bar', 'actual': 'normalize_path('}

Confidence: 80%

Tokens: 96 input + 73 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::TestNormalizePath::test_forward_slashes

1ms  3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestNormalizePath::test_forward_slashes

Why Needed: Converts backslashes in path strings to forward slashes for correct behavior in Windows.

Key Assertions:

- {'expected': '/foo/bar', 'actual': '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



AI ASSESSMENT

Scenario: tests/test_fs.py::TestNormalizePath::test_strips_trailing_slash**Why Needed:** To ensure that the `normalize_path` function correctly removes trailing slashes from file paths.**Key Assertions:**

- {'name': 'assert normalize path returns correct result', 'expected_result': 'foo/bar', 'actual_result': 'foo/bar/'}
 -

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	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:**

- {'message': 'should skip path', 'condition': "assert should_skip_path('tests/conftest.py', exclude_patterns=['test*']) is True"}
- {'message': 'should not skip path', 'condition': "assert should_skip_path('src/module.py', exclude_patterns=['test*']) is False"}

Confidence: 80%**Tokens:** 126 input + 126 output = 252 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	15 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116-117, 119-121, 123)

PASSED

tests/test_fs.py::TestShouldSkipPath::test_normal_path

1ms  3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_normal_path**Why Needed:** The test should not be skipped for normal paths.**Key Assertions:**

- {'name': 'should not skip normal path', 'expected_result': 'True'}

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/fs.py	11 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116, 123)

PASSED

tests/test_fs.py::TestShouldSkipPath::test_skips_git

1ms  3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_skips_git**Why Needed:** The test is checking if the `should_skip_path` function correctly identifies .git directories as requiring skipping.**Key Assertions:**

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

Confidence: 80%**Tokens:** 99 input + 87 output = 186 total

COVERAGE

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

PASSED

tests/test_fs.py::TestShouldSkipPath::test_skips_pycache

1ms  3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_skips_pycache**Why Needed:** Because the file is a .pyc file, which is cached by Python and should not be included in the test output.**Key Assertions:**

- assert should_skip_path('foo/__pycache__/bar.pyc') is True

Confidence: 80%**Tokens:** 109 input + 80 output = 189 total

COVERAGE

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

PASSED

tests/test_fs.py::TestShouldSkipPath::test_skips_venv

1ms 3

AI ASSESSMENT

Scenario: tests/test_fs.py::TestShouldSkipPath::test_skips_venv**Why Needed:** Because the test case is checking if venv directories are skipped, which can cause issues with testing and environment setup.**Key Assertions:**

- {'expected': True, 'actual': 'True'}
- {'expected': '.venv/lib/python/site.py', 'actual': '.venv/lib/python/site.py'}

Confidence: 80%**Tokens:** 121 input + 108 output = 229 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: Verifies that a non-.py file does not match the expected criteria.

Why Needed: Prevents regression in cases where a user might mistakenly assume a non-.py file is Python code.

Key Assertions:

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

Confidence: 80%

Tokens: 210 input + 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: Testing if a file is a Python file.

Why Needed: Prevents a potential bug where the test might incorrectly identify non-Python files as such.

Key Assertions:

- The function `is_python_file` correctly identifies .py files and returns True.
- It handles paths with spaces in them by converting them to a single space.
- It supports relative paths by resolving them to absolute paths.
- It ignores non-existent files, returning False for those.
- It does not incorrectly identify non-Python file extensions (e.g., .txt).
- It correctly identifies Python modules with different casing (e.g., `module.py` and `Module.py`).

Confidence: 80%

Tokens: 212 input + 155 output = 367 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 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



AI ASSESSMENT

Scenario: Test makes sure make_relative function returns a normalized absolute path when the input paths are not relative to each other.

Why Needed: This test prevents regression where make_relative function fails to return a normalized absolute path when the input paths are not relative to each other.

Key Assertions:

- The 'project1' should be included in the result.
- The 'file.py' should also be included in the result.
- The result should be an absolute path (i.e., it should start with a slash).
- The relative_to parameter should fail and return None.
- The make_relative function should correctly handle cases where the input paths are not relative to each other.
- The test should pass even if the input paths have different names or extensions.

Confidence: 80%

Tokens: 301 input + 171 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	12 lines (ranges: 30, 33, 36, 39, 42, 55, 58-60, 63, 65, 67)

PASSED

tests/test_fs_coverage.py::TestMakeRelative::test_make_relative_success

1ms

3

AI ASSESSMENT

Scenario: Test Make Relative - Success

Why Needed: To test that the `make_relative` function correctly resolves relative paths to file paths.

Key Assertions:

- {'expected': '/subdir/file.py', 'actual': 'subdir/file.py'}

Confidence: 80%

Tokens: 147 input + 70 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/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: test_make_relative_with_none_base

Why Needed: To ensure that the `make_relative` function correctly handles cases where the base path is None.

Key Assertions:

- {'name': 'assert result is equal to expected result', 'expected_result': 'path/to/file.py'}

Confidence: 80%

Tokens: 116 input + 77 output = 193 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	7 lines (ranges: 30, 33, 36, 39, 42, 55-56)

PASSED

tests/test_fs_coverage.py::TestNormalizePath::test_normalize_path_backslashes

1ms

3

AI ASSESSMENT

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

Why Needed: To ensure that backslashes are correctly converted to forward slashes in file paths.

Key Assertions:

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

Confidence: 80%

Tokens: 114 input + 78 output = 192 total

COVERAGE

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

PASSED

tests/test_fs_coverage.py::TestNormalizePath::test_normalize_path_path_object

1ms

3

AI ASSESSMENT

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

Why Needed: To ensure the `normalize_path` function correctly normalizes path objects, including those with nested directories.

Key Assertions:

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

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/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 `pathlib` module correctly handles paths with trailing slashes.

Key Assertions:

- {'assertion': 'The normalized path has no trailing slash.', 'expected_result': 'path/to/dir'}

Confidence: 80%

Tokens: 111 input + 85 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::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: To ensure that regular paths are not skipped and should be included in the coverage report.

Key Assertions:

- {'description': 'Regular path is included in the coverage report', 'expected_result': True}
- {'description': 'Regular path should be excluded from the coverage report', 'expected_result': False}

Confidence: 80%

Tokens: 120 input + 109 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/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 the .git directory because it contains Git hooks that are not necessary for the current scenario.

Key Assertions:

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

Confidence: 80%

Tokens: 102 input + 93 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/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 paths starting with a skip directory name are properly skipped.

Key Assertions:

- {'name': 'should be True', 'description': 'The function should return True for the given path.'}
- {'name': '.venv', 'description': '.venv is a file that starts with a skip directory name.'}

Confidence: 80%

Tokens: 124 input + 116 output = 240 total

COVERAGE

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

PASSED

tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_pycache

1ms

3

AI ASSESSMENT

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

Why Needed: Because the '__pycache__' directory contains a cached Python file.

Key Assertions:

- {'name': 'should skip __pycache__ directory', 'description': 'The test should return True indicating that the path is skipped.'}

Confidence: 80%

Tokens: 116 input + 90 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	10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)

PASSED

tests/test_fs_coverage.py::TestShouldSkipPath::test_should_skip_site_packages

1ms

3

AI ASSESSMENT

Scenario: /usr/lib/python3.12/site-packages/pkg/mod.py

Why Needed: Because it's a site-package directory.

Key Assertions:

- should be skipped

Confidence: 80%

Tokens: 111 input + 45 output = 156 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 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, which should be skipped.

Key Assertions:

- {'path': 'venv/lib/python3.12/site.py', 'expected': True}
- {'path': '.venv/lib/python3.12/site.py', 'expected': True}

Confidence: 80%

Tokens: 130 input + 108 output = 238 total

COVERAGE

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

Key Assertions:

- {'name': 'should be true', 'description': 'The function should return True when the path is excluded.', 'expected_value': 'True'}
- {'name': 'should be false', 'description': 'The function should return False when the path is not excluded.', 'expected_value': 'False'}

Confidence: 80%

Tokens: 132 input + 128 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/fs.py	15 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-112, 116-117, 119-121, 123)

 tests/test_gemini_provider.py

25 tests

PASSED

tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_annotate _loop_daily_limit_hit 3ms 4

AI ASSESSMENT

Scenario: The test verifies that the 'Gemini requests-per-day limit reached' error is thrown when the daily limit for a model is hit.

Why Needed: This test prevents regression where the Gemini provider does not correctly handle daily limits being exceeded.

Key Assertions:

- assert 'Gemini requests-per-day limit reached' in res.error
- assert 'requests per day' in res.error
- assert 'limit exceeded' in res.error
- assert 'daily limit hit' in res.error
- assert 'requests per day exceeded' in res.error
- assert 'limit exceeded daily' in res.error
- assert 'requests per day limit reached' in res.error

Confidence: 80%

Tokens: 367 input + 157 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/llm/base.py	22 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	50 lines (ranges: 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-227, 232-233, 318-320, 340, 343, 471-473)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_annotation_exceptions_coverage 3ms 4

AI ASSESSMENT

Scenario: Test that 'GenerationFailure' exception is raised when Gemini generation fails

Why Needed: Prevents regression where 'GenerationFailure' exception is not raised due to internal model exhaustion.

Key Assertions:

- MockGenFailure (Line 293)
- MockResExhausted (Lines 300, 301)
- _GeminiRateLimitExceeded (Lines 300, 301)
- assert 'Gemini requests-per-day' in res.error or 'rate limits reached' in res.error

Confidence: 80%

Tokens: 730 input + 119 output = 849 total

COVERAGE

src/pytest_llm_report/collector.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

161ms



AI ASSESSMENT

Scenario: Prevents bug in coverage gaps test by ensuring proper rate limiting and annotation logic for various scenarios.

Why Needed: This test prevents regression that could cause coverage gaps due to improper rate limiting or annotation logic.

Key Assertions:

- 1. Mock imports are set up correctly without causing errors.
- 2. Context too long error is raised with the correct error message.
- 3. RPD (Rate Limiting and Parsing) function returns the expected value for requests_per_day.
- 4. Fallback models are properly fetched and added to the list of available models.
- 5. Input limits logic works correctly, providing the expected input token limit.

Confidence: 80%

Tokens: 821 input + 149 output = 970 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	27 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-331)
src/pytest_llm_report/llm/gemini.py	173 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181-182, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246, 249-250, 252, 254-255, 259, 340, 343, 346, 348-356, 358-361, 363-364, 366-367, 435, 437-439, 441-442, 449-455, 457, 459, 461-466, 471-473, 476-478, 497-498, 502-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-564, 574)

src/pytest_llm_report/llm/schemas.py

7 lines (ranges: 38, 42-43, 50-52, 55)

src/pytest_llm_report/plugin.py

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

PASSED

tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_parse_preferred_models_coverage

1ms



4

AI ASSESSMENT

Scenario: Test parsing preferred models coverage

Why Needed: To test the behavior of GeminiProvider when no model is specified in the configuration.

Key Assertions:

- {'name': 'Expected _parse_preferred_models to return an empty list', 'expected_value': [], 'message': 'The function should not parse any preferred models.'}
- {'name': 'Expected mock_config.model to be None after setting it to "ALL"', 'expected_value': 'None', 'message': 'mock_config.model should be set to None'}

Confidence: 80%

Tokens: 156 input + 131 output = 287 total

COVERAGE

src/pytest_llm_report/collector.py

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

src/pytest_llm_report/llm/base.py

2 lines (ranges: 65-66)

src/pytest_llm_report/llm/gemini.py

13 lines (ranges: 134-135, 137-141, 143-144, 524-527)

src/pytest_llm_report/plugin.py

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

PASSED

tests/test_gemini_provider.py::TestGeminiCoverageGaps::test_prune_d
aily_requests

1ms

3

AI ASSESSMENT

Scenario: test_prune_daily_requests

Why Needed: To test the behavior of the GeminiRateLimiter when a daily request is pruned after being older than 24 hours.

Key Assertions:

- {'assertion': 'The length of _daily_requests should be 0 after pruning', 'expected_result': 0}

Confidence: 80%

Tokens: 157 input + 85 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/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_availableFallback 2ms 3

AI ASSESSMENT

Scenario: Test verifies that the test_tpm_availableFallback function behaves correctly when a token is requested.

Why Needed: This test prevents regression where the function does not wait for the TPM to become available after requesting tokens.

Key Assertions:

- The loop should finish without returning if `remaining` decreases and `request_tokens <= limit`.
- If `request_tokens` is massive, the function should return 0.0 at line 106/108.
- The `tokens_used` value should be equal to `limit` when all tokens are used up.
- The `remaining` value should decrease by `request_tokens` for each iteration of the loop.
- If `remaining + request_tokens <= limit`, the function should not return and wait for the TPM to become available.
- The `time.monotonic()` difference between the start and end times of the loop should be less than or equal to 30 seconds.
- The total number of tokens used by the function should be equal to `limit` when all tokens are used up.

Confidence: 80%

Tokens: 524 input + 233 output = 757 total

COVERAGE

src/pytest_llm_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 annotation when google-generativeai is not installed.

Why Needed: This test prevents a potential import error that could occur when trying to annotate an annotation with a module that does not exist (i.e., `google.generativeai` is not installed).

Key Assertions:

- The annotation contains the string 'google-generativeai not installed' in its error message.
- The annotation includes the key-value pair 'module': 'google.generativeai'.
- The annotation includes the key-value pair 'outcome': 'passed'.
- The annotation includes the nodeid 't'.

Confidence: 80%

Tokens: 259 input + 139 output = 398 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	14 lines (ranges: 134-135, 137-141, 143-144, 164-165, 167-169)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiProvider::test_annotate_no_token 3ms 5

AI ASSESSMENT

Scenario: Test that annotation fails when token is missing and GEMINI_API_TOKEN is not set.**Why Needed:** Prevents a potential bug where the test annotates an empty result without checking for the presence of a GEMINI_API_TOKEN environment variable.**Key Assertions:**

- The error message should contain 'GEMINI_API_TOKEN is not set'.
- The annotation should fail when GEMINI_API_TOKEN is missing.
- The annotation should report an error indicating that GEMINI_API_TOKEN is not set.
- The test should verify the presence of a GEMINI_API_TOKEN environment variable before annotating the result.
- The test should check for the absence of a GEMINI_API_TOKEN environment variable in the annotation string.
- The test should report an error when trying to annotate without setting GEMINI_API_TOKEN.
- The test should verify that the annotation is not successful with missing GEMINI_API_TOKEN.

Confidence: 80%**Tokens:** 313 input + 207 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/llm/base.py	22 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	21 lines (ranges: 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-188)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiProvider::test_annotate_rate_limit_retry 5ms 4

AI ASSESSMENT

Scenario: Test that the GeminiProvider correctly annotates a rate limit retry scenario.

Why Needed: This test prevents regression in the case where the GeminiProvider is called with a 429 status code due to rate limiting, and then receives a successful response after a retry.

Key Assertions:

- The annotation's `scenario` field matches the expected value 'Recovered Scenario'.
- The mock `mock_post.call_count` is set to 2 when the test call fails with a 429 status code, indicating that the provider retries the request.
- The mock `mock_get.return_value.json.return_value` is set to a valid JSON response for a successful rate limit retry scenario.
- The `test_result` nodeid matches the expected value 'test1'.
- The annotation's `error` field is None when the test call succeeds with a 200 status code, indicating that no error occurred.
- The mock `_parse_response` returns a valid response for a successful rate limit retry scenario.
- The `mock_get.return_value.status_code` matches the expected value '429' when the test call fails with a 429 status code.
- The `mock_get.return_value.headers` set to `{'

Confidence: 80%

Tokens: 636 input + 264 output = 900 total

COVERAGE

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

412ms



4

AI ASSESSMENT

Scenario: Test that the _annotate_internal method returns a successful annotation with no error when the response from _call_gemini is in the expected format.

Why Needed: This test prevents regression where the GeminiProvider does not correctly handle responses from _call_gemini with the expected format.

Key Assertions:

- The scenario 'Success Scenario' should be returned by _annotate_internal.
- _parse_response returns a Mock object with the correct scenario and no error.
- The annotation does not contain any error information.
- The annotation's scenario matches the expected value.
- _build_prompt is called before _parse_response to avoid complex dependencies, but this test does not require it.
- The response from _call_gemini has the correct format (text and tokens).
- The annotation correctly handles a successful response from _call_gemini with no error.

Confidence: 80%

Tokens: 649 input + 188 output = 837 total

COVERAGE

src/pytest_llm_report/collector.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: Verifies that the availability of a Gemini provider is checked correctly when environment variables are set.

Why Needed: To prevent a potential bug where the availability of a Gemini provider is not checked in certain environments, such as when environment variables like GEMINI_API_TOKEN are set.

Key Assertions:

- The function _check_availability() returns False when environment variable GEMINI_API_TOKEN is set.
- The function _check_availability() returns True when environment variable GEMINI_API_TOKEN is not set.

Confidence: 80%**Tokens:** 235 input + 119 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/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 for models with high request rates.

Why Needed: This test prevents a regression where the GeminiProvider class fails to annotate retry exceptions for models with high request rates, potentially leading to incorrect model exhaustion tracking.

Key Assertions:

- The 'm1' model should be marked as exhausted after a daily limit is exceeded.
- The 'm1' model should be marked as available again after a retry-after cleanup is triggered.
- The cooldown for the 'm1' model should exceed the time it took to reach the daily limit.
- The cooldown for the 'm1' model should not be reset immediately after a retry-after cleanup is triggered.
- Mock calls to '_ensure_models_and_limits' and '_call_gemini' should result in the correct side effects being simulated.

Confidence: 80%

Tokens: 651 input + 187 output = 838 total

COVERAGE

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

PASSED

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_annotate_retry_loop_coverage

3ms



5

AI ASSESSMENT

Scenario: Test that the GeminiProvider correctly clears the `'_model_exhausted_at` dictionary after a successful annotation.

Why Needed: This test prevents regression where the `GeminiProvider` fails to clear the `'_model_exhausted_at` dictionary after a successful annotation, potentially leading to inconsistent model exhaustion checks.

Key Assertions:

- The value of `'_model_exhausted_at[model]` is set to None before the annotation process starts.
- After calling `provider._annotate_internal`, the value of `'_model_exhausted_at[model]` becomes False.
- If a successful annotation occurs, then the value of `'_model_exhausted_at[model]` should be cleared to avoid inconsistent model exhaustion checks.

Confidence: 80%

Tokens: 482 input + 160 output = 642 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	27 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-331)
src/pytest_llm_report/llm/gemini.py	97 lines (ranges: 39-42, 45-46, 48, 52-54, 66, 68-70, 73, 76-78, 81-82, 84, 87-88, 92-94, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-210, 212-213, 215-216, 218, 222-224, 228-230, 232, 235-236, 239-244, 246-247, 249-252, 254, 259, 340, 343, 471-473)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-52, 55)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_ensure_rate_limits_error

1ms



4

AI ASSESSMENT

Scenario: TestGeminiProviderDetailed::test_ensure_rate_limits_error**Why Needed:** This test ensures that the GeminiProvider raises an error when rate limiting is attempted with a non-integer value.**Key Assertions:**

- {'name': 'Rate limits should be enforced', 'description': 'The function should return a valid rate limit configuration.', 'expected_value': 10, 'actual_value': 10}
- {'name': 'Exception is raised when rate limiting is attempted with non-integer value', 'description': 'The function should raise an exception when rate limiting is attempted with a non-integer value.', 'expected_exception': 'RateLimitError'}

Confidence: 80%**Tokens:** 156 input + 160 output = 316 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	27 lines (ranges: 134-135, 137-141, 143-144, 346, 348-356, 358-361, 363-364, 366-367)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_fetch_available_models_error

1ms



5

AI ASSESSMENT

Scenario: TestGeminiProviderDetailed::test_fetch_available_models_error**Why Needed:** To test that an exception is raised when fetching available models fails due to a network error.**Key Assertions:**

- {'name': 'models are empty', 'expected_value': [], 'actual_value': '[]'}
- {'name': 'limit_map is empty', 'expected_value': {}, 'actual_value': {}}

Confidence: 80%**Tokens:** 132 input + 106 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	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/gemini.py	15 lines (ranges: 134-135, 137-141, 143-144, 537, 539-541, 544-545)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_fetch_available_models_invalid_json

2ms



AI ASSESSMENT

Scenario: Test that fetching available models with invalid JSON data prevents a bug.

Why Needed: To prevent a potential issue where the GeminiProvider fetches models from an invalid JSON response, which could lead to incorrect model availability information.

Key Assertions:

- The 'm1' model should not be included in the list of available models.
- The 'm2' model should not be included in the list of available models.
- The 'm3' model should be included in the list of available models.
- The 'inputTokenLimit' value for the 'm3' model should match its supported generation methods.
- The 'limit_map' dictionary should contain only valid keys ('m1', 'm2', and 'm3').

Confidence: 80%

Tokens: 340 input + 169 output = 509 total

COVERAGE

src/pytest_llm_report/collector.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: TestGeminiProviderDetailed

Why Needed: To ensure that the `get_max_context_tokens` method calls the `mock_ensure` function.

Key Assertions:

- {'name': 'Mock Ensure Function', 'expected_calls': [1], 'message': 'The `mock_ensure` function was called once.'}

Confidence: 80%

Tokens: 144 input + 86 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	3 lines (ranges: 65-66, 163)
src/pytest_llm_report/llm/gemini.py	15 lines (ranges: 134-135, 137-141, 143-144, 486, 488-491, 493)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_parse_rate_limits_types

1ms



AI ASSESSMENT

Scenario:

tests/test_gemini_provider.py::TestGeminiProviderDetailed::test_parse_rate_limits_types

Why Needed: To ensure that the Gemini provider can correctly parse rate limits and return the expected configuration.

Key Assertions:

- {'name': 'requests_per_minute', 'expected_value': 'None'}
- {'name': 'tokens_per_minute', 'expected_value': 100}

Confidence: 80%

Tokens: 156 input + 101 output = 257 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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: Test the prune logic of the GeminiRateLimiter to ensure it removes old requests within a specified time window.

Why Needed: This test prevents a potential issue where old requests are not removed from the limiter's cache, leading to performance degradation or incorrect rate limiting behavior.

Key Assertions:

- The length of _request_times should be 1 after pruning.
- The length of _token_usage should be 1 after pruning.
- The first element in _request_times should be equal to the current time minus 10 seconds.
- _request_times[0] should be equal to now - 10.0
- The first element in _token_usage should be equal to (now - 10.0, 10).

Confidence: 80%

Tokens: 323 input + 168 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	11 lines (ranges: 39-42, 81-85, 87-88)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiRateLimiter::test_record_to_kens_invalid

1ms

3

AI ASSESSMENT

Scenario: Test that record tokens with a negative value returns an empty list of token usage.

Why Needed: To ensure the rate limiter correctly handles invalid input values, specifically negative token counts.

Key Assertions:

- {'message': 'Expected len(limiter._token_usage) to be 0'}
- {'message': 'Actual: {len(limiter._token_usage)}', 'value': 1}

Confidence: 80%

Tokens: 127 input + 96 output = 223 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	6 lines (ranges: 39-42, 66-67)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

AI ASSESSMENT

Scenario: Test the rate limiting mechanism

Why Needed: The test ensures that the rate limiting mechanism correctly limits the number of requests to a resource within a certain time frame.

Key Assertions:

- {'assertion': 'limiter.next_available_in(100) is None', 'description': 'The next available in time should be None after 100 requests'}

Confidence: 80%

Tokens: 129 input + 91 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/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 resets after the third request.

Why Needed: This test prevents a potential bug where the rate limiter does not reset properly after the third request, leading to unexpected behavior in subsequent requests.

Key Assertions:

- The next_available_in method returns 0.0 for the first two requests and 60.0 for the third request.
- The limit is correctly reset after the third request.
- The rate limiter does not wait for more than 1 minute between requests.

Confidence: 80%

Tokens: 280 input + 130 output = 410 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	27 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-97, 100-102)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiRateLimiter::test_seconds_until_tpm_available_branches

1ms



3

AI ASSESSMENT

Scenario: Verify that the rate limiter correctly returns seconds until TPM available when no tokens are requested or more than limit is exceeded.

Why Needed: This test prevents a regression where the rate limiter does not return a correct value for seconds_until_tpm_available when no tokens are requested, and also ensures that it returns a correct value even when more than the limit of tokens is requested but usage is still within the limit.

Key Assertions:

- assert limiter._seconds_until_tpm_available(now, 0) == 0.0
- assert limiter._seconds_until_tpm_available(now, 150) == 0.0
- assert wait > 0 and wait <= 60.0 + 1e-9

Confidence: 80%

Tokens: 377 input + 168 output = 545 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/gemini.py	18 lines (ranges: 39-42, 100-101, 103-114)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_gemini_provider.py::TestGeminiRateLimiter::test_wait_for_slot_daily_limit_exceeded 1ms 3

AI ASSESSMENT

Scenario: Verify that the `wait_for_slot` method raises a `GeminiRateLimitExceeded` exception when the daily limit is exceeded.

Why Needed: Prevents a potential rate limiter error where the daily limit is exceeded and the user does not receive an error message.

Key Assertions:

- The `limit_type` attribute of the raised exception is set to 'requests_per_day'.
- The `value` attribute of the raised exception contains the string 'requests_per_day'.
- The `exc.value.limit_type` attribute matches the expected value 'requests_per_day'.
- The `exc.value.limit_type` attribute matches the expected value 'requests_per_day'.
- The `exc.value.limit_type` attribute is set to 'requests_per_day' and its value is a string.
- The `exc.value.limit_type` attribute is set to 'requests_per_day' and its value is a string.
- The `exc.value.limit_type` attribute matches the expected value 'requests_per_day'.
- The `exc.value.limit_type` attribute matches the expected value 'requests_per_day'.

Confidence: 80%

Tokens: 263 input + 245 output = 508 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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 waits for a sufficient amount of time before returning.

Why Needed: This test prevents a potential regression where the rate limiter does not wait long enough for subsequent requests to be processed.

Key Assertions:

- The next_available_in method is called with an argument greater than or equal to 10.0 seconds.
- The mock_sleep function is called once with an argument of 10.0 seconds.
- The time.monotonic value returned by the mock_next_available_in function is greater than or equal to 10.0 seconds.

Confidence: 80%

Tokens: 325 input + 133 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/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 same provider produce different hashes.

Key Assertions:

- {'name': 'config1 and config2 should have different hashes', 'expected_value': 'different hashes'}

Confidence: 80%

Tokens: 119 input + 79 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	8 lines (ranges: 96-101, 103-104)

PASSED

tests/test_hashing.py::TestComputeConfigHash::test_returns_short_hash

1ms

4

AI ASSESSMENT

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

Why Needed: To ensure the computed hash is short and can be easily stored in a database or used for other purposes where storage space is limited.

Key Assertions:

- {'name': 'hash length', 'expected': 16, 'actual': {'type': 'int', 'value': 16}}

Confidence: 80%

Tokens: 109 input + 100 output = 209 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	8 lines (ranges: 96-101, 103-104)

PASSED

tests/test_hashing.py::TestComputeFileSha256::test_consistent_with_bytes

1ms



AI ASSESSMENT

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

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

Key Assertions:

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

Confidence: 80%

Tokens: 144 input + 78 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	6 lines (ranges: 32, 44-48)

PASSED

tests/test_hashing.py::TestComputeFileSha256::test_hashes_file

1ms



AI ASSESSMENT

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

Why Needed: To ensure that the `compute_file_sha256` function correctly hashes file contents.

Key Assertions:

- {'name': 'Expected length of hash to be 64 bytes', 'expected': 64, 'actual': 0}

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/hashing.py	5 lines (ranges: 44-48)

PASSED

tests/test_hashing.py::TestComputeHmac::test_different_key

1ms  3

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestComputeHmac::test_different_key

Why Needed: To ensure that different keys produce different signatures.

Key Assertions:

- {'message': 'Different keys should produce different signatures.', 'description': 'The HMAC signature for a given input content and key should be unique.'}

Confidence: 80%

Tokens: 125 input + 78 output = 203 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	1 lines (ranges: 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 test the computation of HMAC with a key.

Key Assertions:

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

Confidence: 80%

Tokens: 108 input + 68 output = 176 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	1 lines (ranges: 61)

PASSED

tests/test_hashing.py::TestComputeSha256::test_consistent

1ms  3

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestComputeSha256::test_consistent

Why Needed: To ensure that the hash function is consistent and produces the same output for the same input.

Key Assertions:

- {'expected': {'h1': 'hash1', 'h2': 'hash2'}, 'actual': {'h1': 'hash1', 'h2': 'hash1'}}

Confidence: 80%

Tokens: 115 input + 100 output = 215 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/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: The length of the hash is expected to be 64 characters.

Key Assertions:

- {'message': 'Expected hash length to be 64. Actual: %r'}
- expected_value
- actual_value

Confidence: 80%

Tokens: 103 input + 82 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/hashing.py	1 lines (ranges: 32)

PASSED

tests/test_hashing.py::TestGetDependencySnapshot::test_includes_pytest 77ms 3

AI ASSESSMENT

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

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

Key Assertions:

- {'name': 'includes pytest package', 'description': "The 'pytest' package should be present in the dependency snapshot."}

Confidence: 80%

Tokens: 102 input + 86 output = 188 total

COVERAGE

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

PASSED

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

AI ASSESSMENT

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

Why Needed: To ensure that the `get_dependency_snapshot` function returns a dictionary.

Key Assertions:

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

Confidence: 80%

Tokens: 98 input + 80 output = 178 total

COVERAGE

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

PASSED

tests/test_hashing.py::TestLoadHmacKey::test_loads_key

1ms  4

AI ASSESSMENT

Scenario: tests/test_hashing.py::TestLoadHmacKey::test_loads_key**Why Needed:** To test the functionality of loading a HMAC key from a file.**Key Assertions:**

- {'expected_value': "b'my-secret-key\\n'", 'actual_value': "b'...!'"}

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

PASSED

tests/test_hashing.py::TestLoadHmacKey::test_missing_key_file

1ms



AI ASSESSMENT

Scenario: tests/test_hashing.py::TestLoadHmacKey::test_missing_key_file**Why Needed:** The test should return None if the key file does not exist.**Key Assertions:**

- {'message': 'Expected to return None when missing key file exists'}

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

- {'name': 'Expected result', 'type': 'NoneType', 'message': 'The function should return None.'}

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/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/hashing.py	2 lines (ranges: 73-74)

 tests/test_integration_gate.py

16 tests

PASSED

tests/test_integration_gate.py::TestConfigDefaults::test_aggregation_defaults

1ms

3

AI ASSESSMENT

Scenario: Test aggregation defaults with no aggregate directory specified.

Why Needed: Without an aggregate directory, aggregation may not work as expected or may result in unexpected behavior.

Key Assertions:

- config.aggregate_dir should be None
- config.aggregate_policy should be 'latest'
- config.aggregate_include_history should be False

Confidence: 80%

Tokens: 201 input + 79 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/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: The LLM (Language Model) is not enabled by default.

Key Assertions:

- {'name': 'LLM is not enabled by default', 'description': 'The LLM should be disabled by default.'}

Confidence: 80%

Tokens: 109 input + 85 output = 194 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 omits all tests from the coverage report by default.

Key Assertions:

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

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/options.py	3 lines (ranges: 123, 171, 293)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_integration_gate.py::TestConfigDefaults::test_provider_default_none

1ms

3

AI ASSESSMENT

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

Why Needed: To ensure that the provider is set to 'none' by default when no other value is specified.

Key Assertions:

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

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/options.py	3 lines (ranges: 123, 171, 293)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_integration_gate.py::TestConfigDefaults::test_secret_exclude_globs

1ms



AI ASSESSMENT

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

Why Needed: This test is needed because the default configuration includes secret files and environment variables, which are not intended to be exposed.

Key Assertions:

- {'name': 'Excluding secret files', 'expected': ['llm_context_exclude_globs'], 'actual': ['secret']}
- {'name': 'Excluding .env files', 'expected': ['llm_context_exclude_globs'], 'actual': []}

Confidence: 80%

Tokens: 132 input + 127 output = 259 total

COVERAGE

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

6ms



5

AI ASSESSMENT

Scenario: Tests the deterministic output of the integration gate.**Why Needed:** This test prevents regression that could cause non-deterministic output.**Key Assertions:**

- The `nodeid` values in the report are sorted by nodeid.
- All `nodeid` values are present in the report.
- No duplicates are present in the `nodeid` values.
- All `outcome` values are either 'passed' or 'failed'.
- The `tests` list contains exactly one test for each `nodeid`.
- Each `test_z`, `test_a`, and `test_m` is a valid test function.
- The output of the integration gate is deterministic (sorted by nodeid).

Confidence: 80%**Tokens:** 313 input + 164 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/models.py	80 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

src/pytest_llm_report/report_writer.py

122 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 347, 350-352, 355-356, 359-361, 364, 367-371, 477-478, 502, 504, 506-508, 510, 513)

PASSED

tests/test_integration_gate.py::TestFullPipeline::test_empty_test_suite 5ms 6

AI ASSESSMENT

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

Why Needed: This test prevents regression where the test suite is empty, ensuring the report is always valid and accurate.

Key Assertions:

- The total count of tests in the report should be zero.
- The summary section of the report should contain no data.
- The 'total' key in the summary section should have a value of zero.
- The test counts are not affected by an empty test suite.
- The report does not contain any duplicate test names or IDs.
- There are no missing test results in the report.
- The test suite is correctly identified as empty in the report.

Confidence: 80%

Tokens: 240 input + 153 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/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 37ms 6

AI ASSESSMENT

Scenario: The test verifies that the full pipeline generates an HTML report.

Why Needed: This test prevents a regression where the HTML report is not generated correctly due to a change in the configuration.

Key Assertions:

- The file "report.html" exists at the specified path.
- The string '
- The string 'test_pass' is present in the content of 'report.html'.

Confidence: 80%

Tokens: 270 input + 107 output = 377 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	118 lines (ranges: 55, 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-325, 337, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

PASSED

tests/test_integration_gate.py::TestFullPipeline::test_json_report_generation 62ms 7

AI ASSESSMENT

Scenario: Test that the full pipeline generates a valid JSON report.**Why Needed:** This test prevents regression in the pipeline where JSON reports are not generated correctly.**Key Assertions:**

- The `report_json` and `report_html` paths should exist in the temporary directory.
- The schema version of the report data should match the expected value.
- The summary statistics (total, passed, failed, skipped) should be correct.

Confidence: 80%**Tokens:** 419 input + 102 output = 521 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,

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: This test prevents a bug where the report root is missing required fields, making it invalid to use as a schema.

Key Assertions:

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

Confidence: 80%

Tokens: 250 input + 111 output = 361 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	54 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_integration_gate.py::TestSchemaCompatibility::test_run_meta_has_aggregation_fields

1ms

3

AI ASSESSMENT

Scenario:

tests/test_integration_gate.py::TestSchemaCompatibility::test_run_meta_has_aggregation_fields

Why Needed: To ensure that the RunMeta object has an 'is_aggregated' key and a 'run_count' value when it is used in aggregation.

Key Assertions:

- {'name': 'is_aggregated', 'expected_value': 'True'}
- {'name': 'run_count', 'expected_value': 0}

Confidence: 80%

Tokens: 136 input + 111 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	29 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_integration_gate.py::TestSchemaCompatibility::test_run_meta_has_status_fields

1ms



AI ASSESSMENT

Scenario: Test 'RunMeta has run status fields' verifies that the RunMeta object contains the required status fields.

Why Needed: This test prevents a potential bug where the RunMeta object is missing certain status fields, potentially leading to incorrect analysis results.

Key Assertions:

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

Confidence: 80%

Tokens: 237 input + 144 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/models.py	29 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_integration_gate.py::TestSchemaCompatibility::test_schema_version_defined

1ms

2

AI ASSESSMENT

Scenario:

tests/test_integration_gate.py::TestSchemaCompatibility::test_schema_version_defined

Why Needed: The schema version is needed to ensure compatibility with the gate.

Key Assertions:

- {'name': 'SCHEMA_VERSION', 'value': '1.2.3'}

Confidence: 80%

Tokens: 103 input + 75 output = 178 total

COVERAGE

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

PASSED

tests/test_integration_gate.py::TestSchemaCompatibility::test_test_case_has_required_fields

1ms



AI ASSESSMENT

Scenario: Test verifies that the `TestCaseResult` object has required fields.

Why Needed: This test prevents a potential bug where the `TestCaseResult` object is missing some of its required fields, potentially causing incorrect results or errors.

Key Assertions:

- nodeid
- outcome
- duration

Confidence: 80%

Tokens: 223 input + 77 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_litellm_retry_coverage.py

4 tests

PASSED

tests/test_litellm_retry_coverage.py::TestLiteLLMTokenRefreshRetry::test_all_retries_exhausted

2.00s



AI ASSESSMENT

Scenario: Test verifies that all retries are exhausted when API call fails.

Why Needed: Prevents regression where LLMTokenRefreshRetry may not retry if all retries are exhausted.

Key Assertions:

- The `provider.annotate` method should raise an exception with the error message.
- The `result.error` attribute should be set to a non-None value indicating an error.
- The `result` object should contain a `message` key with the error message.
- The `error` attribute of the `result.message` string should match the expected error message.
- The `provider.annotate` method should not return any result when all retries are exhausted.
- The `provider.annotate` method should raise an exception when all retries are exhausted and no result is returned.
- The `provider.annotate` method should set the `error` attribute of the `result` object to a non-None value indicating an error.

Confidence: 80%

Tokens: 346 input + 205 output = 551 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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, where the LLMTokenRefreshRetry test would fail due to a forced refresh.**Key Assertions:**

- The annotation should not return an error with status code 500 (Internal server error).
- The annotation should have an error attribute set to the exception object.
- The annotation should not call the completion function with any arguments.
- The annotation should not raise an exception.
- The annotation should be None if the API call fails without raising an exception.
- The annotation's status code should match the expected 500 status code.
- The annotation's error attribute should have a non-None value.

Confidence: 80%**Tokens:** 367 input + 169 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	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.**Why Needed:** To prevent regression in case of transient errors where the API call fails and needs to be retried.**Key Assertions:**

- The test verifies that the retry mechanism works correctly even when the initial API call fails.
- The test ensures that the retry process does not get stuck in an infinite loop due to transient errors.
- The test verifies that the scenario is updated correctly after a transient error occurs.
- The test checks if the error message is properly set and matches the expected format.
- The test verifies that the key assertions are met, including 'assert true' as required.
- The test ensures that the result object has an 'error' attribute that is None when no error occurred.
- The test verifies that the scenario is updated correctly after a transient error occurs.

Confidence: 80%**Tokens:** 433 input + 187 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	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)

PASSED

tests/test_litellm_retry_coverage.py::TestLiteLLMTokenRefreshRetry::test_token_refresh_on_401

5.96s



AI ASSESSMENT

Scenario: Test that 401 error triggers token refresh (lines 123-126).**Why Needed:** To ensure the LLMTokenRefreshRetry test case covers a scenario where the API call to refresh the token fails with a 401 status code.**Key Assertions:**

- The function `provider.annotate` should be called twice with an error status code of 401.
- The second call to `provider.annotate` should have a valid response containing a new token.
- The third check in the test case should not trigger due to successful API call.
- The LLMTokenRefreshRetry test case should fail when the API call to refresh the token fails with a 401 status code.

Confidence: 80%**Tokens:** 473 input + 157 output = 630 total

COVERAGE

src/pytest_llm_report/collector.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)

PASSED

tests/test_llm.py::TestGetProvider::test_gemini_returns_provider

1ms



5

AI ASSESSMENT

Scenario: tests/test_llm.py::TestGetProvider::test_gemini_returns_provider**Why Needed:** The test is necessary to ensure that the GeminiProvider class is correctly instantiated when the 'gemini' provider is used.**Key Assertions:**

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

Confidence: 80%**Tokens:** 131 input + 89 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/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_value': 'LiteLLMProvider'}

Confidence: 80%**Tokens:** 140 input + 91 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	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: The test is necessary because the LLM returns a `NoopProvider` when the provider is set to 'none'. This is not a realistic scenario, as it would indicate that the LLM has no knowledge of the input data.

Key Assertions:

- {'name': 'provider', 'expected': 'None', 'got': 'NoopProvider'}

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/llm/base.py	6 lines (ranges: 65-66, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

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

AI ASSESSMENT

Scenario: tests/test_llm.py::TestGetProvider::test_ollama_returns_provider

Why Needed: To ensure that the OllamaProvider is correctly instantiated when a specific provider ('ollama') is used.

Key Assertions:

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

Confidence: 80%

Tokens: 154 input + 89 output = 243 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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 the NoopProvider class implements all required interface methods.**Why Needed:** This test prevents a potential bug where the NoopProvider class is not implemented correctly and may cause unexpected behavior or errors.**Key Assertions:**

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

Confidence: 80%**Tokens:** 232 input + 119 output = 351 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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: Test that NoopProvider returns an empty annotation when no nodes are annotated.

Why Needed: This test prevents a regression where the annotate method of NoopProvider returns an annotation even if no nodes are annotated.

Key Assertions:

- annotation is an instance of LlmAnnotation
- annotation scenario is an empty string
- annotation why_needed is an empty string
- annotation key_assertions is an empty list

Confidence: 80%

Tokens: 249 input + 102 output = 351 total

COVERAGE

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

- {'name': 'assert provider.get_model_name() == "", 'expected_value': '', 'message': 'Expected get_model_name to return an empty string'}

Confidence: 80%**Tokens:** 114 input + 97 output = 211 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 67)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm.py::TestNoopProvider::test_is_available

1ms  5

AI ASSESSMENT

Scenario: tests/test_llm.py::TestNoopProvider::test_is_available**Why Needed:** The NoopProvider class should always be available.**Key Assertions:**

- {'name': 'provider.is_available() should return True', 'expected_result': True}

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/llm/base.py	5 lines (ranges: 65-66, 134, 137-138)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 59)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

 **tests/test_llm_contract.py**

13 tests

PASSED

tests/test_llm_contract.py::TestAnnotationSchema::test_required_fields

1ms



AI ASSESSMENT

Scenario: Test the schema requirements

Why Needed: This test ensures that the LLM contract's annotation schema requires 'scenario' and 'why_needed' fields.

Key Assertions:

- {'field_name': 'required', 'expected_value': ['scenario', 'why_needed'], 'actual_value': [0, 1]}

Confidence: 80%

Tokens: 115 input + 85 output = 200 total

COVERAGE

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

PASSED

tests/test_llm_contract.py::TestAnnotationSchema::test_schema_from_dict

1ms



3

AI ASSESSMENT

Scenario: The test verifies that the `AnnotationSchema.from_dict` method correctly parses a dictionary into an AnnotationSchema instance.

Why Needed: This test prevents potential issues where the `AnnotationSchema.from_dict` method fails to parse a dictionary with required keys, potentially leading to incorrect schema creation or other errors.

Key Assertions:

- checks password
- checks username
- correctly parses the 'scenario' key
- correctly parses the 'why_needed' key
- correctly parses the 'confidence' key
- has exactly two assertions for the 'key_assertions' list

Confidence: 80%

Tokens: 274 input + 137 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/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: TestAnnotationSchema.test_schema_handles_empty

Why Needed: This test ensures that the AnnotationSchema class correctly handles empty input scenarios.

Key Assertions:

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

Confidence: 80%

Tokens: 109 input + 96 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/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 invalid or unexpected behavior.

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']

Confidence: 80%

Tokens: 215 input + 109 output = 324 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: Test AnnotationSchema::test_schema_to_dict verifies that the `AnnotationSchema` instance is correctly serialized to a dictionary.

Why Needed: This test prevents regression by ensuring that the `AnnotationSchema` instance can be successfully converted to a dictionary, which is necessary for proper data storage and retrieval.

Key Assertions:

- assertion 1
- assertion 2
- # Ensure 'key_assertions' key exists in the dictionary

Confidence: 80%**Tokens:** 247 input + 105 output = 352 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 90-92, 94-96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

AI ASSESSMENT

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

Why Needed: The test is necessary to ensure that the NoopProvider class can be instantiated correctly when a provider is set to 'none'.

Key Assertions:

- {'name': 'provider', 'expected_type': 'NoopProvider', 'actual_type': 'get_provider(config)', 'message': 'Expected get_provider(config) to return NoopProvider, but got '}

Confidence: 80%

Tokens: 118 input + 117 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	6 lines (ranges: 65-66, 384, 386, 388-389)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_contract.py::TestNoopProvider::test_noop_is_llm_provider 1ms 5

AI ASSESSMENT

Scenario: tests/test_llm_contract.py::TestNoopProvider::test_noop_is_llm_provider**Why Needed:** The test is necessary because the `isinstance` function checks if an object is of a certain type. In this case, we want to ensure that the `provider` variable is actually an instance of `LLMProvider`, which is the expected implementation.**Key Assertions:**

- {'name': 'The provider is an instance of LLMProvider', 'description': 'We expect the `provider` variable to be an instance of `LLMProvider`. This can be verified by checking if it has a certain attribute or method.'}

Confidence: 80%**Tokens:** 117 input + 150 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/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 test verifies that NoopProvider returns an empty annotation when the test function does not return any value.

Why Needed: This test prevents a regression where the LLM contract incorrectly returns an annotation for a non-executable test function.

Key Assertions:

- The annotation returned by NoopProvider is expected to be empty.
- No assertion was made in the provided code, indicating that it should return an empty annotation.
- The `annotate` method of the `NoopProvider` class is expected to return a `TestCaseResult` object with an empty string as its scenario.

Confidence: 80%

Tokens: 253 input + 134 output = 387 total

COVERAGE

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

PASSED

tests/test_llm_contract.py::TestProviderContract::test_annotate_returns_annotation

1ms



5

AI ASSESSMENT

Scenario: Test that the `annotate` method returns a valid LlmAnnotation-like object with the correct attributes.

Why Needed: This test prevents regression where the `annotate` method does not return an expected LlmAnnotation-like object.

Key Assertions:

- The result has the expected 'scenario' attribute.
- The result has the expected 'why_needed' attribute.
- The result has the expected 'key_assertions' attribute.

Confidence: 80%**Tokens:** 263 input + 103 output = 366 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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 contract handles empty code gracefully and returns a valid result.

Key Assertions:

- {'description': 'Expected the provider to return a non-empty result for an empty test code.', 'value': 'True'}

Confidence: 80%

Tokens: 145 input + 85 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	8 lines (ranges: 65-66, 87-89, 97-98, 105)
src/pytest_llm_report/llm/noop.py	2 lines (ranges: 32, 51)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_contract.py::TestProviderContract::test_provider_handles_none_context

1ms



AI ASSESSMENT

Scenario: Provider handles None context gracefully**Why Needed:** To ensure the provider can handle cases where `None` is passed as a context.**Key Assertions:**

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

Confidence: 80%**Tokens:** 148 input + 69 output = 217 total

COVERAGE

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

- {'message': 'All providers should have an annotate method.', 'expected_result': 'True'}
- {'message': 'Each provider should have a callable annotate method.', 'expected_result': 'True'}

Confidence: 80%

Tokens: 145 input + 102 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/base.py	15 lines (ranges: 65-66, 384, 386, 388-389, 391-392, 394, 396-397, 399, 401-402, 404)
src/pytest_llm_report/llm/gemini.py	9 lines (ranges: 134-135, 137-141, 143-144)
src/pytest_llm_report/llm/litellm_provider.py	3 lines (ranges: 37-38, 41)
src/pytest_llm_report/llm/noop.py	1 lines (ranges: 32)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

 tests/test_llm_providers.py

52 tests

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_handles_context_too_large

1ms



5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py**Why Needed:** The current implementation of annotate_handles_context may lead to a MemoryError when dealing with large contexts.**Key Assertions:**

- {'name': 'annotate_handles_context', 'description': 'This function is expected to handle the context correctly and not raise an exception.'}
- {'name': 'context_size', 'description': 'The size of the context should be within a reasonable limit (e.g., 1024 bytes).}'}

Confidence: 80%**Tokens:** 98 input + 121 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/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	187 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 263-265, 299, 311-312, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 435, 437-439, 441-444, 449-452, 463-466, 471-473, 476-478, 497-498, 502-505, 507-508, 511, 514-516, 518-521, 524-525, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_missing_dependency

1ms



5

AI ASSESSMENT

Scenario: Test that the LiteLLMProvider annotates a missing dependency correctly.

Why Needed: This test prevents the provider from reporting an error when a required library is not installed.

Key Assertions:

- The annotation message includes the correct error message for installing the required library.
- The annotation message does not include any misleading information about the installation process.
- The annotation message provides a clear and concise explanation of what needs to be done to resolve the issue.
- The test passes with an empty annotation object if no errors are found during the annotation process.
- The test fails with a non-empty annotation object if an error is found during the annotation process.
- The annotation object contains the correct information about the required library and its installation instructions.
- The annotation object does not contain any unnecessary or misleading information.

Confidence: 80%

Tokens: 270 input + 181 output = 451 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	34 lines (ranges: 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-195, 471-473, 497-498, 502-503, 537)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_missing_token

1ms



5

AI ASSESSMENT

Scenario: Test that the `annotate` method of the `GeminiProvider` class throws an error when an API token is missing.

Why Needed: This test prevents a potential bug where the `annotate` method fails to recognize the absence of an API token, potentially leading to incorrect results or errors.

Key Assertions:

- The annotation does not throw an error when the `GEMINI_API_TOKEN` environment variable is not set.
- The annotation throws an error with a message indicating that the `GEMINI_API_TOKEN` environment variable is not set.
- The annotation correctly identifies the absence of an API token and returns it as the error message.
- The annotation does not throw any errors when the `GEMINI_API_TOKEN` environment variable is explicitly unset using `unsetenv` or other methods.
- The annotation throws a `ResourceExhausted` exception with a mock `GenerationFailure` instance, which is consistent with the expected behavior of the `annotate` method.
- The annotation correctly handles cases where the `GEMINI_API_TOKEN` environment variable is not set and returns an appropriate error message.

Confidence: 80%

Tokens: 440 input + 244 output = 684 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/gemini.py	21 lines (ranges: 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-188)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_records_tokens

1ms



6

AI ASSESSMENT

Scenario: Verify that the `annotate` method records tokens on the limiter and rate limits correctly.

Why Needed: Prevents regressions where token usage is not recorded or reported accurately.

Key Assertions:

- The 'json' key in the captured dictionary contains a valid JSON payload with the correct response data.
- The 'totalTokenCount' key in the captured dictionary matches the expected value of 123.
- The 'candidates' list in the captured dictionary contains at least one record with the correct content and metadata.
- The 'usageMetadata' object in the captured dictionary has a valid total token count.
- No exceptions are raised when calling `fake_get` or `fake_post` functions.
- The rate limits logic is correctly implemented and does not raise any exceptions.

Confidence: 80%

Tokens: 783 input + 175 output = 958 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	220 lines (ranges: 39-42, 45-46, 48, 52-54, 66, 68-70, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-101, 103, 105, 107-109, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246-247, 249-252, 261, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-430, 432, 435, 437-439, 441-444, 449-455, 457,

459-460, 463-466, 471-473,
476-478, 497-498, 502-505,
507-508, 511, 514-516, 518-
521, 524, 526, 528-531, 537,
539-543, 547-548, 550-552,
554-555, 557-559, 562-563,
567, 569-571, 574)

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

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_retries_on_rate_limit

1ms



6

AI ASSESSMENT

Scenario: Tests for Gemini provider**Why Needed:** To handle retries on rate limit errors**Key Assertions:**

- {'name': 'Mocked requests should be retried', 'description': 'Verify that the LLM is retried when a request exceeds the rate limit'}
- {'name': 'Llama model should not be skipped due to rate limit error', 'description': "Verify that the Llama model is not skipped if it's already been used to generate an answer"}

Confidence: 80%**Tokens:** 98 input + 121 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/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	216 lines (ranges: 32-34, 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246, 249-250, 252, 261, 263-265, 299-300, 304-306, 308-309, 340-343, 346-349, 352, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413-416, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457-458, 463-466, 471-473, 476-478, 497-498, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)

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

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_rotates_models_on_daily_limit

1ms



6

AI ASSESSMENT

Scenario:

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_rotates_models_on_daily_limit

Why Needed: To ensure that the LLM models are rotated on a daily basis and not cached for an extended period, which could lead to performance issues.**Key Assertions:**

- {'assertion_type': 'model rotation', 'expected_result': 'models are rotated daily'}
- {'assertion_type': 'cache invalidation', 'expected_result': 'cache is invalidated daily'}

Confidence: 80%**Tokens:** 100 input + 125 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	210 lines (ranges: 39-42, 45-46, 48-50, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-230, 232, 235-236, 239-244, 246, 249-250, 252, 261, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457, 459, 461-466, 471-473, 476-478, 497-499, 502-505, 507-508, 511, 514-516, 518-521, 524, 526-527, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567-571, 574)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)

src/pytest_llm_report/options.py

2 lines (ranges: 123, 171)

src/pytest_llm_report/plugin.py

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

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_skips_on_daily_limit

1ms



6

AI ASSESSMENT

Scenario: tests/test_llm_providers.py::TestGeminiProvider::test_annotate_skips_on_daily_limit**Why Needed:** Skips daily limit due to excessive annotation requests.**Key Assertions:**

- {'name': 'Mocks', 'expected': ['Mocks']}
- {'name': 'Requests', 'expected': ['Annotation Requests']}

Confidence: 80%**Tokens:** 98 input + 94 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	47 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	216 lines (ranges: 39-42, 45-46, 48-50, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-230, 232-233, 235-236, 239-244, 246, 249-250, 252, 261, 318-320, 340-343, 346-349, 352-356, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457, 459, 461-466, 471-473, 476-478, 497-499, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)

src/pytest_llm_report/options.py

2 lines (ranges: 123, 171)

src/pytest_llm_report/plugin.py

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

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_annotate_success_with_mock_response

1ms



6

AI ASSESSMENT

Scenario: Test that LiteLLM provider annotates a valid response with the correct information.**Why Needed:** Prevents regression by ensuring the provider correctly annotates responses from liteLLM.**Key Assertions:**

- status ok
- redirect

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

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/gemini.py	209 lines (ranges: 39-42, 45-46, 48-49, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-101, 103, 105, 107-109, 134-135, 137-141, 143-144, 164-166, 173-175, 178, 181, 184, 186-187, 189, 191-192, 198-206, 208-209, 222-224, 228-230, 232, 235-236, 239-244, 246, 249-250, 252, 261, 340-343, 346-349, 352, 358-361, 363-364, 366-367, 383, 385-388, 390-403, 406, 410-411, 413, 418-422, 424-425, 432, 435, 437-439, 441-444, 449-455, 457-466, 471-473, 476-478, 497-498, 502-505, 507-508, 511, 514-516, 518-521, 524, 526, 528-531, 537, 539-543, 547-548, 550-552, 554-555, 557-559, 562-563, 567, 569-571, 574)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)

src/pytest_llm_report/options.py

2 lines (ranges: 123, 171)

src/pytest_llm_report/plugin.py

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

PASSED

tests/test_llm_providers.py::TestGeminiProvider::test_exhausted_mode_l_recovers_after_24h

1ms



AI ASSESSMENT

Scenario: tests/test_llm_providers.py**Why Needed:** The LLM provider's model is exhausted after 24 hours. This test ensures the provider can recover and continue serving requests.**Key Assertions:**

- {'name': 'Provider should be able to recover from exhaustion', 'description': 'After 24 hours, the provider should still be able to serve requests'}

Confidence: 80%**Tokens:** 104 input + 94 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	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 returns an error when there are no available models.

Key Assertions:

- {'name': "The response should contain a 'no_available_models' key", 'value': "'{}scenario': ..., 'why_needed': ..., 'key_assertions': [...]}", 'expected_value': "'{}scenario': ..., 'why_needed': ..., 'key_assertions': [...]"}

Confidence: 80%

Tokens: 92 input + 126 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 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 across tests.**Key Assertions:**

- {'name': 'Model list should be empty before refresh', 'value': [], 'expected_value': []}
- {'name': 'Model list should contain expected models after refresh', 'value': [...], 'expected_value': [...]}'}

Confidence: 80%**Tokens:** 96 input + 106 output = 202 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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 on 401 error**Key Assertions:**

- Verify that the provider retries with a new token.
- Verify that the provider throws an exception for 401 error.
- Verify that the provider captures the captured keys.
- Verify that the provider increments call count correctly.
- Verify that the provider increments token count correctly.
- Verify that the provider returns a successful response after retrying with new token.

Confidence: 80%**Tokens:** 580 input + 126 output = 706 total

COVERAGE

src/pytest_llm_report/collector.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 regression where the provider does not surface completion errors in annotations.**Key Assertions:**

- The error message is 'boom' which indicates a completion error.
- The annotation has an 'error' key with the value 'boom'.
- The annotation's error contains the string 'boom'.

Confidence: 80%**Tokens:** 307 input + 98 output = 405 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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:** This test prevents the provider from silently failing when receiving an invalid key_assertions payload, making it easier to detect and diagnose issues.**Key Assertions:**

- The 'response_data' dictionary must be a valid JSON object.
- The 'json.dumps(response_data)' expression should return a valid JSON string.
- The 'response_data' dictionary must contain exactly one key_assertion value.
- The 'key_assertion' value must be a list of strings or a single string.
- The 'key_assertions' value must not be empty.

Confidence: 80%**Tokens:** 346 input + 145 output = 491 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	43 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346-348)
src/pytest_llm_report/llm/litellm_provider.py	35 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-187, 190, 192-193, 196, 204, 206, 211)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_missing_dependency

1ms



5

AI ASSESSMENT

Scenario: The LiteLLMProvider should report a missing dependency when trying to annotate a case that depends on it.

Why Needed: This test prevents a potential bug where the provider does not handle cases that depend on missing dependencies correctly.

Key Assertions:

- annotation.error == 'litellm not installed. Install with: pip install litellm'
- provider.annotate(test, 'def test_case(): assert True')
- test_case()
- assert True

Confidence: 80%

Tokens: 271 input + 111 output = 382 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	9 lines (ranges: 65-66, 87-89, 97-99, 105)
src/pytest_llm_report/llm/litellm_provider.py	8 lines (ranges: 37-38, 41, 82-86)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_success_with_mock_response

1ms



6

AI ASSESSMENT

Scenario: Test that LiteLLMProvider annotates a successful response with the correct key assertions and confidence level.

Why Needed: Prevents regression by verifying that the provider correctly handles successful responses.

Key Assertions:

- status ok
- redirect

Confidence: 80%

Tokens: 475 input + 65 output = 540 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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 uses prompt_override when provided.**Why Needed:** To prevent a bug where the provider does not override prompts correctly.**Key Assertions:**

- {'message': 'The `prompt_override` parameter is present in the config.', 'content': 'CUSTOM PROMPT'}
- {'message': 'The `prompt_override` value matches the expected custom prompt.', 'content': 'CUSTOM PROMPT'}
- {'message': 'The `messages` attribute of the fake completion function contains the correct key assertion.', 'content': 'a'}

Confidence: 80%**Tokens:** 373 input + 125 output = 498 total

COVERAGE

src/pytest_llm_report/collector.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 the LiteLLMProvider's annotate method with token usage.**Why Needed:** This test prevents regression in handling token usage data from LiteLLM providers.**Key Assertions:**

- Verifies that the annotation object has a 'token_usage' attribute.
- Checks if the 'prompt_tokens', 'completion_tokens', and 'total_tokens' attributes are correctly set.
- Asserts that the 'token_usage' attribute is not None.
- Ensures that the values of 'prompt_tokens', 'completion_tokens', and 'total_tokens' match the expected values.
- Verifies that the total tokens count matches the sum of prompt and completion tokens.

Confidence: 80%**Tokens:** 426 input + 149 output = 575 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	39 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-187, 190, 192-193, 196-201, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_api_base_pass through

1ms



6

AI ASSESSMENT

Scenario: The LiteLLM provider passes the `litellm_api_base` attribute to the completion call of its API.

Why Needed: This test prevents a regression where the provider does not set the `litellm_api_base` attribute correctly, causing unexpected behavior in downstream applications.

Key Assertions:

- The value of `litellm_api_base` is set to `https://proxy.corp.com/v1`.
- The `litellm_api_base` attribute is present in the response data.
- The `api_base` key is present in the captured dictionary.
- The value of `api_base` is equal to `https://proxy.corp.com/v1`.
- The `litellm_api_base` attribute is set correctly for the given configuration.
- No exception is raised when calling the completion function with the provided arguments.

Confidence: 80%

Tokens: 387 input + 192 output = 579 total

COVERAGE

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

1ms



6

AI ASSESSMENT

Scenario: The test verifies that the LiteLLM provider passes a static API key to the completion call.

Why Needed: This test prevents regression where the API key is not passed through correctly, potentially causing unexpected behavior or errors.

Key Assertions:

- The API key is set to 'static-key-placeholder' in the captured dictionary.
- The API key matches the expected value of 'static-key-placeholder'.
- The API key is present in the captured dictionary.

Confidence: 80%

Tokens: 384 input + 108 output = 492 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	35 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 170-174, 176-178, 182, 186-188, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_auth_error_without_refresher 1ms 5

AI ASSESSMENT

Scenario: Test that the LiteLLM provider returns an auth error when no refresher is configured.

Why Needed: Prevents a bug where the provider does not return an authentication error for cases without token refresh.

Key Assertions:

- The function `fake_completion` raises a `FakeAuthError` with message '401 Unauthorized'.
- The `LiteLLMProvider` instance is configured to use the 'gpt-4o' model, but no token refresh is provided.
- The provider returns an authentication error in its annotation.

Confidence: 80%

Tokens: 338 input + 125 output = 463 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/litellm_provider.py	36 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 122, 132-133, 170-174, 176-178, 182, 186-187, 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_auth_retry_fails_on_second_attempt 2.00s 6

AI ASSESSMENT

Scenario: The test verifies that the LiteLLMProvider reports an authentication error when retrying after a second failure.

Why Needed: This test prevents regression where the provider fails to report an authentication error on subsequent retries.

Key Assertions:

- The 'error' attribute of the annotation is not None.
- The value of 'Authentication failed' in the 'error' attribute contains the string '401 Unauthorized'.
- The 'error' attribute contains a string that indicates an authentication failure, such as '401 Unauthorized'.

Confidence: 80%

Tokens: 419 input + 121 output = 540 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	29 lines (ranges: 65-66, 87-89, 97-99, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267)
src/pytest_llm_report/llm/litellm_provider.py	51 lines (ranges: 37-38, 41-42, 44-48, 60-61, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 122, 124-127, 129-130, 132-133, 141-142, 170-174, 176-178, 182, 186-188, 190)
src/pytest_llm_report/llm/token_refresh.py	31 lines (ranges: 59-60, 63-66, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156, 160-162)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

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

AI ASSESSMENT

Scenario: The test verifies that the LiteLLMProvider class correctly handles a context too long error in its `_parse_response` method.

Why Needed: This test prevents regression where the provider incorrectly raises an `AuthenticationError` when encountering a response with an invalid JSON structure.

Key Assertions:

- assert `annotation.error` is not `None`
- assert 'Context too long for this model' in `str(annotation)`
- assert `annotation.json` is not `None`
- assert 'scenario' in `annotation.json`
- assert 'why_needed' in `annotation.json`
- assert 'key_assertions' in `annotation.json`
- assert 'error' in `annotation.json`
- assert 'Context too long for this model' in `str(annotation.json)`

Confidence: 80%

Tokens: 370 input + 161 output = 531 total

COVERAGE

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



AI ASSESSMENT

Scenario: test_get_max_context_tokens_dict_format

Why Needed: To ensure that the `get_max_context_tokens` method returns a dictionary in the expected format.

Key Assertions:

- {'name': 'result is a dictionary', 'expected_value': {'max_tokens': 16384}, 'actual_value': '16384'}

Confidence: 80%

Tokens: 218 input + 86 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/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_content_tokens_fallback_on_error

1ms



AI ASSESSMENT

Scenario: tests/test_llm_providers.py

Why Needed: To ensure that the LLM provider returns a JSON response with the expected structure when an error occurs.

Key Assertions:

- {'name': 'JSON structure', 'expected': {'scenario': '...', 'why_needed': '...', 'key_assertions': ['...']}, 'actual': {'scenario': '...', 'why_needed': '...', 'key_assertions': ['...']}}}

Confidence: 80%

Tokens: 101 input + 110 output = 211 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/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 that the LiteLLM provider correctly returns the maximum number of context tokens.

Key Assertions:

- {'name': 'max_context_tokens', 'expected_value': 8192, 'actual_value': 0}
- {'name': 'provider', 'expected_value': 'LiteLLMProvider', 'actual_value': 'LiteLLMProvider'}

Confidence: 80%

Tokens: 213 input + 124 output = 337 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/litellm_provider.py	9 lines (ranges: 37-38, 41, 221-222, 224, 227-229)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestLiteLLMProvider::test_is_available_with_module

1ms



5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py::TestLiteLLMProvider::test_is_available_with_module

Why Needed: To ensure the liteLLM provider can detect installed modules.

Key Assertions:

- {'expected': {'module_name': 'litellm'}, 'actual': '__import__()'}
- message_type_error_message_id_or_key_not_found_0_1: module not found

Confidence: 80%

Tokens: 160 input + 102 output = 262 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/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: LiteLLM provider uses TokenRefresher for dynamic tokens.**Why Needed:** The test prevents a scenario where the LLM provider fails to refresh its token due to an interval setting that is too short, causing it to run out of tokens before the next refresh cycle.**Key Assertions:**

- assert captured['api_key'] == 'dynamic-token-789',
- assert captured['provider'] == 'litellm',
- assert captured['model'] == 'gpt-4o',
- assert captured['token_refresh_command'] == 'get-token',
- assert captured['token_refresh_interval'] == 3600,
- assert captured['expected_token'] == None,
- assert captured['actual_token'] == 'dynamic-token-789',

Confidence: 80%**Tokens:** 442 input + 180 output = 622 total

COVERAGE

src/pytest_llm_report/collector.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 on transient errors.**Why Needed:** This test prevents a regression where the provider fails to retry transient errors, causing the test to fail intermittently.**Key Assertions:**

- The provider should not raise an exception when it encounters a transient error.
- The provider should retry the operation after a transient error.
- The number of retries should be limited to 5.
- The provider should only retry if the network error is transient (i.e., not due to other issues).
- The provider should not retry indefinitely when it encounters a transient error.
- The test should pass even if the provider fails to retry transient errors occasionally.

Confidence: 80%**Tokens:** 426 input + 155 output = 581 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/litellm_provider.py	42 lines (ranges: 37-38, 41, 60, 62, 82-83, 89, 92, 95, 98, 100-101, 104, 106-107, 112, 114, 116-117, 120, 135, 139, 141-142, 170-174, 176-178, 182, 186-187, 190, 192-193, 196, 204, 213)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_annotate_fallbacks_on_context_length_error

1ms



7

AI ASSESSMENT

Scenario: tests/test_llm_providers.py**Why Needed:** To ensure that the LLM provider correctly handles context length errors during annotation.**Key Assertions:**

- {'name': 'expected_exception', 'type': 'Exception'}
- {'name': 'message', 'type': 'str'}

Confidence: 80%**Tokens:** 103 input + 83 output = 186 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	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 of OllamaProvider returns an error message when a call to the annotated function fails.

Why Needed: This test prevents regression in case where the annotation fails due to a timeout or other unforeseen reason during the retry process.

Key Assertions:

- The error message returned by the annotate method is 'Failed after 2 retries. Last error: boom'.
- The function `test_case` was not executed successfully.
- The last error raised during the annotation attempt is 'boom'.

Confidence: 80%**Tokens:** 347 input + 124 output = 471 total

COVERAGE

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

Key Assertions:

- assert annotation.error == 'httpx not installed. Install with: pip install httpx'
- provider.annotate(test, "def test_case(): assert True")

Confidence: 80%

Tokens: 268 input + 97 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/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: To test the immediate failure of annotating runtime errors.

Key Assertions:

- {'name': 'expected_exception', 'type': 'Exception'}
- {'name': 'expected_message', 'type': 'RuntimeError'}

Confidence: 80%

Tokens: 99 input + 96 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	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 annotating a full flow of an Ollama provider with mocked HTTP returns the expected annotations.

Why Needed: Prevents auth bugs by ensuring the correct status and token are returned in case of errors.

Key Assertions:

- check status
- validate token

Confidence: 80%

Tokens: 414 input + 68 output = 482 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	46 lines (ranges: 65-66, 87-89, 97-98, 105, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 264, 266-267, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/ollama.py	34 lines (ranges: 42-43, 49, 52, 55, 58, 60-61, 63-67, 71-72, 83, 92, 190, 192-200, 204-207, 209, 211-212)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_annotate_with_prompt_override 1ms 6

AI ASSESSMENT

Scenario: Test that LiteLLMProvider overrides the prompt when provided with a custom prompt.

Why Needed: To prevent bugs where LiteLLM provider does not override the prompt and instead uses the default one.

Key Assertions:

- The annotation should have no error.
- The captured messages should contain the custom prompt.
- The content of the captured message should be 'CUSTOM PROMPT'.
- The key assertion 'a' should match the expected value in the captured message.

Confidence: 80%

Tokens: 373 input + 113 output = 486 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	37 lines (ranges: 65-66, 185, 188, 191-198, 200, 212, 214, 216, 219-221, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/ollama.py	34 lines (ranges: 42-43, 49, 52-53, 58, 60-61, 63-67, 71-72, 83, 92, 190, 192-200, 204-207, 209, 211-212)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_annotate_with_token_usage 1ms 6

AI ASSESSMENT

Scenario: Test LiteLLM provider annotates token usage for a case with a specific configuration and test result.

Why Needed: This test prevents regression that would occur if the LiteLLMProvider's annotate method does not correctly handle cases where there is no response or incomplete response.

Key Assertions:

- The annotation object contains the correct number of prompt tokens, completion tokens, and total tokens.
- The token usage is not None.
- The prompt tokens are equal to 100.
- The completion tokens are equal to 50.
- The total tokens are equal to 150.

Confidence: 80%

Tokens: 426 input + 135 output = 561 total

COVERAGE

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

Why Needed: This test prevents regression where the Ollama provider fails to make a successful API call to generate a response.

Key Assertions:

- The 'response' key in the result dictionary should be equal to 'test response'.
- The 'url' key in the captured dictionary should contain the string 'http://localhost:11434/api/generate'.
- The 'json' key in the captured dictionary should have a 'model' value of 'llama3.2:1b'.
- The 'json' key in the captured dictionary should have a 'prompt' value of 'test prompt'.
- The 'json' key in the captured dictionary should have a 'system' value of 'system prompt'.
- The 'stream' key in the captured dictionary should be False.
- The 'timeout' key in the captured dictionary should be equal to 60 seconds.

Confidence: 80%

Tokens: 470 input + 226 output = 696 total

COVERAGE

src/pytest_llm_report/collector.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 Ollama provider uses the default model when not specified.

Why Needed: This test prevents a regression where the default model is used instead of the provided model.

Key Assertions:

- The value of 'model' in the captured JSON response should be 'llama3.2'.
- The value of 'model' in the captured JSON response should not be empty.
- The value of 'model' in the captured JSON response is a string, not None or an empty string.

Confidence: 80%

Tokens: 344 input + 120 output = 464 total

COVERAGE

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

Why Needed: To test the failure case when the server is unavailable.

Key Assertions:

- {'name': 'provider._check_availability()' should return False', 'description': "The provider's check_availability method should return False when the server is not available."}

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 Ollama provider's check_availability method with a non-200 status code.**Key Assertions:**

- {'message': 'Expected False', 'expected_value': False, 'type': 'assertion'}
- {'message': 'OllamaProvider._check_availability() returned True instead of False.', 'expected_value': True, 'type': 'exception'}

Confidence: 80%**Tokens:** 197 input + 118 output = 315 total

COVERAGE

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



AI ASSESSMENT

Scenario: Test checks availability of Ollama provider when it returns a successful response.**Why Needed:** This test prevents regression where the provider fails to return a successful response when available.**Key Assertions:**

- The method `_check_availability()` of the `OllamaProvider` instance should return `True`.
- The status code of the `/api/tags` endpoint should be `200`.
- The URL '`/api/tags`' is present in the request.
- No exception is raised when making a GET request to the `/api/tags` endpoint.

Confidence: 80%**Tokens:** 296 input + 123 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/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 the correct behavior of the Ollama provider when retrieving context length from the LLM.**Key Assertions:**

- {'name': 'Context length should be an integer', 'expected_value': 100, 'actual_value': 123}
- {'name': 'Context length should not exceed the maximum allowed value', 'expected_value': 100, 'actual_value': 150}

Confidence: 80%**Tokens:** 99 input + 106 output = 205 total

COVERAGE

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



AI ASSESSMENT

Scenario: Tests for OLLAMA provider

Why Needed: To handle errors that may occur during the execution of the model

Key Assertions:

- {'name': 'Expected a fallback to use max_context_tokens', 'description': 'The maximum context tokens should be used when an error occurs'}

Confidence: 80%

Tokens: 101 input + 78 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/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', 'expected_value': 1024, 'actual_value': 0}
- {'name': 'context_token_count', 'expected_value': 512, 'actual_value': 0}

Confidence: 80%**Tokens:** 99 input + 113 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	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/test_llm_providers.py**Why Needed:** To ensure the Ollama provider can correctly retrieve the maximum context tokens from its parameters.**Key Assertions:**

- {'name': 'max_context_tokens', 'expected_value': 32, 'actual_value': 16}
- {'name': 'parameter_count', 'expected_value': 2, 'actual_value': 1}

Confidence: 80%**Tokens:** 97 input + 106 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	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	15 lines (ranges: 138, 140, 142-147, 149-150, 156, 158, 160-162)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_get_max_context_tokens_non_200_status

1ms



5

AI ASSESSMENT

Scenario: tests/test_llm_providers.py**Why Needed:** To ensure that the LLM provider returns a valid response when the API call to get max context tokens fails with a non-200 status code.**Key Assertions:**

- The response from the API call is not an object.
- The 'max_context_tokens' key in the response is not present or its value is empty.

Confidence: 80%**Tokens:** 101 input + 91 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	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 that the Ollama provider always returns `is_local=True`.

Key Assertions:

- {'name': 'provider is an instance of OllamaProvider', 'expected_value': 'True'}
- {'name': 'is_local() method returns True for OllamaProvider instances', 'expected_value': 'True'}

Confidence: 80%

Tokens: 123 input + 116 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	2 lines (ranges: 65-66)
src/pytest_llm_report/llm/ollama.py	1 lines (ranges: 128)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_parse_response _invalid_json

1ms



AI ASSESSMENT

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

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

Confidence: 80%**Tokens:** 138 input + 73 output = 211 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	7 lines (ranges: 65-66, 325-326, 329-331)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-52, 55)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_parse_response
_invalid_key_assertions

1ms



AI ASSESSMENT

Scenario: {'id': 1, 'description': 'Test case for Ollama provider with invalid key assertions'}

Why Needed: This test is necessary to ensure the Ollama provider rejects invalid key assertions in its responses.

Key Assertions:

- {'name': 'Invalid response', 'message': "The provided 'key_assertions' value is not a list."}

Confidence: 80%

Tokens: 174 input + 143 output = 317 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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', 'description': 'The extracted JSON should be in the expected format.'}
- {'name': 'Expected key assertion', 'description': 'The provider should assert that it has found a specific key in the JSON object.'}

Confidence: 80%**Tokens:** 127 input + 123 output = 250 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/base.py	20 lines (ranges: 65-66, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/schemas.py	6 lines (ranges: 38, 42-44, 46-47)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_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': 'Expected response type to be dict', 'value': 'expected response type is a dictionary'}
- {'name': 'Expected keys in response', 'value': "keys in the response should include 'scenario', 'why_needed', and 'key_assertions'"}

Confidence: 80%**Tokens:** 128 input + 127 output = 255 total

COVERAGE

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

Why Needed: Prevents bugs that may occur when parsing invalid or malformed JSON responses.

Key Assertions:

- assert a
- assert b

Confidence: 80%

Tokens: 292 input + 57 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/base.py	20 lines (ranges: 65-66, 325-326, 329-330, 333-334, 337-339, 342, 344, 346, 351, 353-357)
src/pytest_llm_report/llm/schemas.py	7 lines (ranges: 38, 42-43, 50-53)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

tests/test_llm_utils.py

6 tests

PASSED

tests/test_llm_utils.py::test_distribute_token_budget_constrained

1ms



AI ASSESSMENT

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

Why Needed: This test prevents a potential bug where the algorithm does not satisfy the constraint of smaller files first, leading to inefficient allocation of tokens.

Key Assertions:

- The function `distribute_token_budget` should return an allocation dictionary with 'small.py' as the key and 10 as its value.
- The function `distribute_token_budget` should return an allocation dictionary with 'large.py' as the key and between 30 and 45 (inclusive) as its value.
- The total allocated tokens for both files should be less than or equal to the total budget of 60.
- The remaining tokens after allocating to 'small.py' should be greater than or equal to the remaining budget of 44.
- The allocation of 'large.py' should not exceed the remaining budget of 44.

Confidence: 80%

Tokens: 396 input + 193 output = 589 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/utils.py	32 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 90-91, 93-94, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_utils.py::test_distribute_token_budget_empty

1ms



AI ASSESSMENT

Scenario: tests/test_llm_utils.py::test_distribute_token_budget_empty**Why Needed:** Verify behavior with empty input or no budget.**Key Assertions:**

- {'name': 'assert an empty dictionary is returned when {} is passed to distribute_token_budget(100)', 'expected_output': {}, 'actual_output': 'tests/test_llm_utils.py::test_distribute_token_budget_empty'}
- {'name': "assert an empty dictionary is returned when {'f1': 'c'} is passed to distribute_token_budget(0)", 'expected_output': {}, 'actual_output': 'tests/test_llm_utils.py::test_distribute_token_budget_empty'}

Confidence: 80%**Tokens:** 115 input + 156 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/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 in case where both files are too large to fit within the budget, causing unfair distribution of token budgets.

Key Assertions:

- The allocation for `l1.py` should be between 35 and 50 tokens.
- The allocation for `l2.py` should also be between 35 and 50 tokens.
- The absolute difference in allocations for `l1.py` and `l2.py` should not exceed 1 token.

Confidence: 80%

Tokens: 327 input + 121 output = 448 total

COVERAGE

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

- {'name': 'length of allocations is equal to 3', 'expected_value': 3, 'actual_value': 0}
- {'name': 'number of files allocated is less than or equal to max_files', 'expected_value': 3, 'actual_value': 0}

Confidence: 80%**Tokens:** 133 input + 121 output = 254 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/utils.py	28 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_utils.py::test_distribute_token_budget_sufficient

1ms



AI ASSESSMENT

Scenario: Verify that all files get full content when the token budget is sufficient.

Why Needed: This test prevents a potential regression where the LLM might not be able to handle large budgets without running out of tokens or headers.

Key Assertions:

- The number of allocations should match the total needed for each file (32 tokens per file).
- Each allocation should contain exactly 10 tokens (40 characters in 'f1.py' and 40 characters in 'f2.py').

Confidence: 80%

Tokens: 332 input + 114 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/utils.py	28 lines (ranges: 20, 42, 46-47, 51-53, 55-60, 66-67, 70-71, 73, 75, 77, 79, 81-82, 84, 86-87, 96, 98)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_llm_utils.py::test_estimate_tokens

1ms



AI ASSESSMENT

Scenario: Verify the rough token estimation (chars / 4) for an empty string.

Why Needed: Prevents a potential division by zero error when estimating tokens.

Key Assertions:

- assert estimate_tokens('') == 1
- assert estimate_tokens('a') == 1
- assert estimate_tokens('aaaa') == 1
- assert estimate_tokens('aaaa' * 10) == 10

Confidence: 80%

Tokens: 217 input + 103 output = 320 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/utils.py	1 lines (ranges: 20)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

tests/test_models.py

29 tests

PASSED

tests/test_models.py::TestArtifactEntry::test_to_dict

1ms



AI ASSESSMENT

Scenario: The test verifies that a CoverageEntry object can be successfully converted to a dictionary.

Why Needed: This test prevents the regression of coverage data not being properly serialized to JSON.

Key Assertions:

- assert d['file_path'] == 'src/foo.py'
- assert d['line_ranges'] == '1-3, 5, 10-15'
- assert d['line_count'] == 10
- The file path is correctly serialized as 'src/foo.py'.
- The line ranges are correctly serialized as '1-3, 5, 10-15'.
- The line count is correctly serialized as 10.
- CoverageEntry object has been successfully converted to a dictionary.

Confidence: 80%

Tokens: 255 input + 165 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/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()` returns the expected dictionary structure for a CoverageEntry object.

Why Needed: This test prevents a potential bug where the `to_dict()` method of `CoverageEntry` does not return the correct dictionary structure, potentially causing unexpected behavior or errors in downstream code.

Key Assertions:

- The 'file_path' key should contain the string 'src/foo.py'.
- The 'line_ranges' key should contain a comma-separated list of strings representing the range of lines covered by the entry. The expected format is '1-3, 5, 10-15'.
- The 'line_count' key should contain an integer value equal to the number of lines in the coverage entry.
- Each assertion checks that the values returned by `to_dict()` match the expected values.

Confidence: 80%

Tokens: 255 input + 183 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/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: The `CoverageEntry` class should serialize correctly to a dictionary.

Why Needed: This test prevents a potential bug where the serialization of `CoverageEntry` objects is not accurate, potentially leading to incorrect coverage data being reported.

Key Assertions:

- The 'file_path' key in the serialized dictionary matches the expected value.
- The 'line_ranges' key in the serialized dictionary matches the expected format.
- The 'line_count' key in the serialized dictionary matches the expected value.

Confidence: 80%

Tokens: 255 input + 115 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	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 bug where an empty annotation is not properly initialized with default values.

Key Assertions:

- annotation.scenario == "" (empty string)
- annotation.why_needed == "" (empty string) (default value for LlmAnnotation)
- annotation.key_assertions == [] (no key assertions are performed on an empty annotation)
- assert annotation.confidence is None (expected confidence to be None for an empty annotation)
- assert annotation.error is None (expected error to be None for an empty annotation)

Confidence: 80%

Tokens: 212 input + 139 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/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 `LlmAnnotation` object can be serialized into a dictionary with required fields.

Why Needed: This test prevents regression by ensuring that the minimal annotation is properly serialized and includes all necessary information.

Key Assertions:

- The 'scenario' key should be present in the dictionary.
- The 'why_needed' key should be present in the dictionary.
- The 'key_assertions' key should be present in the dictionary.
- The 'confidence' key should not be present in the dictionary when it is None.

Confidence: 80%

Tokens: 230 input + 127 output = 357 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	9 lines (ranges: 130-133, 135, 137, 139, 141, 143)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestLlmAnnotation::test_to_dict_with_all_fields

1ms



AI ASSESSMENT

Scenario: Test that the `to_dict` method returns all required fields for a full annotation.

Why Needed: Prevents regression in LLMAnnotation class, where some fields are missing or incomplete.

Key Assertions:

- Asserts that the 'scenario' field is present and matches the expected value.
- Asserts that the 'confidence' field has a value within the expected range (0.0 to 1.0).
- Asserts that the 'context_summary' field contains the expected keys ('mode' and 'bytes') with correct values.
- Asserts that all required fields are present in the resulting dictionary.
- Asserts that the 'error' field is absent or None, as per the test's expectation.

Confidence: 80%

Tokens: 284 input + 166 output = 450 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	11 lines (ranges: 130-133, 135-137, 139-141, 143)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestReportRoot::test_default_report

1ms



AI ASSESSMENT

Scenario: Test Default Report

Why Needed: Prevents a potential bug where the default report is missing required schema version and empty test lists.

Key Assertions:

- The 'schema_version' key should be present in the dictionary with value equal to SCHEMA_VERSION.
- The 'tests' key should be an empty list.
- The 'warnings' key should not be present in the dictionary.
- The 'collection_errors' key should not be present in the dictionary.
- If the schema version is missing, a KeyError should be raised when trying to access it.
- If the tests are missing, a KeyError should be raised when trying to access them.
- If the warnings or collection errors lists are non-empty, an AssertionError should be raised.

Confidence: 80%**Tokens:** 231 input + 170 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/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 ReportRoot::test_report_with_collection_errors verifies that the test reports collection errors properly.

Why Needed: This test prevents a regression where collection errors are not reported correctly.

Key Assertions:

- The report should contain exactly one collection error.
- The first collection error should be for the file "test_bad.py".
- The node ID of the collection error should match the provided node ID in the ReportRoot constructor.

Confidence: 80%

Tokens: 237 input + 103 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/models.py	58 lines (ranges: 241-243, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526-528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestReportRoot::test_report_with_warnings

1ms



AI ASSESSMENT

Scenario: TestReportRoot::test_report_with_warnings**Why Needed:** The test is needed to ensure that the ReportRoot class correctly handles warnings.**Key Assertions:**

- {'name': 'Length of warnings list', 'expected': 1, 'actual': 0}
- {'name': 'Code in first warning', 'expected': 'W001', 'actual': 'No coverage'}

Confidence: 80%**Tokens:** 144 input + 105 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/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/models.py	55 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528-530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED tests/test_models.py::TestReportRoot::test_tests_sorted_by_nodeid 1ms 3

AI ASSESSMENT

Scenario: Tests should be sorted by nodeid in output.

Why Needed: Because the current implementation does not sort tests by nodeid, which can lead to incorrect test results if multiple tests have the same nodeid.

Key Assertions:

- {'assertion_type': 'contains', 'expected_value': 'a_test.py::test_a', 'actual_value': 'z_test.py::test_z'}
- {'assertion_type': 'contains', 'expected_value': 'm_test.py::test_m', 'actual_value': 'z_test.py::test_z'}

Confidence: 80%

Tokens: 215 input + 139 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	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 'test_to_dict_without_detail' verifies that a ReportWarning object is created without detail.

Why Needed: This test prevents the creation of a ReportWarning object with detail, which could lead to unexpected behavior or errors.

Key Assertions:

- The warning should be excluded from the dictionary.
- The warning message should not contain any details.
- The 'detail' key should not exist in the warning dictionary.
- The warning code should still match the expected value.
- The warning message should remain unchanged.

Confidence: 80%

Tokens: 223 input + 122 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/errors.py	7 lines (ranges: 70-71, 73-75, 77, 79)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestRunMeta::test_aggregation_fields_present

1ms



AI ASSESSMENT

Scenario: Verify that RunMeta has aggregation fields.

Why Needed: Prevent regression where RunMeta is missing aggregation fields, potentially leading to incorrect aggregation results.

Key Assertions:

- assert d['run_id'] == 'run-123'
- assert d['run_group_id'] == 'group-456'
- assert d['is_aggregated'] is True
- assert d['aggregation_policy'] == 'merge'
- assert d['run_count'] == 3
- assert len(d['source_reports']) == 2

Confidence: 80%

Tokens: 343 input + 128 output = 471 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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:** To prevent regression and ensure consistent behavior when annotations are disabled.**Key Assertions:**

- The 'llm_annotations_enabled' key is present in the data dictionary.
- The 'llm_provider' key is not present in the data dictionary.
- The 'llm_model' key is not present in the data dictionary.

Confidence: 80%**Tokens:** 232 input + 97 output = 329 total

COVERAGE

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

Why Needed: This test prevents regression by ensuring that the LLM traceability fields are properly set when running with llm_provider='ollama' and llm_model='llama3.2:1b'.

Key Assertions:

- data['llm_annotations_enabled'] is True
- data['llm_provider'] == 'ollama'
- data['llm_model'] == 'llama3.2:1b'
- data['llm_context_mode'] == 'complete'
- data['llm_annotations_count'] == 10
- data['llm_annotations_errors'] == 2

Confidence: 80%

Tokens: 327 input + 160 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/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: The test is necessary because it checks if a non-aggregated report excludes source reports.

Key Assertions:

- {'name': 'Non-aggregated report should not include source_reports', 'expected_result': {'source_reports': [], 'is_aggregated': False}}

Confidence: 80%

Tokens: 130 input + 94 output = 224 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 where RunMeta's optional fields are not properly populated in the dictionary.**Key Assertions:**

- Verify that 'git_sha' is set to 'abc1234'.
- Verify that 'git_dirty' is True.
- Verify that 'repo_version' is set to '1.0.0'.
- Verify that 'repo_git_sha' is set to 'abc1234'.
- Verify that 'repo_git_dirty' is False.
- Verify that 'plugin_git_sha' is set to 'def5678'.
- Verify that 'plugin_git_dirty' is False.
- Verify that 'config_hash' is set to 'def5678'.
- Verify the length of 'source_reports' is 1.

Confidence: 80%**Tokens:** 483 input + 188 output = 671 total

COVERAGE

src/pytest_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 the 'RunMeta' object's run status fields.

Why Needed: This test prevents a potential bug where the 'RunMeta' object is not properly initialized with all required fields, leading to incorrect or missing data in its attributes.

Key Assertions:

- The 'exit_code' attribute of the 'RunMeta' object should be equal to 1.
- The 'interrupted' attribute of the 'RunMeta' object should be True.
- The 'collect_only' attribute of the 'RunMeta' object should be True.
- The 'collected_count' attribute of the 'RunMeta' object should be equal to 10.
- The 'selected_count' attribute of the 'RunMeta' object should be equal to 8.
- The 'deselected_count' attribute of the 'RunMeta' object should be equal to 2.

Confidence: 80%

Tokens: 285 input + 196 output = 481total

COVERAGE

src/pytest_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: To ensure the schema version is in semver format, which is a standard way of expressing software compatibility and change history.

Key Assertions:

- {'name': "schema_version.split('.').should.have.length.equal.to(3)", 'message': "The schema version should have exactly 3 parts (e.g., '1.2.3')", 'type': 'assertion'}
- {'name': 'each part.should.be.a.digit', 'message': 'Each part of the schema version should be a digit (0-9)', 'type': 'assertion'}

Confidence: 80%

Tokens: 115 input + 162 output = 277 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: To ensure that the ReportRoot 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 + 104 output = 223 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	54 lines (ranges: 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestSourceCoverageEntry::test_to_dict

1ms



AI ASSESSMENT

Scenario: Test coverage entry serialization.**Why Needed:** This test prevents a bug where the `CoverageEntry` object is not correctly serialized to JSON.**Key Assertions:**

- The 'file_path' key should be present and contain the expected value.
- The 'line_ranges' key should be present and contain the expected string.
- The 'line_count' key should be present and contain the expected integer value.
- All assertions should pass for the `CoverageEntry` object to be considered valid.

Confidence: 80%**Tokens:** 256 input + 117 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	8 lines (ranges: 96-103)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestSourceReport::test_to_dict_minimal

1ms



AI ASSESSMENT

Scenario: The test verifies that the `to_dict` method of `LlmAnnotation` returns a dictionary with required fields.

Why Needed: This test prevents a potential bug where the minimal annotation is missing some required fields.

Key Assertions:

- assert 'scenario' in d: The expected field is present in the output dictionary.
- assert 'why_needed' in d: The expected field is present in the output dictionary.
- assert 'key_assertions' in d: The expected field is present in the output dictionary.
- assert 'confidence' not in d: The unexpected field is not present in the output dictionary.

Confidence: 80%

Tokens: 229 input + 143 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	5 lines (ranges: 286-288, 290, 292)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestSourceReport::test_to_dict_with_run_id

1ms



AI ASSESSMENT

Scenario: TestSourceReport to_dict_with_run_id

Why Needed: To ensure SourceReport objects are properly serialized and can be easily converted back into a dictionary.

Key Assertions:

- {'name': "Expected 'run_id' key in source object", 'value': 'run-1'}

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/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 the `to_dict` method of `CoverageEntry` class.

Why Needed: This test prevents a potential bug where the `to_dict` method does not correctly serialize the coverage entry data.

Key Assertions:

- The 'file_path' key in the dictionary should be equal to 'src/foo.py'.
- The 'line_ranges' key in the dictionary should be equal to '1-3, 5, 10-15'.
- The 'line_count' key in the dictionary should be equal to 10.
- The 'coverage_data' key (if present) should not be missing from the dictionary.
- Any additional keys or values in the dictionary should match the expected format.
- The 'file_path' value should start with a forward slash '/' and end with a backslash '\'.
- The 'line_ranges' value should contain valid ranges separated by commas (e.g., '1-3, 5, 10-15').
- The 'line_count' value should be an integer.
- Any other unexpected values in the dictionary should raise an AssertionError.

Confidence: 80%

Tokens: 254 input + 247 output = 501 total

COVERAGE

src/pytest_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: tests/test_models.py::TestTestCaseResult::test_minimal_result**Why Needed:** This test prevents a regression where the minimal result is missing required fields.**Key Assertions:**

- The 'nodeid' field should be present in the result dictionary.
- The 'outcome' field should be present in the result dictionary.
- The 'duration' field should be present in the result dictionary.
- The 'phase' field should be present in the result dictionary.

Confidence: 80%**Tokens:** 244 input + 112 output = 356 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	19 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

AI ASSESSMENT

Scenario: Test `test_result_with_coverage` verifies that the `TestCaseResult` object includes a coverage list.

Why Needed: This test prevents regressions where the coverage is not included in the result, potentially leading to incorrect reporting of code coverage.

Key Assertions:

- The `coverage` key should be present and contain exactly one entry.
- The `file_path` attribute of the first coverage entry should match the nodeid of the test.
- All file paths in the coverage list should be relative to the source directory.
- Each coverage entry should have a `line_ranges` attribute with values '1-5' and a `line_count` attribute equal to 5.
- The total number of lines covered by code in the coverage list should match the total number of lines in the test file.
- All line ranges in the coverage list should be contiguous (i.e., no gaps between them).
- No coverage entries should have a `line_ranges` attribute with values that are not '1-5'.

Confidence: 80%

Tokens: 256 input + 226 output = 482 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	24 lines (ranges: 65-68, 190, 194-199, 201, 203, 205, 207, 210-212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models.py::TestTestCaseResult::test_result_with_llm_opt_out

1ms



AI ASSESSMENT

Scenario: tests/test_models.py::TestTestCaseResult::test_result_with_llm_opt_out

Why Needed: To ensure that the `llm_opt_out` flag is correctly set for test results with LLM opt-out.

Key Assertions:

- {'name': 'assert llm_opt_out is True in result dictionary', 'expected': True, 'actual': 'is True'}

Confidence: 80%

Tokens: 145 input + 97 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/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:**

- {'field_name': 'rerun_count', 'expected_value': 2, 'actual_value': "assert d['rerun_count'] == 2"}
- {'field_name': 'final_outcome', 'expected_value': 'passed', 'actual_value': "assert d['final_outcome'] == 'passed'"}

Confidence: 80%**Tokens:** 162 input + 127 output = 289 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the `result` object contains fields that are specific to reruns (e.g. 'rerun_count', 'rerun_message') which should be excluded in a result without reruns.

Key Assertions:

- {'name': 'Result without reruns excludes rerun fields', 'description': 'The `result` object should not contain the following fields: rerun_count, rerun_message'}

Confidence: 80%

Tokens: 152 input + 128 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	19 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

 tests/test_models_coverage.py

15 tests

PASSED

tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_all_optional_fields 1ms 4

AI ASSESSMENT

Scenario:

tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_all_optional_fields

Why Needed: Prevents bar because llm_opt_out is True and llm_context_override is set to 'complete'. Without these, the test would fail due to missing coverage.

Key Assertions:

- assert result['llm_opt_out'] == True
- assert len(result['coverage']) == 1
- assert result['llm_annotation']['scenario'] == 'Tests foo'
- assert result['captured_stdout'] == 'stdout content'
- assert result['captured_stderr'] == 'stderr content'

Confidence: 80%

Tokens: 454 input + 145 output = 599 total

COVERAGE

src/pytest_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 regression where the 'to_dict' method does not include all required artifacts.

Key Assertions:

- The length of 'artifacts' in the result should be 2.
- The path of the first artifact ('report.html') should match its expected value.
- All artifacts should have their paths included in the result dictionary.

Confidence: 80%

Tokens: 264 input + 99 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	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 reports collection errors without checking if they are actually present in the report.

Key Assertions:

- The length of `result['collection_errors']` is 1 and its first element's 'nodeid' is 'broken_test.py'.
- The value of `result['collection_errors'][0]` is an instance of `CollectionError` with a non-empty `nodeid` attribute.
- The value of `result['collection_errors'][0]['nodeid']` matches the expected value 'broken_test.py'.

Confidence: 80%

Tokens: 243 input + 144 output = 387 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	58 lines (ranges: 241-243, 376-392, 394, 397, 399, 402, 405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526-528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_models_coverage.py::TestReportRootToDict::test_to_dict_with_custom_metadata

1ms



AI ASSESSMENT

Scenario: Test to_dict includes custom_metadata when set.**Why Needed:** This test prevents a bug where the custom metadata is not included in the report even though it was provided.**Key Assertions:**

- The 'custom_metadata' key should be present in the result dictionary with the correct values.
- The 'project' value of 'custom_metadata' should match the provided value.
- The 'environment' value of 'custom_metadata' should match the provided value.
- The 'build_number' value of 'custom_metadata' should match the provided value.
- The custom metadata dictionary should not be empty after calling to_dict.
- The report root object passed to to_dict() should have a 'custom_metadata' attribute with the correct values.

Confidence: 80%**Tokens:** 264 input + 167 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/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: Test to_dict includes hmac_signature when set.

Why Needed: HMAC signature is required for the report to be considered valid.

Key Assertions:

- {'assertion_type': 'contains', 'expected_value': 'signature123'}

Confidence: 80%

Tokens: 128 input + 93 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/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: To ensure that the ReportRoot class correctly includes a sha256 value in its dictionary representation.

Key Assertions:

- {'name': 'Expected result', 'value': 'abcdef1234567890'}

Confidence: 80%

Tokens: 131 input + 83 output = 214 total

COVERAGE

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



AI ASSESSMENT

Scenario: Test to_dict includes source_coverage when set.**Why Needed:** Prevents test failure due to missing source coverage information in the report.**Key Assertions:**

- The 'source_coverage' key is present in the result dictionary.
- The value of 'source_coverage' is a list containing exactly one element.
- The first element of 'source_coverage' has the correct 'file_path' attribute.
- The 'file_path' attribute matches the expected value 'src/mod.py'.
- The coverage percentage and ranges are correctly calculated.

Confidence: 80%**Tokens:** 282 input + 126 output = 408 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 because the `to_dict` method of ReportRoot includes warnings when set.

Key Assertions:

- {'name': "len(result['warnings']) == 1", 'expected': 1, 'actual': 0}
- {'name': "result['warnings'][0]['code'] == 'W001'", 'expected': 'W001', 'actual': ''}

Confidence: 80%

Tokens: 151 input + 127 output = 278 total

COVERAGE

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

Why Needed: The test is needed because it checks if the `to_dict` method of `Summary` class correctly returns the total coverage percentage.

Key Assertions:

- {'name': 'coverage_total_percent', 'expected_value': 85.5, 'actual_value': 85.5}

Confidence: 80%

Tokens: 153 input + 116 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/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_witho
ut_coverage_total_percent

1ms



AI ASSESSMENT

Scenario: Test to_dict excludes coverage_total_percent when None.**Why Needed:** The test is needed because it checks the behavior of `summary.to_dict()` when `coverage_total_percent` is `None`. Without this test, the test might fail due to unexpected behavior in other tests.**Key Assertions:**

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

Confidence: 80%**Tokens:** 131 input + 121 output = 252 total

COVERAGE

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

- param_id should be 'a-b-c',
- param_summary should contain 'a=1, b=2, c=3',
- captured_stdout and captured_stderr should match 'stdout content' and 'stderr content' respectively',
- requirements list should include 'REQ-100',
- llm_opt_out should be True,
- llm_context_override should be 'complete',
- coverage length should be 1 (only one entry),

Confidence: 80%**Tokens:** 454 input + 143 output = 597 total

COVERAGE

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

Key Assertions:

- {'name': 'result', 'expected': 'captured_stderr', 'actual': 'Error output here'}

Confidence: 80%

Tokens: 149 input + 87 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	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: to cover the scenario where captured stdout is needed for accurate test results

Key Assertions:

- {'name': 'result', 'expected_value': {'captured_stdout': 'Debug output here'}}

Confidence: 80%

Tokens: 149 input + 83 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/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: This test is necessary because it checks if the `to_dict` method of `TestCaseResult` includes requirements when set.

Key Assertions:

- {'assertion': "result['requirements'] == ['REQ-001', 'REQ-002']", 'expected_value': ['REQ-001', 'REQ-002']}

Confidence: 80%

Tokens: 151 input + 105 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/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 LLMContext.**Why Needed:** Prevents a potential bug where default exclude globs are not correctly identified.**Key Assertions:**

- The function `llm_context_exclude_globs` is called and its return value is checked.
- The string `*.pyc` is found in the returned list of default exclude globs.
- The string `__pycache__/*` is found in the returned list of default exclude globs.
- The string `*secret*` is found in the returned list of default exclude globs.
- The string `*password*` is found in the returned list of default exclude globs.
- The function returns a list with all expected strings.
- The function does not return an empty list when no exclude globs are specified.
- The function does not raise any exceptions when encountering unknown files or directories.

Confidence: 80%**Tokens:** 222 input + 202 output = 424 total

COVERAGE

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

PASSED

tests/test_options.py::TestConfig::test_default_redact_patterns

1ms



AI ASSESSMENT

Scenario: Verifies that default redact patterns include sensitive information.

Why Needed: Prevents a potential security vulnerability where sensitive information like passwords and tokens are not properly redacted.

Key Assertions:

- The pattern '--password' is present in the default redact patterns.
- The pattern '--token' is present in the default redact patterns.
- The pattern '--api[_-]?key' is present in the default redact patterns.

Confidence: 80%

Tokens: 228 input + 105 output = 333 total

COVERAGE

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

PASSED

tests/test_options.py::TestConfig::test_default_values

1ms  3

AI ASSESSMENT

Scenario: Test that default values are set correctly.**Why Needed:** Prevents a regression where the default values of Config 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 + 180 output = 498 total

COVERAGE

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

PASSED

tests/test_options.py::TestConfig::test_get_default_config

1ms



AI ASSESSMENT

Scenario: test_get_default_config

Why Needed: to ensure that the default configuration is created correctly without any external dependencies.

Key Assertions:

- {'name': 'cfg is an instance of Config', 'expected': 'True'}
- {'name': "cfg.provider == 'none'", 'expected': 'True'}

Confidence: 80%**Tokens:** 104 input + 87 output = 191 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: Verifies that the `is_llm_enabled` check returns False for a provider without an LLM.

Why Needed: Prevents regression in case the LLM provider is changed to 'none'.

Key Assertions:

- The function `Config.is_llm_enabled()` should return `False` when the provider is set to 'none'.
- The function `Config.is_llm_enabled()` should return `True` when the provider is set to 'ollama' or 'litellm'.
- The function `Config.is_llm_enabled()` should return `True` when the provider is set to 'gemini'.

Confidence: 80%

Tokens: 263 input + 150 output = 413 total

COVERAGE

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

- {'description': 'The number of errors returned by the validate() method should be 1.', 'expected_value': 1, 'actual_value': 0}
- {'description': "The first error message should contain 'Invalid aggregate_policy 'random'.", 'expected_value': "'Invalid aggregate_policy 'random'", 'actual_value': ''}

Confidence: 80%

Tokens: 128 input + 140 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/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-221, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options.py::TestConfig::test_validate_invalid_context_mode

1ms

3

AI ASSESSMENT

Scenario: test_validate_invalid_context_mode

Why Needed: to test the validation of an invalid context mode

Key Assertions:

- {'assertion_type': 'contains', 'pattern': "Invalid llm_context_mode 'mega_max'", 'expected_value': "Invalid llm_context_mode 'mega_max'”}

Confidence: 80%

Tokens: 131 input + 81 output = 212 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-213, 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: tests/test_options.py::TestConfig::test_validate_invalid_include_phase

Why Needed: To test the validation of an invalid include phase.

Key Assertions:

- {'name': 'Expected error message', 'value': "Invalid include_phase 'lunch_break'")}

Confidence: 80%

Tokens: 129 input + 73 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, 225-229, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options.py::TestConfig::test_validate_invalid_provider

1ms



AI ASSESSMENT

Scenario: tests/test_options.py::TestConfig::test_validate_invalid_provider**Why Needed:** To test the validation of an invalid provider.**Key Assertions:**

- {'assertion_type': 'contains', 'condition': "Invalid provider 'invalid_provider'", 'expected_value': "Invalid provider 'invalid_provider'"}

Confidence: 80%**Tokens:** 122 input + 81 output = 203 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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.**Why Needed:** Prevents regression where the llm_context_bytes is set to a value less than 1000, potentially causing issues with LLM context creation.**Key Assertions:**

- cfg.validate() returns an error message indicating that llm_context_bytes must be at least 1000
- llm_context_bytes in errors contains the expected string 'llm_context_bytes must be at least 1000'
- The number of errors is greater than or equal to 5
- llm_context_bytes must be at least 1000 is found in the list of errors
- llm_max_tests must be 0 (no limit) or positive is found in the list of errors
- llm_requests_per_minute must be at least 1 is found in the list of errors
- llm_timeout_seconds must be at least 1 is found in the list of errors
- llm_max_retries must be 0 or positive is found in the list of errors

Confidence: 80%**Tokens:** 329 input + 228 output = 557 total

COVERAGE

src/pytest_llm_report/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 the config is correctly validated and no errors are returned.**Key Assertions:**

- {'name': 'config is empty', 'expected': [], 'actual': []}

Confidence: 80%**Tokens:** 100 input + 72 output = 172 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 options are not loaded correctly, potentially causing issues with downstream processing.

Key Assertions:

- The `aggregate_dir` attribute of the configuration object is set to `aggr_dir`.
- The `aggregate_policy` attribute of the configuration object is set to `merge`.
- The `aggregate_run_id` attribute of the configuration object is set to `run-123`.
- The `aggregate_group_id` attribute of the configuration object is set to `group-abc`.

Confidence: 80%

Tokens: 295 input + 145 output = 440 total

COVERAGE

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

Why Needed: To ensure that the batch parameter in LLMs is correctly set to False when it's disabled.

Key Assertions:

- {'name': 'cfg.batch_parametrized_tests', 'expected_value': {'scenario': 'False', 'why_needed': 'disabled batch flag works'}}

Confidence: 80%

Tokens: 138 input + 89 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	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 regression where the LLM configuration is not properly loaded due to the absence of pyproject.toml.**Key Assertions:**

- The 'llm_max_retries' attribute in the config should be set to 10 by default.
- The 'llm_provider' attribute in the config should be None.
- The 'llm_model' attribute in the config should be None.
- The 'llm_context_mode' attribute in the config should be None.
- The 'llm_prompt_tier' attribute in the config should be None.
- The 'llm_batch_parametrized' attribute in the config should be False.
- The 'llm_context_compression' attribute in the config should be None.
- The 'llm_aggregate_dir' attribute in the config should not exist.
- The 'llm_coverage_source' attribute in the config should not exist.
- The 'llm_evidence_bundle' attribute in the config should not exist.
- The 'llm_report_html', 'llm_report_json', and 'llm_report_pdf' attributes in the config should be None.

Confidence: 80%**Tokens:** 413 input + 268 output = 681 total

COVERAGE

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

- {'id': 'assert_cfg_llm_coverage_source', 'expected_value': 'cov_dir'}

Confidence: 80%**Tokens:** 126 input + 69 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	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

3ms  3

AI ASSESSMENT

Scenario: tests/test_options.py::TestLoadConfig::test_load_defaults**Why Needed:** To test the default configuration of the Pytest plugin.**Key Assertions:**

- {'name': "cfg.provider == 'none'", 'expected': 'None', 'message': 'Expected cfg.provider to be None'}
- {'name': 'cfg.report_html is None', 'expected': 'None', 'message': 'Expected cfg.report_html to be None'}

Confidence: 80%**Tokens:** 116 input + 116 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	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 should override pyproject.toml options', 'expected_value': {'option1': 'value1', 'option2': 'value2'}, 'actual_value': {}}

Confidence: 80%

Tokens: 134 input + 90 output = 224 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	132 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482-484, 486, 488, 490, 492-494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options.py::TestLoadConfig::test_load_from_cli_provider_override 5ms 3

AI ASSESSMENT

Scenario: Test that CLI provider option overrides pyproject.toml.

Why Needed: To ensure that the correct configuration is loaded based on the provided CLI provider.

Key Assertions:

- {'name': 'pyproject.toml', 'expected_content': """toml\n[tool.pytest.ini]\nversion =\n6.2.0\ndirectories = ['tests', 'docs']\npytest.ini\n""", 'actual_content': ''}

Confidence: 80%

Tokens: 130 input + 112 output = 242 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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': 0}

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)

PASSED

tests/test_options.py::TestLoadConfig::test_load_from_pyproject

5ms  3

AI ASSESSMENT

Scenario: test_load_from_pyproject**Why Needed:** To test the functionality of the `load_config` function in the `LoadConfig` class.**Key Assertions:**

- {'name': 'pyproject.toml file exists and is not empty', 'expected': {'status': 0, 'message': ''}, 'actual': {'status': 0, 'message': ''}}
- {'name': 'pyproject.toml file has the correct structure', 'expected': {'status': 1, 'message': "The pyproject.toml file should have a 'tool' section with the required keys"}, 'actual': {'status': 0, 'message': ''}}

Confidence: 80%**Tokens:** 119 input + 162 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/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 3ms 3

AI ASSESSMENT

Scenario: Test loading token optimization options from CLI.

Why Needed: This test prevents a potential bug where the `llm_prompt_tier` and `batch_parametrized_tests` options are not correctly set for token optimization.

Key Assertions:

- cfg.prompt_tier should be 'minimal'
- cfg.batch_parametrized_tests should be False
- cfg.context_compression should be 'none'

Confidence: 80%

Tokens: 264 input + 97 output = 361 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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: Testing the CLI override for dependency snapshot configuration.

Why Needed: This test prevents a potential bug where the `llm_dependency_snapshot` option is not correctly set to 'deps.json' when running the CLI.

Key Assertions:

- The `report_dependency_snapshot` option in the configuration file should be set to 'deps.json'.
- The value of `llm_dependency_snapshot` in the configuration file should match 'deps.json'.
- The `llm_dependency_snapshot` option should have a valid value.
- The configuration file should contain the correct dependency snapshot path.
- The CLI output should display the expected dependency snapshot path.
- The `report_dependency_snapshot` option should be enabled in the CLI configuration.
- The `llm_dependency_snapshot` option should be set to 'deps.json' in the CLI configuration.

Confidence: 80%

Tokens: 213 input + 187 output = 400 total

COVERAGE

src/pytest_llm_report/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 test CLI override for evidence bundle sets the correct value.

Why Needed: This test prevents a potential regression where the CLI override is not set to the expected evidence bundle.

Key Assertions:

- The `llm_evidence_bundle` option in the configuration file should be set to 'bundle.zip'.
- The `report_evidence_bundle` value in the configuration file should match 'bundle.zip'.

Confidence: 80%

Tokens: 217 input + 101 output = 318 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 case sets the `report_json` option to 'output.json' in the mock configuration.

Why Needed: This test prevents a regression where the `llm_report_json` option is not set correctly for CLI override reports.

Key Assertions:

- The value of `cfg.report_json` should be 'output.json'.
- The `llm_report_json` option in the mock configuration should match the value specified by the test.
- The expected value of `cfg.report_json` is not being set correctly in the mock configuration.
- The `test_cli_report_json` test case does not prevent a regression where the `llm_report_json` option is not set for CLI override reports.
- The `report_json` option in the mock configuration should be updated to 'output.json' after setting it by the test.
- The expected value of `cfg.report_json` is being set correctly in the mock configuration.
- The `test_cli_report_json` test case prevents a regression where the `llm_report_json` option is not set for CLI override reports.

Confidence: 80%

Tokens: 212 input + 246 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/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 `llm_report_pdf` option is correctly overridden to point to a PDF file.

Why Needed: This test prevents a potential bug where the default report location is not updated when CLI overrides are applied.

Key Assertions:

- The value of `llm_report_pdf` in the configuration is set to `output.pdf` after applying the override.
- The expected value of `report_pdf` in the configuration is `output.pdf`.
- No other values for `llm_report_pdf` are present in the configuration.
- The configuration file does not contain any other overrides that would affect the default report location.
- The `llm_report_pdf` option is correctly overridden to point to a PDF file.
- The test passes without raising an assertion error if no override is applied.

Confidence: 80%

Tokens: 212 input + 181 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::TestConfigValidationCoverage::test_validate_invalid_token_output_format

1ms



AI ASSESSMENT

Scenario: test_validate_invalid_token_output_format

Why Needed: To ensure that the token output format is valid and not causing coverage issues.

Key Assertions:

- {'assertion': "litellm_token_output_format should be either 'json', 'xml' or 'yaml'", 'expected_value': ['json', 'xml', 'yaml'], 'message': 'Invalid token output format'}

Confidence: 80%

Tokens: 130 input + 98 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	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



AI ASSESSMENT

Scenario: Test validation when token refresh interval is too short

Why Needed: Token refresh intervals that are too short may cause issues with the application's security and functionality.

Key Assertions:

- {'description': 'The token refresh interval must be at least 60 seconds.', 'expected_value': 60, 'actual_value': 30}

Confidence: 80%

Tokens: 146 input + 88 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/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 3

AI ASSESSMENT

Scenario: Test validation of valid LiteLLM config

Why Needed: To ensure that the LiteLLM configuration is correctly validated and no errors are returned.

Key Assertions:

- {'assertion_type': 'is None', 'expected_value': [], 'actual_value': 0}

Confidence: 80%

Tokens: 142 input + 77 output = 219 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 properly loaded from the pyproject.toml file.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': 'aggregate_include_history = [..., "..."]', 'actual': 'aggregate_include_history = [...]'}
 -

Confidence: 80%**Tokens:** 118 input + 104 output = 222 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 is not empty', 'expected_value': 'True'}
- {'name': 'pyproject.toml contains the correct keys', 'expected_value': {'aggregate_policy': 'some_value'}}}

Confidence: 80%

Tokens: 121 input + 119 output = 240 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the 'all' configuration key is loaded in a single pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml should contain all config keys', 'expected': True, 'actual': 'all'}

Confidence: 80%

Tokens: 120 input + 94 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	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 that the cache directory is loaded correctly from the pyproject.toml file.**Key Assertions:**

- {'name': 'pyproject.toml exists', 'expected': 'True'}
- {'name': 'cache_dir exists in pyproject.toml', 'expected': '/path/to/cache/dir'}

Confidence: 80%**Tokens:** 113 input + 107 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_cache_ttl_seconds 1ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_cache_ttl_seconds

Why Needed: To ensure the cache TTL seconds are correctly loaded from pyproject.toml.

Key Assertions:

- {'name': 'pyproject.toml file exists and is not empty', 'expected_value': 'True'}
- {'name': 'cache_ttl_seconds key exists in pyproject.toml', 'expected_value': 'True'}

Confidence: 80%

Tokens: 116 input + 112 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	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



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_failed_output

Why Needed: To ensure that the `capture_failed_output` option in the `pyproject.toml` file is correctly loaded and used when running tests.

Key Assertions:

- {'name': 'pyproject.toml exists', 'expected': 'pyproject.toml was created successfully'}
- {'name': 'pyproject.toml content', 'expected': 'The pyproject.toml file contains the necessary information to load capture_failed_output.'}

Confidence: 80%

Tokens: 116 input + 133 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	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418-420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_output_max_chars

1ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_capture_output_max_chars

Why Needed: To ensure that the `capture_output_max_chars` option is correctly loaded from the `pyproject.toml` file and used to set the capture output character limit.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': 'max_chars = 1024\\n', 'actual': ''}

Confidence: 80%**Tokens:** 119 input + 108 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	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  3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_bytes

Why Needed: To ensure that the context_bytes from pyproject.toml is correctly loaded and used in tests.

Key Assertions:

Confidence: 80%

Tokens: 113 input + 155 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/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 1ms 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 correctly excluded from coverage reports.

Key Assertions:

- {'name': 'Context exclude globs are not included in coverage reports', 'description': 'The context_exclude_globs setting in pyproject.toml should be excluded from coverage reports.', 'expected_value': False, 'actual_value': 'True'}

Confidence: 80%

Tokens: 119 input + 125 output = 244 total

COVERAGE

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

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'context_file_limit = 100', 'actual': 'context_file_limit = 50'}

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/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



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_context_include_globs

Why Needed: To ensure that the `context_include_globs` setting is correctly loaded from the PyProject.toml file.

Key Assertions:

- {'name': 'Context include globs are being loaded correctly', 'expected_value': [''], 'actual_value': []}

Confidence: 80%

Tokens: 119 input + 103 output = 222 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366-368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_hmac_key_file

1ms



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_hmac_key_file

Why Needed: To ensure that the HMAC key file is loaded correctly and used for signing messages.**Key Assertions:**

- {'expected': 'The HMAC key file should exist in the pyproject.toml file.', 'actual': 'The HMAC key file does not exist or is empty.'}
- {'expected': 'The HMAC key file should be loaded correctly from the pyproject.toml file.', 'actual': 'The HMAC key file is not properly formatted or is missing required information.'}

Confidence: 80%**Tokens:** 118 input + 141 output = 259 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446-447, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_param_values 2ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_param_values

Why Needed: To ensure that the `include_param_values` option is correctly loaded from the `pyproject.toml` file.

Key Assertions:

- {'name': 'The `include_param_values` option should be present in the `pyproject.toml` file.', 'expected_value': True, 'message': 'Expected to find the `include_param_values` option.'}
- {'name': 'The `include_param_values` option should have a valid value.', 'expected_value': 'True', 'message': 'Expected the value to be True.'}

Confidence: 80%

Tokens: 116 input + 158 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	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 properly loaded from the pyproject.toml file.**Key Assertions:**

- {'name': 'pyproject.toml exists and is not empty', 'expected': '/path/to/pyproject.toml', 'actual': 'exists and is not empty'}
- {'name': "pyproject.toml has a 'include' section", 'expected': '/path/to/pyproject.toml', 'actual': "has a 'include' section"}

Confidence: 80%**Tokens:** 113 input + 143 output = 256 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412-413, 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



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_include_pytest_invocation

Why Needed: To ensure that the `include_pytest_invocation` option is correctly loaded from the PyProject.toml file.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': 'contents of pyproject.toml', 'actual': 'contents of pyproject.toml'}

Confidence: 80%**Tokens:** 122 input + 105 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	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: test_load_invocation_redact_patterns**Why Needed:** To ensure that the `invocation_redact_patterns` are correctly loaded from the `pyproject.toml` file.**Key Assertions:**

- {'name': 'pyproject.toml exists and is not empty', 'expected_value': 'True'}
- {'name': 'pyproject.toml contains invocation_redact_patterns key', 'expected_value': 'True'}

Confidence: 80%**Tokens:** 121 input + 110 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	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430-432, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_api_base 2ms 3

AI ASSESSMENT

Scenario: test_load_litellm_api_base**Why Needed:** To ensure that the `litellm_api_base` module is loaded correctly from the PyProject.toml file.**Key Assertions:**

- {'name': 'pyproject.toml contents', 'expected': "The contents of pyproject.toml should contain a section named 'litellm_api_base'."}
- {'name': 'pyproject.toml module exports', 'expected': ['litellm_api_base']}

Confidence: 80%**Tokens:** 122 input + 121 output = 243 total

COVERAGE

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

- {'path': '/path/to/pyproject.toml', 'expected_content': '...', 'actual_content': ''}

Confidence: 80%

Tokens: 122 input + 93 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	110 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-334, 336, 340, 342-344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_token_json_key 2ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_litellm_token_json_key

Why Needed: To ensure that the litellm token JSON key is correctly loaded from the pyproject.toml file.

Key Assertions:

- {'name': 'pyproject.toml exists', 'expected': 'True', 'actual': 'False'}
- {'name': 'pyproject.toml file has correct content', 'expected': "The pyproject.toml file should contain a section named 'tool litellm' with the following key: 'litellm_token_json_key'.", 'actual': 'The pyproject.toml file does not have this section or its key is incorrect.'}

Confidence: 80%

Tokens: 125 input + 170 output = 295 total

COVERAGE

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



AI ASSESSMENT

Scenario: Test loading of `llm_token_output_format` from `pyproject.toml`**Why Needed:** To ensure that the `llm_token_output_format` is correctly loaded and used in the Pytest project.**Key Assertions:**

- {'name': 'File exists', 'expected': 'True', 'actual': 'False'}
- {'name': 'pyproject.toml file created', 'expected': 'True', 'actual': 'False'}

Confidence: 80%**Tokens:** 125 input + 121 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	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

1ms



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': "pyproject.toml exists and has a 'tool' section", 'expected': "The 'tool' section should exist in the pyproject.toml file."}
- {'name': "pyproject.toml has a 'tool.llm.token_refresh_command' entry", 'expected': "The 'tool.llm.token_refresh_command' entry should exist in the pyproject.toml file."}

Confidence: 80%**Tokens:** 125 input + 163 output = 288 total

COVERAGE

src/pytest_llm_report/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 litellm_token_refresh_interval from pyproject.toml**Why Needed:** To ensure that the token is refreshed at the correct interval, we need to verify that the 'litellm_token_refresh_interval' configuration value in the pyproject.toml file matches the expected value.**Key Assertions:**

- {'name': 'Expected value of litellm_token_refresh_interval', 'value': 300, 'expected_type': 'int'}

Confidence: 80%**Tokens:** 125 input + 110 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	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

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/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



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_max_concurrency

Why Needed: To ensure that the `max_concurrency` setting in the `pyproject.toml` file is correctly loaded and used when running tests.**Key Assertions:**

- {'name': 'pyproject.toml exists', 'expected': "The 'pyproject.toml' file was found at the specified path."}
- {'name': 'max_concurrency setting exists in pyproject.toml', 'expected': "The 'max_concurrency' setting in the 'pyproject.toml' file is a valid integer value."}

Confidence: 80%**Tokens:** 116 input + 151 output = 267 total

COVERAGE

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

AI ASSESSMENT

Scenario: Loading max_tests from pyproject.toml**Why Needed:** To ensure that the coverage is loaded correctly when running tests.**Key Assertions:**

- {'name': 'Coverage is loaded from pyproject.toml', 'description': 'The coverage should be loaded from the specified file in the pyproject.toml.'}

Confidence: 80%**Tokens:** 113 input + 83 output = 196 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 and contains all necessary information.**Key Assertions:**

- {'name': 'pyproject.toml exists', 'expected_value': 'True'}
- {'name': 'metadata_file path is correct', 'expected_value': '/path/to/pyproject.toml'}

Confidence: 80%**Tokens:** 113 input + 106 output = 219 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444-446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_ollama_host

1ms



3

AI ASSESSMENT

Scenario: Test PyprojectLoadingCoverage**Why Needed:** To ensure that the ollama_host is loaded correctly from the pyproject.toml file.**Key Assertions:**

- {'name': 'The ollama_host setting should be present in the pyproject.toml file', 'expected_value': 'ollama_host', 'actual_value': 'ollama_host'}
- {'name': 'The ollama_host setting should have a valid value', 'expected_value': "a string value (e.g. 'https://example.com/ollama')", 'actual_value': 'https://example.com/ollama'}

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	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: PyProjectLoadingCoverage

Why Needed: To ensure that the 'omit_tests_from_coverage' feature flag is correctly applied to tests.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'omitting tests from coverage', 'actual': 'omitting tests from coverage'}

Confidence: 80%

Tokens: 121 input + 83 output = 204 total

COVERAGE

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



AI ASSESSMENT

Scenario: Tests for PyProject loading coverage**Why Needed:** To ensure that the `param_value_max_chars` value in `pyproject.toml` is correctly loaded and used.**Key Assertions:**

- {'name': 'File exists', 'expected': 'True', 'actual': 'False'}
- {'name': "File path contains 'max_chars'", 'expected': 'True', 'actual': 'False'}

Confidence: 80%**Tokens:** 119 input + 110 output = 229 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 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 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectLoadingCoverage::test_load_report_collect_only

Why Needed: To ensure that the 'collect' option in 'report Collect Only' is loaded correctly from pyproject.toml.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'Collecting all dependencies', 'actual': 'Collecting only dependencies'}

Confidence: 80%

Tokens: 116 input + 99 output = 215 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 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



AI ASSESSMENT

Scenario: Loading timeout_seconds from pyproject.toml**Why Needed:** To ensure that the timeout seconds are correctly loaded into the Pytest plugin.**Key Assertions:**

- {'name': 'pyproject.toml file exists', 'expected': 'True'}
- {'name': 'timeout_seconds key exists in pyproject.toml', 'expected': 'True'}

Confidence: 80%**Tokens:** 113 input + 96 output = 209 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	109 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384-386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_batch_max_tests

4ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_batch_max_tests

Why Needed: To ensure that the `batch_max_tests` option in the `pyproject.toml` file is correctly loaded and used for token optimization.**Key Assertions:**

- {'name': 'The `batch_max_tests` option should be present in the `pyproject.toml` file', 'value': True}
- {'name': 'The value of `batch_max_tests` should be a number', 'value': 10}

Confidence: 80%**Tokens:** 117 input + 134 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	130 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400-402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_batch_parametrized_tests

5ms



3

AI ASSESSMENT

Scenario: Load batch parametrized tests**Why Needed:** To ensure Pytest coverage is accurate for batch parameterized tests**Key Assertions:**

- {'name': 'Pytest test discovery', 'description': 'The pytest test discovery should be able to find and run the specified tests'}
- {'name': 'Test execution', 'description': 'The tests should execute successfully and produce the expected output'}

Confidence: 80%**Tokens:** 123 input + 105 output = 228 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	131 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396-398, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_context_compression

4ms



3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_context_compression

Why Needed: To ensure that the context compression feature is properly loaded and used in PyProject files.

Key Assertions:

- {'name': 'pyproject.toml file exists', 'expected_value': 'True'}
- {'name': 'context_compression section exists', 'expected_value': 'True'}

Confidence: 80%

Tokens: 117 input + 106 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	130 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402-404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_context_line_padding 5ms 3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestPyprojectTokenOptimization::test_load_context_line_padding

Why Needed: To ensure that the `context_line_padding` option is correctly loaded and applied to the context.

Key Assertions:

- {'name': 'Context line padding is applied correctly', 'expected_value': '', 'actual_value': '...}'}

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/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': 'The `prompt_tier` should be present in the `pyproject.toml` file.', 'expected_value': 'prompt_tier', 'actual_value': 'None'}
- {'name': 'The `prompt_tier` should have the correct type (dict).', 'expected_value': 'dict', 'actual_value': 'None'}

Confidence: 80%**Tokens:** 117 input + 150 output = 267 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	130 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332, 334, 336, 340, 342, 344, 348, 352, 356, 360, 362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392-393, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 502, 504-505, 507, 511-512, 514, 516-517, 519, 521-522, 524, 528-529, 531, 534-535, 537-538, 540, 542-543, 545, 547-548, 550-551, 554-555, 557-558, 561-562, 564, 566-567, 569, 572-573, 575-576, 578, 581-584, 588-589, 591, 593-594, 596, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_batch_max_tests_too_small

1ms

3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_batch_max_tests_too_small

Why Needed: The test is necessary to ensure that the batch_max_tests configuration option has a valid value.

Key Assertions:

- {'message': 'batch_max_tests must be at least 1', 'expected_value': 1}

Confidence: 80%

Tokens: 135 input + 88 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_options_coverage.py::TestValidationCoverageExtended::test_validate_context_line_padding_negative

1ms



AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_context_line_padding

Why Needed: To ensure that the context line padding is not negative and raises an error when it is.

Key Assertions:

- {'assertion_type': 'contains', 'pattern': 'context_line_padding must be 0 or positive', 'value': 'Negative value for context_line_padding'}

Confidence: 80%

Tokens: 129 input + 99 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	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



AI ASSESSMENT

Scenario: test_validate_invalid_context_compression

Why Needed: To ensure that the validation of context compression is not affected by invalid settings.

Key Assertions:

- {'assertion_type': 'contains', 'pattern': 'Invalid context_compression', 'value': "Any error message containing 'Invalid context_compression'"}
 - { 'assertion_type': 'contains', 'pattern': 'Invalid context_compression', 'value': "Any error message containing 'Invalid context_compression'"}

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/options.py	29 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-269, 271, 273, 276)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_invalid_prompt_tier

1ms

3

AI ASSESSMENT

Scenario:

tests/test_options_coverage.py::TestValidationCoverageExtended::test_validate_invalid_prompt_tier

Why Needed: To ensure that the validation of invalid `prompt_tier` values does not return any error messages.

Key Assertions:

- {'type': 'assertion', 'message': 'Invalid prompt_tier', 'expected': ['Invalid prompt_tier']}

Confidence: 80%

Tokens: 125 input + 93 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	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 config defaults are correct and safe.**Key Assertions:**

- {'name': 'cfg' is an instance of Config', 'expected_type': 'Config'}

Confidence: 80%**Tokens:** 119 input + 73 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/options.py	124 lines (ranges: 123, 171, 308, 311-312, 320-325, 327-328, 332-337, 340, 342, 344, 348, 352, 356, 360-362, 364, 366, 368, 372, 374, 378-380, 382, 384-386, 388, 390, 392, 396, 400, 402, 404, 408-410, 412-413, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460-461, 463-464, 466-467, 470, 472-473, 476-477, 482-488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603-605, 607, 611, 613)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_plugin_integration.py::TestPluginConfigLoading::test_markers_exist_in_config

1ms



AI ASSESSMENT

Scenario:

tests/test_plugin_integration.py::TestPluginConfigLoading::test_markers_exist_in_config

Why Needed: This test ensures that markers are present in the plugin configuration.

Key Assertions:

- {'name': 'pytestconfig is not None', 'expected_value': 'True'}

Confidence: 80%

Tokens: 108 input + 75 output = 183 total

COVERAGE

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

PASSED

tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_both_json_and_html_outputs 89ms 8

AI ASSESSMENT

Scenario: Verify that the test generates both JSON and HTML reports for Pytest.**Why Needed:** This test prevents a regression where the report generation is only done in JSON format, but not in HTML.**Key Assertions:**

- The `report.json` file should exist at the specified path.
- The `report.html` file should exist at the specified path.
- Both files should be present in the test directory.
- The report generation should work correctly even when run with different output formats (JSON and HTML).
- The `pytester.runpytest()` call should successfully generate both JSON and HTML reports.

Confidence: 80%**Tokens:** 279 input + 140 output = 419 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 57ms 7

AI ASSESSMENT

Scenario:

tests/test_plugin_integration.py::TestPluginHooksWithPytester::test_collection_finish_counts_items

Why Needed: pytest_collection_finish counts items (line 378)

Key Assertions:

- {'name': "assert data['run_meta']['collected_count'] == 3", 'expected_value': 3}

Confidence: 80%

Tokens: 198 input + 83 output = 281 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 53ms 7

AI ASSESSMENT

Scenario: Test that output directories are created if missing.**Why Needed:** Prevents regression where the plugin does not create an output directory.**Key Assertions:**

- The `nested` subdirectory should be present in the report.json file.
- The `report.json` file should exist in the specified location.
- The `makepyfile` command should create the necessary directories and files.

Confidence: 80%**Tokens:** 247 input + 94 output = 341 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

59ms



7

AI ASSESSMENT

Scenario: Test that fixture errors are captured in report.

Why Needed: Fixture failures are not properly reported and can lead to false positives or missed issues.

Key Assertions:

- The 'summary' key in the report contains an error code of 1, indicating a failure.
- The 'error' key in the 'summary' dictionary contains the string 'RuntimeError: Fixture failed'.
- The test passes even though the fixture failed, due to the captured error being reported as a false positive.

Confidence: 80%

Tokens: 286 input + 117 output = 403 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

143ms



7

AI ASSESSMENT

Scenario: Test pytest_runttest_makereport captures all outcomes**Why Needed:** pytest_runttest_makereport may not capture all outcomes if the test is skipped or fails, leading to incorrect reporting.**Key Assertions:**

- The test verifies that the report includes 'passed', 'failed', and 'skipped' outcomes.
- The test asserts that there are at least three types of outcomes in the report (passed, failed, and skipped).
- The test checks if the report contains all three types of outcomes.
- The test ensures that the report does not exclude any outcome by checking for missing values.
- The test verifies that the report includes a mix of passing and failing tests.
- The test asserts that the report correctly identifies which tests were skipped.
- The test checks if the report accurately reflects the number of passed, failed, and skipped tests.

Confidence: 80%**Tokens:** 335 input + 195 output = 530 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)

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 52ms 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': 'exists', 'expected_result': True}

Confidence: 80%

Tokens: 150 input + 76 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	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

554ms



8

AI ASSESSMENT

Scenario: Test that --llm-pdf option enables the plugin.**Why Needed:** Prevents regression where the plugin is not enabled due to missing Playwright configuration.**Key Assertions:**

- The test verifies that `pytester.runpytest('--llm-pdf=report.pdf')` exits with code 0 (success) or warning, but definitely runs.
- The test checks if collection/hooks run successfully after enabling the plugin via --llm-pdf.
- The test verifies that passing only --llm-pdf works to trigger the plugin logic and does not say 'no reports configured'.
- The test ensures that the plugin is enabled correctly even when logging a report file name like 'report.pdf'.

Confidence: 80%**Tokens:** 435 input + 157 output = 592 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

56ms



7

AI ASSESSMENT

Scenario: Test that the pytest_sessionstart records start time is correctly reported in the report.json file.

Why Needed: This test prevents a regression where the start time of the session might not be accurately reported in the report.json file.

Key Assertions:

- The 'start_time' key should be present in the run_meta dictionary within the 'report.json' file.
- The value of the 'start_time' key should match the actual start time of the pytest session.
- If no start time is recorded, the 'start_time' key should still be present and have a default value (e.g., 0)
- Any changes to the start time should be reflected in the report.json file immediately after running pytest
- The 'start_time' value should not be affected by any external factors that might impact the session timing (e.g., system clock, network latency)
- If multiple sessions are run concurrently, the start times should be correctly reported and consistent across all sessions

Confidence: 80%

Tokens: 276 input + 216 output = 492 total

COVERAGE

src/pytest_llm_report/collector.py	39 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	75 lines (ranges: 190, 194-199, 201, 203, 205, 207, 210, 212, 214, 216, 218, 220, 222, 224, 376-392, 394, 397, 399, 402-405, 407, 409, 411, 413, 415, 419, 437, 467-475, 477, 479, 518, 520-524, 526, 528, 530, 532, 534, 536, 538, 540)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-

258, 265-266, 271, 273, 276,
284, 308, 311-312, 320-322,
460, 463, 466, 470, 472-473,
476-477, 482, 484-486, 488,
490, 492, 494, 499-500, 504-
505, 511-512, 516-517, 521-
522, 528-529, 534, 537-538,
542-543, 547-548, 554-555,
561-562, 566-567, 572, 575-
576, 581, 583, 588-589, 593-
594, 599, 601, 603, 605, 607,
611, 613)

src/pytest_llm_report/plugin.py

288 lines (ranges: 41, 44-48,
50-54, 56-60, 62-66, 68-72,
74-79, 81-86, 90-94, 96-100,
102-106, 108-112, 114-118,
122-126, 128-132, 134-138,
142-146, 148-153, 155-159,
161-165, 169-174, 176-181,
185-190, 192-197, 199-204,
208-213, 215-219, 223-227,
229-233, 235-239, 241-245,
247-252, 254-258, 260-264,
268-272, 274-279, 283-287,
289-293, 297-302, 304-309,
311-315, 328-330, 332-334,
336-338, 342, 346-347, 349,
351, 354-355, 362, 371-373,
399, 403, 407, 410, 429-430,
437-438, 441-442, 444-445,
448-452, 454, 457-458, 460,
463-464, 485, 491-492, 534-
544, 558-559, 562, 566-568,
579, 583, 602, 606-608, 619,
623, 626, 628-629)

src/pytest_llm_report/report_writer.py

110 lines (ranges: 55, 67-73,
85-86, 98-100, 127-128, 130,
156-158, 186, 192-193, 197-
198, 202, 211-218, 222-223,
226, 230, 233, 254, 256-259,
262-264, 266, 268-275, 277-
278, 280-289, 291-294, 296-
297, 299-300, 302-303, 305-
307, 319, 321-322, 324-325,
337, 347, 350-352, 355-356,
359-361, 364, 367-371, 477-
478, 502, 504, 506-508, 510,
513)

PASSED

tests/test_plugin_integration.py::TestPluginIntegration::test_llm_content_marker

1ms  2

COVERAGE

src/pytest_llm_report/collector.py

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

src/pytest_llm_report/plugin.py

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

PASSED

tests/test_plugin_integration.py::TestPluginIntegration::test_llm_output_out_marker

1ms  2

COVERAGE

src/pytest_llm_report/collector.py

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

src/pytest_llm_report/plugin.py

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

PASSED

tests/test_plugin_integration.py::TestPluginIntegration::test_requirement_marker

1ms  2

AI ASSESSMENT

Scenario: tests/test_plugin_integration.py::TestPluginIntegration::test_requirement_marker

Why Needed: This test is necessary to ensure that the requirement marker does not cause any errors.

Key Assertions:

- {'assertion_type': 'does_not_raise', 'expected_output': [], 'actual_output': []}

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)

PASSED

tests/test_plugin_integration.py::TestReportGeneration::test_report_writer_integration 39ms 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 with pytest_llm_report.**Key Assertions:**

- Verify that a full report is generated and saved to the specified JSON file.
- Verify that the report includes summary data with total count of tests, passed, and failed.
- Verify that the HTML report includes references to all test files (test_a.py and test_b.py).

Confidence: 80%**Tokens:** 417 input + 113 output = 530 total

COVERAGE

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

 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:

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_enabled

Why Needed: The `pytest_collectreport` plugin is enabled.**Key Assertions:**

- {'name': 'mock_collector.handle_collection_report', 'expected_calls': [1]}

Confidence: 80%**Tokens:** 204 input + 76 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/plugin.py

12 lines (ranges: 558-559,
562, 566-568, 579-580, 586-
590-592)

PASSED

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_no_session

1ms

2

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_no_session

Why Needed: To ensure that collectreport skips when session is not available.

Key Assertions:

- {'message': 'pytest_collectreport should be called with a session object', 'expected_result': 'No session attribute'}

Confidence: 80%

Tokens: 138 input + 85 output = 223 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	8 lines (ranges: 558-559, 562, 566-568, 579, 583)

PASSED

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_session_none

1ms

2

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_session_none

Why Needed: To ensure that the collectreport plugin behaves correctly when a Pytest session is None.

Key Assertions:

- {'name': 'mock_report.session is None', 'expected': 'None'}

Confidence: 80%

Tokens: 134 input + 82 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	8 lines (ranges: 558-559, 562, 566-568, 579, 583)

PASSED

tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_llm_enabled_warning 3ms 3

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_llm_enabled_warning

Why Needed: LLM enabled warning is raised when pytest is run with the --llm flag.

Key Assertions:

- {'name': 'LLM enabled warning should be raised', 'expected_value': True, 'actual_value': 'False'}

Confidence: 80%

Tokens: 143 input + 92 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	136 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-325, 327-328, 332-336, 340, 342, 344, 348, 352, 356, 360-362, 364, 366, 368, 372, 374, 378, 380, 382, 384, 386, 388, 390, 392, 396, 400, 402, 404, 408, 412, 416, 418, 420, 426, 430, 436, 438, 444, 446, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	30 lines (ranges: 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362-364, 366-367, 371-373, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_validation_errors 3ms 3

AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_validation_errors

Why Needed: Validation errors are raised when the plugin is used with invalid configuration files.**Key Assertions:**

- {'name': 'Expected a UsageError to be raised', 'message': 'A UsageError should be raised when the plugin is used with an invalid pyproject.toml file.'}

Confidence: 80%**Tokens:** 134 input + 97 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	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:

- {'name': 'mock_config.addinvalue_line.called', 'expected_value': 1}

Confidence: 80%

Tokens: 170 input + 81 output = 251 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	17 lines (ranges: 328-330, 332-334, 336-338, 342-343, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginConfigureFallback::test_pyte
st_configure_fallback_load

3ms 2

AI ASSESSMENT

Scenario: Test fallback to load_config if Config.load is missing.

Why Needed: This test prevents a potential bug where the plugin would attempt to load configuration files without calling Config.load, potentially leading to unexpected behavior or errors.

Key Assertions:

- mock_load.return_value.mock_cfg == mock_cfg
- # Test that load_config returns the correct mock config object

Confidence: 80%

Tokens: 747 input + 230 output = 977 total

COVERAGE

src/pytest_llm_report/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 ensure that CLI options override pyproject.toml options when loading plugin configurations.

Key Assertions:

- {'name': 'pyproject.toml contents', 'expected': 'pyproject.toml contents'}
- {'name': 'load_config function call', 'expected': 'load_config function call'}

Confidence: 80%

Tokens: 140 input + 110 output = 250 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 97ms 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 file.

Key Assertions:

- {'name': 'pyproject.toml exists', 'expected': 'True'}
- {'name': 'pyproject.toml is readable', 'expected': 'True'}

Confidence: 80%

Tokens: 136 input + 101 output = 237 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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: Prevents a potential regression where the plugin's terminal summary functionality is not properly handled when it is disabled.

Key Assertions:

- Mocked stash.get() with _enabled_key and False returns None, as expected.
- Mocked stash.get() with _enabled_key and True does not return None, but instead calls test_terminal_summary() with the provided arguments.
- Test_terminal_summary() is called with the provided arguments (0, mock_config) when stash.get(_enabled_key, False) returns False.

Confidence: 80%

Tokens: 281 input + 133 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/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



AI ASSESSMENT

Scenario:

tests/test_plugin_maximal.py::TestPluginMaximal::test_terminal_summary_worker_skip

Why Needed: The test is necessary because it checks for the correct behavior of the `pytest_terminal_summary` plugin when skipping terminal summary workers.

Key Assertions:

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

Confidence: 80%

Tokens: 164 input + 86 output = 250 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	8 lines (ranges: 399-400, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginMaximal::testload_config

5ms



AI ASSESSMENT

Scenario: Test config loading from pytest objects (CLI) for maximal plugin functionality.

Why Needed: This test prevents a potential issue where the llm_report option is not set correctly, potentially leading to incorrect configuration or errors during execution.

Key Assertions:

- mock_config.option.llm_report_html == 'out.html'
- mock_config.option.llm_report_json == None
- mock_config.option.llm_report_pdf == None
- mock_config.option.llm_evidence_bundle == None
- mock_config.option.llm_dependency_snapshot == None
- mock_config.option.llm_requests_per_minute == None
- mock_config.option.llm_aggregate_dir == None
- mock_config.option.llm_aggregate_policy == None
- mock_config.option.llm_aggregate_run_id == None
- mock_config.option.llm_aggregate_group_id == None
- mock_config.option.llm_max_retries == None
- mock_config.option.llm_coverage_source == None
- mock_config.option.llm_prompt_tier == None
- mock_config.option.llm_batch_parametrized == None
- mock_config.option.llm_context_compression == None
- mock_config.option.llm_context_bytes == None
- mock_config.option.llm_context_file_limit == None
- mock_config.option.llm_max_tests == None
- mock_config.option.llm_max_concurrency == None
- mock_config.option.llm_timeout_seconds == None
- mock_config.option.llm_capture_failed == None
- mock_config.option.llm_ollama_host == None
- mock_config.option.llm_llm_api_base == None
- mock_config.option.llm_llm_api_key == None
- mock_config.option.llm_llm_token_refresh_command == None
- mock_config.option.llm_llm_token_refresh_interval == None
- mock_config.option.llm_llm_token_output_format == None
- mock_config.option.llm_llm_token_json_key == None
- mock_config.option.llm_cache_dir == None
- mock_config.option.llm_cache_ttl == None

Confidence: 80%

Tokens: 639 input + 466 output = 1105 total

COVERAGE

src/pytest_llm_report/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 verify that makereport skips when disabled.

Key Assertions:

- {'name': 'mock_item.config.stash.get.return_value', 'expected_value': 'False'}

Confidence: 80%

Tokens: 220 input + 84 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/plugin.py	7 lines (ranges: 558-559, 562-563, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginRuntest::test_runtest_makereport_enabled

2ms



2

AI ASSESSMENT

Scenario: Test makereport calls collector when enabled.**Why Needed:** This test prevents a potential regression where the plugin does not call the collector when makereport is enabled.**Key Assertions:**

- The `pytest_runtest_makereport` function should be called with the correct mock collector instance.
- The `mock_collector.handle_runtest_logreport` method should be called with the provided mock report and item.
- The `stash_get` method of the mock item should return True when it is set to `'_enabled_key'` or `'_collector_key'`.
- The `mock_item.config.stash.get` method should not call the collector instance directly.
- The `pytest_runtest_makereport` function should yield a point after calling `send` on the mock outcome.
- The `handle_runtest_logreport` method of the mock collector instance should be called with the provided mock report and item.

Confidence: 80%**Tokens:** 371 input + 205 output = 576 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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: To ensure that the plugin correctly handles collection_finish when disabled.

Key Assertions:

- {'name': 'mock_session.config.stash.get.return_value', 'expected': {'False': {}}}

Confidence: 80%

Tokens: 149 input + 81 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/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: Test PluginSessionHooks

Why Needed: To test if the `pytest_collection_finish` function calls the `_collector_key` collector when collection_finish is enabled.

Key Assertions:

- {'name': 'mock_collector.handle_collection_finish' was called once with mock_session.items', 'expected': 1, 'actual': 0}

Confidence: 80%

Tokens: 219 input + 88 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	10 lines (ranges: 558-559, 562, 566-568, 602, 606-608)

PASSED

tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_sessionstart_disabled

1ms  2

COVERAGE

src/pytest_llm_report/collector.py

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

src/pytest_llm_report/plugin.py

8 lines (ranges: 558-559, 562, 566-568, 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.

Why Needed: Prevents a potential bug where the collector is not created 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 + 77 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

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 and verifies specific options.**Why Needed:** pytest_addoption may not add all required arguments if the plugin is not properly configured.**Key Assertions:**

- Verify that `--llm-report` and `--llm-coverage-source` are included in the command line arguments.
- Check that both options are present in the `args[0]` of each call to `addoption`.
- Verify that the correct group name is used for adding options (in this case, `llm-report` and `LLM-enhanced test reports`).

Confidence: 80%**Tokens:** 293 input + 137 output = 430 total

COVERAGE

src/pytest_llm_report/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

Key Assertions:

- {'name': 'parser.addini was not called', 'expected_result': 0, 'actual_result': 1}

Confidence: 80%

Tokens: 140 input + 85 output = 225 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	220 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_termin
al_summary_coverage_calculation

3ms



AI ASSESSMENT

Scenario: Test coverage percentage calculation logic for terminal summary.**Why Needed:** Prevents regression in coverage reporting when terminal summary is enabled and coverage map is loaded.**Key Assertions:**

- The `report_html` option is set to 'out.html' and the test asserts that the Coverage class correctly reports a coverage percentage of 85.5.
- The `stash` dictionary contains `_enabled_key` as True and `_config_key` with the Config object, allowing the test to verify the correct stash configuration.
- The mock Coverage class is created with the loaded CoverageMapper instance, ensuring that the coverage report is generated correctly.
- The `report` method of the mock Coverage class is called once, verifying that it reports a coverage percentage of 85.5.
- The `load` method of the mock Coverage class is called once, verifying that it loads the coverage map correctly.
- The `report_html` option is set to 'out.html', allowing the test to verify that the report writer writes the correct HTML file.
- The stash dictionary contains the required keys for the Config object, ensuring that the stash configuration is valid.
- The mock stash instance is created with the specified stash data, allowing the test to verify its correctness.

Confidence: 80%**Tokens:** 395 input + 269 output = 664 total

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 277, 285)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	53 lines (ranges: 399, 403, 407, 410, 429-430, 432, 434, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-466, 468, 470-473, 485-486, 491-492, 534-544, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_termin
al_summary_llm_enabled

3ms



AI ASSESSMENT

Scenario: Test terminal summary with LLM enabled runs annotations.**Why Needed:** Prevents regression in terminal summary functionality when LLM is enabled.**Key Assertions:**

- Verify that the test passes when `llm_enabled` is set to True.
- Check if the correct configuration is passed to `pytest_terminal_summary`.
- Verify that the `llm_enabled` key is correctly added to the stash dictionary.
- Ensure that the `get_provider` method returns the correct provider instance.
- Verify that the `annotate_tests` method is called with the correct arguments.
- Check if the coverage map is properly updated when LLM is enabled.
- Verify that the report writer is able to write to the out.html file correctly.

Confidence: 80%**Tokens:** 477 input + 166 output = 643 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 bug where the plugin does not create a collector even when it is supposed to be missing.

Key Assertions:

- mock_terminalreporter.call_args[1][0].stash._enabled_key should return True
- mock_terminalreporter.call_args[1][0].stash._config_key should return cfg
- mock_config.stash._enabled_key should return False

Confidence: 80%

Tokens: 391 input + 108 output = 499 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 `aggregate_dir` is set to a non-existent directory.**Key Assertions:**

- The stash created by `pytest_terminal_summary` should support both get() and [] indexing.
- The aggregator returned by `Aggregator` should be called once with the correct arguments.
- The report writer should write JSON and HTML files correctly.
- The aggregate function should return a report object.
- The stash should not have any invalid keys when set to an empty stash.
- The aggregation function should raise an error if `aggregate_dir` is not a valid directory path.
- The terminal reporter should be patched with the correct class and arguments.
- The terminal writer should write JSON and HTML files correctly.

Confidence: 80%**Tokens:** 441 input + 179 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/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	21 lines (ranges: 399, 403, 407, 410-411, 413-414, 417-418, 420, 422-426, 558-559, 562, 566-568)

PASSED

tests/test_plugin_maximal.py::TestPluginTerminalSummaryErrors::test_terminal_summary_coverage_error 5ms 3

AI ASSESSMENT

Scenario: Test coverage calculation error when load fails during computation.

Why Needed: This test prevents a regression where the coverage calculation fails due to an OSError during load.

Key Assertions:

- The function `pytest_terminal_summary` should not raise a UserWarning for a valid configuration.
- The `load` method of `CoverageMapper` should be called with a valid path.
- The `report_writer` should write the coverage report without raising an exception.
- The `coverage.Coverage` object should have a valid load method.
- The `MagicMock()` instance passed to `pytest_terminal_summary` should not raise a UserWarning.
- A valid configuration with a valid path should be loaded successfully.
- The coverage map should be created without raising an exception.
- The report writer should write the coverage report without raising an exception.

Confidence: 80%

Tokens: 389 input + 188 output = 577 total

COVERAGE

src/pytest_llm_report/collector.py	16 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210, 277, 285)
src/pytest_llm_report/options.py	3 lines (ranges: 123, 171, 284)
src/pytest_llm_report/plugin.py	52 lines (ranges: 399, 403, 407, 410, 429-430, 432, 434, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-466, 476-479, 485-486, 491-492, 534-544, 558-559, 562, 566-568)

 tests/test_prompts.py

7 tests

PASSED

tests/test_prompts.py::TestContextAssembler::test_assemble_balanced_context 6ms 5

AI ASSESSMENT

Scenario: Test the ContextAssembler to assemble a balanced context with a test file and dependency.

Why Needed: The test prevents regression when assembling contexts with unbalanced dependencies, ensuring that only necessary code is included.

Key Assertions:

- The 'utils.py' file should be present in the assembled context.
- A function named `util()` should be found within the 'utils.py' file.
- Only the `def util()` line from 'utils.py' should be included in the assembled context.
- No additional code should be included in the assembled context beyond what is necessary for the test.
- The 'utils.py' file should not contain any other functions or variables that are not used by `util()`
- The 'utils.py' file should have a line count of 2, as specified in the coverage report

Confidence: 80%

Tokens: 331 input + 185 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/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



AI ASSESSMENT

Scenario: Test the ContextAssembler to assemble a complete context for a test file

Why Needed: To ensure that the ContextAssembler can correctly assemble a complete context for a test file, including all necessary key assertions.

Key Assertions:

- {'assertion_type': 'ContextAssembled', 'context_assembled': {'test_1': {}}}

Confidence: 80%

Tokens: 176 input + 88 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	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	38 lines (ranges: 33, 49, 52, 55, 58, 60, 63, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 116, 139-140, 268-272)

PASSED

tests/test_prompts.py::TestContextAssembler::test_assemble_minimal_context

1ms



4

AI ASSESSMENT

Scenario: Verifies that the ContextAssembler can assemble a minimal context for a test file with a single test function.

Why Needed: This test prevents regression when using the minimal context mode, as it ensures that all necessary code is included in the assembly.

Key Assertions:

- The 'test_1' function should be present in the source code of the assembled test file.
- The ContextAssembler should return an empty dictionary for the context of the assembled test file.
- The test result nodeid should match the path of the test file being tested.
- Any additional imports or dependencies required by the test function should be included in the assembly.
- No other code should be present in the source code of the assembled test file, except for the 'test_1' function.
- The ContextAssembler should return an empty dictionary when no tests are found in the test file.
- Any error messages or warnings from the assembler should not affect the correctness of the assembly.

Confidence: 80%

Tokens: 267 input + 213 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/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: Verifies that the ContextAssembler correctly limits LLM context size to prevent truncation of long content.

Why Needed: This test prevents a potential bug where the ContextAssembler would truncate long code snippets, potentially leading to incorrect results or errors.

Key Assertions:

- The 'f1.py' file is present in the assembled context.
- The 'f1.py' file contains the truncated content.
- The length of the 'f1.py' file does not exceed 40 bytes (20 + truncation message).

Confidence: 80%

Tokens: 335 input + 124 output = 459 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	46 lines (ranges: 33, 49, 52, 55, 58, 60-61, 65, 78-79, 82-84, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-194, 196-197, 201, 284-285, 287)

PASSED

tests/test_prompts.py::TestContextAssembler::test_complete_context_limits_override

1ms



5

AI ASSESSMENT

Scenario: Test that 'complete' mode does not truncate long files when context bytes are set to a small value.

Why Needed: This test prevents the truncation of long files, which can lead to incorrect results or data loss in certain contexts.

Key Assertions:

- The 'f1.py' file is present in the assembled context.
- The content of 'f1.py' matches the original content.
- Context for 'f1.py' does not contain any 'truncated' keyword.
- Context for 'f1.py' has a length equal to its original content.

Confidence: 80%

Tokens: 361 input + 135 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/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	50 lines (ranges: 33, 49, 52, 55, 58, 60, 63, 65, 78-79, 82-84, 139, 142-145, 147-148, 152-153, 155-156, 159-160, 163, 166-167, 170-171, 173-174, 177, 181-182, 189, 192-193, 196-197, 201, 268-272, 284-285, 287)

PASSED

tests/test_prompts.py::TestContextAssembler::test_get_test_source_ed
ge_cases

1ms



AI ASSESSMENT

Scenario: Verify the ContextAssembler's ability to handle non-existent file names and nested test names with parameters.

Why Needed: This test prevents a potential bug where the ContextAssembler incorrectly handles files that do not exist or have incorrect paths, leading to unexpected behavior in subsequent tests.

Key Assertions:

- The function `'_get_test_source'` returns an empty string when given a non-existent file name.
- The function `'_get_test_source'` correctly includes the nested test name with parameters in its source code output.
- The function `'_get_test_source'` raises an `AssertionError` when given a file that does not exist or has incorrect path.

Confidence: 80%

Tokens: 275 input + 141 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/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 by the LLM.

Why Needed: This test prevents a potential bug where the ContextAssembler incorrectly excludes files that are not actually protected by the llm_context_exclude_globs configuration.

Key Assertions:

- asserts that 'test.pyc' is excluded because it has a .pyc extension
- asserts that 'secret/key.txt' is excluded because it does not match any globs in the config
- asserts that 'public/readme.md' is included because it matches one of the globs in the config

Confidence: 80%

Tokens: 227 input + 140 output = 367 total

COVERAGE

src/pytest_llm_report/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 return any context files when run in minimal mode.**Key Assertions:**

- context_files is an empty list.
- def test_foo() is present in test_source.
- test_file::test_foo is present in the test source.
- test_foo.py is present in the test file.
- Context files are not generated when assemble function returns no context files.

Confidence: 80%**Tokens:** 298 input + 115 output = 413 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 where the assembly of a test with an overridden LLM context is not using the expected balanced mode.**Key Assertions:**

- The assembler should use balanced mode when the LLM context override is set to 'balanced'.
- The coverage entry for module.py should include the line ranges and count of 1.
- The test source file (test_bar.py) should be included in the context files.
- The context files should contain the modified module file (module.py).
- The LLM context override should be respected when assembling a test with an overridden context.
- The assembler should report that it is using balanced mode for the assembly of the test.
- The coverage entry should include line ranges and count 1, indicating that only one line was executed in module.py.

Confidence: 80%**Tokens:** 362 input + 196 output = 558 total

COVERAGE

src/pytest_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 that the balanced context excludes files matching exclude patterns.

Why Needed: This test prevents a regression where the LLM context mode is set to 'balanced' and it includes files with glob patterns like '*secret*', which are excluded by default.

Key Assertions:

- The file `secret_config.py` should not be included in the balanced context.
- No files matching exclude pattern `*secret*` should be included in the balanced context.
- The LLM context mode 'balanced' should exclude files with glob patterns like '*secret*' from being included in the context.
- The file paths of excluded files should match the exclude patterns exactly (case-sensitive).
- No files with glob patterns like '*secret*' should be found in the directory tree of `tmp_path`.
- The LLM context mode 'balanced' should not include any files that are not part of a test or an existing file.
- The balanced context should exclude all files that match the exclude patterns, including subdirectories and nested directories.

Confidence: 80%

Tokens: 331 input + 225 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/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 that the ContextAssembler correctly handles a scenario where a balanced context file does not exist.

Key Assertions:

- {'name': 'context is empty', 'expected_result': {}, 'actual_result': {}}

Confidence: 80%

Tokens: 201 input + 88 output = 289 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 by ensuring the content is truncated when exceeding the maximum bytes limit.**Key Assertions:**

- The length of the content in the source file does not exceed 120 bytes.
- The 'truncated' keyword is present in the content if it exceeds the maximum bytes limit.
- The content is truncated by removing some buffer for truncation messages.

Confidence: 80%**Tokens:** 405 input + 106 output = 511 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/context_util.py	17 lines (ranges: 27, 29, 33, 35-36, 64, 66-69, 108, 124, 126-127, 129, 133, 135)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	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 test the balanced context with no coverage returns an empty dict.

Key Assertions:

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

Confidence: 80%

Tokens: 162 input + 82 output = 244 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	3 lines (ranges: 33, 139-140)

PASSED

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_balanced_context_reaches_max_bytes_before_file

1ms



AI ASSESSMENT

Scenario: Test that loop exits when max bytes is reached before processing file.

Why Needed: Prevents a potential bug where the ContextAssembler exceeds the maximum allowed bytes in a balanced context without reaching the end of any file.

Key Assertions:

- context should have only one node (either 'file1.py' or 'file2.py')
- context should not exceed max_bytes=5 when both files are processed
- context should be empty if neither file is processed
- context should contain the first file only (truncated) if it's the only file in the context
- context should not contain any other nodes if no files are processed

Confidence: 80%

Tokens: 409 input + 150 output = 559 total

COVERAGE

src/pytest_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: Tests for _get_test_source with empty nodeid

Why Needed: To ensure that the function returns an empty string when given an empty nodeid.

Key Assertions:

- {'name': 'Result is an empty string', 'expected_value': '', 'actual_value': 'result'}

Confidence: 80%

Tokens: 148 input + 79 output = 227 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	9 lines (ranges: 33, 78-79, 82-83, 86-89)

PASSED

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_extraction_stops_at_next_def 1ms 4

AI ASSESSMENT

Scenario:

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_extraction_stops_at_next_def

Why Needed: To ensure that source extraction stops at the next function definition, as intended.

Key Assertions:

- {'assertion': 'The test file contains a def statement immediately after the test function', 'expected_result': 'test_stop.py'}

Confidence: 80%

Tokens: 129 input + 95 output = 224 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	25 lines (ranges: 33, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 114, 116)

PASSED

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_file_not_exists 1ms 4

AI ASSESSMENT

Scenario: tests/test_prompts_coverage.py

Why Needed: To test the edge case where the _get_test_source method returns an empty string when a non-existent file is provided.

Key Assertions:

- {'name': 'assert result is not None', 'expected_value': '', 'message': 'Expected result to be an empty string'}

Confidence: 80%

Tokens: 142 input + 88 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	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	6 lines (ranges: 33, 78-79, 82-84)

PASSED

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_with_class 1ms 4

AI ASSESSMENT

Scenario:

tests/test_prompts_coverage.py::TestContextAssemblerEdgeCases::test_get_test_source_with_class

Why Needed: To ensure that the `_get_test_source` function correctly extracts functions with proper indentation, even when they are nested within other functions or blocks of code.

Key Assertions:

- `{'expected': 'The extracted function should have correct indentation (2 spaces per level).', 'actual': 'The extracted function has incorrect indentation (1 space per level).}'`

Confidence: 80%

Tokens: 118 input + 113 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	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/prompts.py	25 lines (ranges: 33, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 114, 116)

 tests/test_ranges.py

13 tests

PASSED

tests/test_ranges.py::TestCompressRanges::test_consecutive_lines

1ms  3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_consecutive_lines**Why Needed:** To ensure that consecutive lines are compressed correctly.**Key Assertions:**

- {'expected': '1-3', 'actual': '1-3'}

Confidence: 80%**Tokens:** 106 input + 69 output = 175 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	12 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65, 67)

PASSED

tests/test_ranges.py::TestCompressRanges::test_duplicates

1ms  3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_duplicates**Why Needed:** To test the handling of duplicate ranges in the compress_ranges function.**Key Assertions:**

- {'expected': '1-3', 'actual': '1-3'}

Confidence: 80%**Tokens:** 107 input + 71 output = 178 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	12 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65, 67)

PASSED

tests/test_ranges.py::TestCompressRanges::test_empty_list

1ms



AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_empty_list**Why Needed:** The test is necessary because the function `compress_ranges` does not handle an empty input correctly.**Key Assertions:**

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

Confidence: 80%**Tokens:** 92 input + 72 output = 164 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	2 lines (ranges: 29-30)

PASSED

tests/test_ranges.py::TestCompressRanges::test_mixed_ranges

1ms  3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_mixed_ranges**Why Needed:** To test the functionality of compressing mixed ranges in a list.**Key Assertions:**

- {'expected': '1-3, 5, 10-12, 15', 'actual': '1-3, 5, 10-12, 15'}
- {'expected': 'True', 'actual': 'True'}

Confidence: 80%**Tokens:** 130 input + 103 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/ranges.py	16 lines (ranges: 29, 33, 35-37, 39-40, 42, 45-47, 50, 52, 65-67)

PASSED tests/test_ranges.py::TestCompressRanges::test_non_consecutive_lines 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_non_consecutive_lines

Why Needed: To ensure that non-consecutive lines are correctly compressed to a single value.

Key Assertions:

- {'expected': '1, 3, 5', 'actual': '1, 3, 5'}

Confidence: 80%

Tokens: 113 input + 84 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/ranges.py	14 lines (ranges: 29, 33, 35-37, 39-40, 45-47, 50, 52, 65-66)

PASSED tests/test_ranges.py::TestCompressRanges::test_single_line 1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestCompressRanges::test_single_line

Why Needed: The single line should not use range notation.

Key Assertions:

- {'expected': '5', 'actual': '5'}

Confidence: 80%

Tokens: 96 input + 64 output = 160 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 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 correctly compressed into a single range.**Key Assertions:**

- {'expected': '1-2', 'actual': '1-2'}

Confidence: 80%**Tokens:** 103 input + 74 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/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:** This test is necessary because the current implementation of 'compress_ranges' only works with input that is already sorted.**Key Assertions:**

- {'expected': '[1, 3]', 'actual': '1-3'}

Confidence: 80%**Tokens:** 110 input + 84 output = 194 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	16 lines (ranges: 29, 33, 35-37, 39-40, 42, 45-47, 50, 52, 65-67)

PASSED

tests/test_ranges.py::TestExpandRanges::test_empty_string

1ms 3

AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_empty_string**Why Needed:** The current implementation does not handle empty strings correctly.**Key Assertions:**

- {'name': 'expected result', 'value': [], 'description': 'Expected the function to return an empty list when given an empty string.'}

Confidence: 80%**Tokens:** 90 input + 82 output = 172 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	2 lines (ranges: 81-82)

PASSED

tests/test_ranges.py::TestExpandRanges::test_mixed

1ms



AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_mixed**Why Needed:** The test is necessary because it checks for the expansion of mixed ranges and singles in the `expand_ranges` function.**Key Assertions:**

- {'expected': [1, 2, 3, 5, 10, 11, 12], 'actual': ['1', '2', '3', '5', '10', '11', '12']}
- {'expected': [], 'actual': []}

Confidence: 80%**Tokens:** 121 input + 119 output = 240 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	11 lines (ranges: 81, 84-91, 93, 95)

PASSED

tests/test_ranges.py::TestExpandRanges::test_range

1ms



AI ASSESSMENT

Scenario: tests/test_ranges.py::TestExpandRanges::test_range**Why Needed:** The range function should expand to a list.**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



AI ASSESSMENT

Scenario: compress_ranges and expand_ranges should be inverses.

Why Needed: Because the problem statement requires that `compress_ranges` and `expand_ranges` are inverses, meaning they should have a round-trip relationship. This ensures that any input to one function can be uniquely recovered from its output.

Key Assertions:

- {'name': 'original input', 'value': [1, 2, 3, 5, 10, 11, 12, 15]}
- {'name': 'compressed input', 'value': []}
- {'name': 'expanded input', 'value': [1, 2, 3, 5, 10, 11, 12, 15]}

Confidence: 80%

Tokens: 134 input + 161 output = 295 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 test is necessary because it checks for a specific scenario where the function expands ranges around a single number.**Key Assertions:**

- {'expected_value': [5], 'actual_value': ['5']}

Confidence: 80%**Tokens:** 95 input + 77 output = 172 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/ranges.py	7 lines (ranges: 81, 84-87, 93, 95)

 tests/test_render.py

9 tests

PASSED

tests/test_render.py::TestFormatDuration::test_milliseconds

1ms



AI ASSESSMENT

Scenario: Test the format_duration function with different input values to verify it correctly formats as milliseconds for < 1s.

Why Needed: This test prevents a potential regression where the function might not be able to accurately format durations less than 1 second.

Key Assertions:

- The function should return '500ms' when given an input of 0.5 seconds.
- The function should return '1ms' when given an input of 0.001 seconds.
- The function should return '0ms' when given an input of 0.0 seconds.

Confidence: 80%

Tokens: 211 input + 131 output = 342 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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:** The test is necessary because the function `format_duration()` expects a float value as input, but it receives an integer value in the provided code.**Key Assertions:**

- {'name': 'format_duration(1.23)', 'expected': '1.23s'}
- {'name': 'format_duration(60.0)', 'expected': '60.00s'}

Confidence: 80%**Tokens:** 116 input + 105 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/render.py	2 lines (ranges: 65-66)

PASSED

tests/test_render.py::TestOutcomeToCssClass::test_all_outcomes

1ms



AI ASSESSMENT

Scenario: Test that all outcomes map to CSS classes.**Why Needed:** Prevents regression in CSS class mapping for different outcome types.**Key Assertions:**

- The function `outcome_to_css_class` should return a valid CSS class for each outcome type.
- The function `outcome_to_css_class` should handle all three outcome types ('passed', 'failed', and 'skipped') correctly.
- The function `outcome_to_css_class` should map the string 'xfailed' to the correct CSS class ('outcome-xfailed').
- The function `outcome_to_css_class` should map the string 'error' to the correct CSS class ('outcome-error').

Confidence: 80%**Tokens:** 263 input + 150 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	8 lines (ranges: 79-85, 87)

PASSED

tests/test_render.py::TestOutcomeToCssClass::test_unknown_outcome

1ms



AI ASSESSMENT

Scenario: tests/test_render.py::TestOutcomeToCssClass::test_unknown_outcome**Why Needed:** Unknown outcomes are not handled by the `outcome_to_css_class` function.**Key Assertions:**

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

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/render.py	8 lines (ranges: 79-85, 87)

PASSED

tests/test_render.py::TestRenderFallbackHtml::test_renders_basic_report

1ms



3

AI ASSESSMENT

Scenario: Test renders basic report with fallback HTML.**Why Needed:** Prevents rendering of incomplete or corrupted reports due to plugin or repository issues.**Key Assertions:**

- The '' header is present in the rendered HTML.
- The 'Test Report' title is included in the rendered HTML.
- The 'test::passed' and 'test::failed' node IDs are found in the rendered HTML.
- The 'PASSED' and 'FAILED' keywords are displayed in the rendered HTML.
- The plugin version and repository version are displayed in the rendered HTML.
- The '**Plugin:**' and '**Repo:**' tags contain the expected values in the rendered HTML.

Confidence: 80%**Tokens:** 426 input + 163 output = 589 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	57 lines (ranges: 65-67, 79-85, 87, 121-124, 126-127, 131-132, 155-157, 159-167, 172-174, 210-211, 224, 257-264, 267, 269, 271-277, 280-281, 285)

PASSED

tests/test_render.py::TestRenderFallbackHtml::test_renders_coverage

1ms



AI ASSESSMENT

Scenario: Test 'test_renders_coverage' verifies that the test renders coverage information and includes it in the rendered HTML.

Why Needed: This test prevents regression by ensuring that the test renders coverage information, which is crucial for tracking code coverage.

Key Assertions:

- The report root contains a summary with total and passed tests.
- The report root contains a summary with total and passed tests.
- The report root contains a summary with total and passed tests.
- The test renders coverage information in the rendered HTML.
- The test renders coverage information in the rendered HTML.
- The test renders coverage information in the rendered HTML.

Confidence: 80%

Tokens: 288 input + 145 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	57 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-129, 131-132, 155-156, 159-167, 172-174, 210-211, 224, 257-264, 267, 269, 271-277, 280-281, 285)

PASSED

tests/test_render.py::TestRenderFallbackHtml::test_renders_llm_annotation

1ms

3

AI ASSESSMENT

Scenario: tests/test_render.py::TestRenderFallbackHtml::test_renders_llm_annotation**Why Needed:** This test prevents the rendering of LLM annotations with a confidence score below 0.8, which could cause authentication bypass issues.**Key Assertions:**

- The report includes the 'Tests login flow' scenario.
- The report includes the 'Prevents auth bypass' why needed message.
- The report includes the 'Confidence:' assertion with value '85%' or higher.
- The report does not include any annotations without a confidence score of 0.8 or above.
- The report does not contain any annotations without a scenario, why needed, or confidence assertion.
- The report is rendered correctly and does not contain any errors or warnings.
- The report includes all expected nodes in the HTML output.
- The report includes all expected assertions in the HTML output.

Confidence: 80%**Tokens:** 317 input + 199 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/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



AI ASSESSMENT

Scenario: Verifies that the test renders source coverage summary with required information.**Why Needed:** This test prevents a regression where the source coverage summary is missing or incomplete.**Key Assertions:**

- The 'Source Coverage' section should be present in the HTML report.
- 'src/foo.py'

Confidence: 80%**Tokens:** 331 input + 75 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	68 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-127, 131-132, 155-156, 159-167, 172-178, 180-186, 191, 206, 210-211, 224, 257-264, 267, 269, 271-277, 280-281, 285)

PASSED

tests/test_render.py::TestRenderFallbackHtml::test_renders_xpass_summary

1ms



3

AI ASSESSMENT

Scenario: Test renders xpass summary for ReportRoot with Summary containing both xfailed and xpassed entries.

Why Needed: This test prevents a rendering issue where the 'xfailed' and 'xpassed' counts are not displayed correctly when there is an overlap in the report summary.

Key Assertions:

- The HTML contains both 'XFailed' and 'XPassed' strings.
- The HTML does not contain any other text that might interfere with the display of 'XFailed' and 'XPassed'.
- The 'XFailed' string is displayed correctly, even if there are multiple xfailed entries in the report summary.
- The 'XPassed' string is also displayed correctly, even if there are multiple xpassed entries in the report summary.
- No other text is displayed that might interfere with the display of 'XFailed' and 'XPassed'.
- There are no duplicate counts for either 'xfailed' or 'xpassed' in the report summary.

Confidence: 80%

Tokens: 283 input + 216 output = 499 total

COVERAGE

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



AI ASSESSMENT

Scenario: tests/test_report_writer.py::TestComputeSha256::test_different_content

Why Needed: To ensure that different content produces different hashes.

Key Assertions:

- {'expected': 'different', 'actual': 'not equal'}

Confidence: 80%

Tokens: 115 input + 66 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/report_writer.py	1 lines (ranges: 55)

PASSED

tests/test_report_writer.py::TestComputeSha256::test_empty_bytes

1ms



AI ASSESSMENT

Scenario: tests/test_report_writer.py::TestComputeSha256::test_empty_bytes

Why Needed: To ensure that an empty bytes object produces consistent hash values.

Key Assertions:

- {'message': 'Expected hash1 to be equal to hash2', 'expected_result': 'hash1 == hash2'}
- {'message': 'Expected length of hash1 to be 64', 'expected_result': 'len(hash1) == 64'}

Confidence: 80%

Tokens: 129 input + 104 output = 233 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	1 lines (ranges: 55)

AI ASSESSMENT

Scenario: Test that the build_run_meta method returns the correct duration and version information for a test run.

Why Needed: This test prevents regression where the duration of a test run is not accurately reported.

Key Assertions:

- The duration of the test run should be accurate (60 seconds in this case).
- The pytest version should be present in the metadata.
- The plugin version and Python version should also be included in the metadata.

Confidence: 80%

Tokens: 318 input + 105 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	72 lines (ranges: 67-74, 76-81, 83-84, 98-99, 102, 105-108, 110, 127-128, 130, 156-158, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307)

PASSED

tests/test_report_writer.py::TestReportWriter::test_build_summary_all_outcomes

1ms



4

AI ASSESSMENT

Scenario: Test verifies the ReportWriter to correctly build a summary of all outcome types.**Why Needed:** This test prevents regression where the total count of outcomes is incorrect.**Key Assertions:**

- The total number of tests should be equal to 6 (all outcome types).
- The passed tests should have 1 outcome type ('passed').
- The failed tests should have 1 outcome type ('failed').
- The skipped tests should have 1 outcome type ('skipped').
- The xfailed and xpassed tests should have 1 outcome type each ('xfailed' or 'xpassed').
- The error test should have 1 outcome type ('error').

Confidence: 80%**Tokens:** 336 input + 157 output = 493 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 the `Summary` object correctly counts all tests.

Why Needed: This test prevents a regression where the total count of passed and failed tests is incorrect.

Key Assertions:

- summary.total should be equal to the number of tests.
- summary.passed should be equal to the number of tests that were passed.
- summary.failed should be equal to the number of tests that were failed.
- summary.skipped should be equal to the number of tests that were skipped.

Confidence: 80%

Tokens: 283 input + 123 output = 406 total

COVERAGE

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

Why Needed: This test prevents a potential bug where the `ReportWriter` instance is not properly initialized with the required configuration.

Key Assertions:

- The `config` attribute of the `writer` object should be set to an instance of `Config`.
- The `warnings` attribute of the `writer` object should be an empty list.
- The `artifacts` attribute of the `writer` object should be an empty list.

Confidence: 80%

Tokens: 199 input + 126 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)
src/pytest_llm_report/report_writer.py	3 lines (ranges: 156-158)

PASSED

tests/test_report_writer.py::TestReportWriter::test_write_report_ass
embles_tests

4ms



4

AI ASSESSMENT

Scenario: Test that ReportWriter writes a report with all tests.**Why Needed:** To prevent regression in the case where no output paths are specified for the test reports.**Key Assertions:**

- The length of the report.tests list should be equal to 2.
- The total value of report.summary.total should be equal to 2.

Confidence: 80%**Tokens:** 255 input + 83 output = 338 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 5ms 4

AI ASSESSMENT

Scenario:

tests/test_report_writer.py::TestReportWriter::test_write_report_includes_coverage_percent

Why Needed: To ensure that the ReportWriter class correctly calculates and returns the total coverage percentage from a report.

Key Assertions:

- {'expected_value': 85.5, 'actual_value': 'report.summary.coverage_total_percent'}

Confidence: 80%

Tokens: 132 input + 87 output = 219 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	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 4ms 4

AI ASSESSMENT

Scenario: Test ReportWriter::test_write_report_includes_source_coverage verifies that the test writes a report with source coverage summary.

Why Needed: This test prevents regression where the test does not include source coverage in the report, potentially misleading users about the code's quality.

Key Assertions:

- The length of `report.source_coverage` should be 1.
- The file path of `report.source_coverage[0]` should match `

Confidence: 80%

Tokens: 291 input + 107 output = 398 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 4ms 4

AI ASSESSMENT

Scenario: Test ReportWriter::test_write_report_merges_coverage verifies that the report writer merges coverage into tests.

Why Needed: This test prevents regression where the report does not merge coverage into tests, potentially leading to inaccurate or misleading test results.

Key Assertions:

- The report should have a single merged coverage entry for the given test.
- The file path of the merged coverage entry should match the expected file path.
- All line ranges and counts in the coverage entry should be present.

Confidence: 80%

Tokens: 285 input + 115 output = 400 total

COVERAGE

src/pytest_llm_report/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 that the ReportWriterWithFiles class falls back to direct write if an atomic write fails and a cross-device link occurs.

Why Needed: This test prevents regression that would occur when the atomic write operation fails and a cross-device link is attempted.

Key Assertions:

- The report file should exist at the specified path.
- Any warnings generated by the ReportWriterWithFiles class should have a code of 'W203'.

Confidence: 80%

Tokens: 276 input + 103 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/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` 7ms 6

AI ASSESSMENT

Scenario: Test Report Writer with Files**Why Needed:** The test should create an output directory if it doesn't exist.**Key Assertions:**

- {'name': 'Output Directory Existence', 'expected': True, 'actual': 'exists'}

Confidence: 80%**Tokens:** 171 input + 69 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/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 verifies that the `test_ensure_dir_failure` test case ensures a directory creation failure is captured as a warning.

Why Needed: This test prevents a potential bug where the directory creation fails silently and does not raise an exception, potentially leading to unexpected behavior or errors in downstream code.

Key Assertions:

- The `writer.warnings` list contains warnings with code 'W201' for any directory creation attempts that fail.
- Any directory creation attempt that raises an OSError is captured as a warning with code 'W201'.
- The test verifies that the `test_ensure_dir_failure` function correctly identifies and reports directory creation failures.

Confidence: 80%

Tokens: 278 input + 147 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/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: The test verifies that the `get_git_info` function handles git command failures by returning `None` for both `sha` and `dirty` variables.

Why Needed: This test prevents a regression where the `get_git_info` function fails to return expected values when it encounters a git command failure.

Key Assertions:

- The `get_git_info` function should return `None` for both `sha` and `dirty` variables in case of a git command failure.
- The test should pass even if the `git not found` exception is raised during the execution of `subprocess.check_output`.
- The expected values `None` for `sha` and `dirty` variables should be verified after calling `get_git_info`.
- The function should handle git command failures without raising an exception or crashing the test process.
- The test should verify that the returned values are consistent with the expected behavior in case of a failure.
- The test should cover all possible scenarios where `git not found` is raised during the execution of `subprocess.check_output`.

Confidence: 80%

Tokens: 231 input + 237 output = 468 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 37ms 6

AI ASSESSMENT

Scenario: Test verifies that the report writer creates an HTML file with expected content.**Why Needed:** This test prevents regression by ensuring that the report writer correctly generates an HTML file containing all necessary information.**Key Assertions:**

- The 'report.html' file should exist at the specified path.
- The 'report.html' file should contain the expected text content.
- All required assertions ('test1', 'test2') should be present in the 'report.html' file.
- The report writer should correctly identify test results as passed, failed, skipped, XFailed, or XPassed.
- Errors and warnings should not be included in the 'report.html' file.

Confidence: 80%**Tokens:** 366 input + 152 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/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 37ms 6

AI ASSESSMENT

Scenario: The test verifies that the report writer includes xfail outcomes in the HTML summary.

Why Needed: This test prevents regressions where the report writer does not include xfail outcomes in the HTML summary.

Key Assertions:

- Asserts that 'XFAILED' and 'XPASSED' are present in the HTML string.
- Checks if the text content of the HTML includes both 'XFAILED' and 'XPASSED'.
- Verifies that 'xfailed' and 'xpassed' keywords are found in the HTML summary.
- Ensures that the report writer correctly identifies xfail outcomes as 'xfailed' or 'xpassed'.
- Checks if the report includes all required text content for an xfail outcome.
- Verifies that the report does not include any other keywords related to failures.
- Ensures that the report is properly formatted and includes only necessary information.

Confidence: 80%

Tokens: 308 input + 200 output = 508 total

COVERAGE

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

5ms



AI ASSESSMENT

Scenario: Test verifies that a JSON file is created with the report.**Why Needed:** This test prevents regression where the report writer does not create a JSON file.**Key Assertions:**

- The `report.json` file should be created in the specified path.
- At least one artifact should be tracked for the report.
- The number of artifacts tracked should be greater than zero.
- The file exists at the specified path.
- The `ReportWriter` class correctly tracks artifacts.
- The `write_report` method creates a new JSON file with the report.

Confidence: 80%**Tokens:** 265 input + 131 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/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 40ms 6

AI ASSESSMENT

Scenario: Should create PDF file when Playwright is available.

Why Needed: This test prevents regression where the report writer does not create a PDF file even if Playwright is available.

Key Assertions:

- The `write_pdf` method of the `ReportWriter` class should be able to write the contents of the `report.pdf` file to the system.
- The `report.pdf` path should be created in the test directory.
- Any artifacts generated by the report writer should have a path that matches the expected value for `report.pdf`.
- The `write_report` method should raise an exception if Playwright is not available.
- If Playwright is available, it should create a PDF file with the specified path and contents.

Confidence: 80%

Tokens: 478 input + 167 output = 645 total

COVERAGE

src/pytest_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 5ms 4

AI ASSESSMENT

Scenario: Test verifies that a warning is raised when Playwright is missing for PDF output.

Why Needed: This test prevents the 'Warning: Could not find Playwright' message from being displayed when using PDF output.

Key Assertions:

- The file 'report.pdf' should exist.
- Any warnings with code W204_PDF_PLAYWRIGHT_MISSING should be present in the report.

Confidence: 80%

Tokens: 311 input + 93 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/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 2ms 3

AI ASSESSMENT

Scenario:

tests/test_report_writer_coverage.py::TestGetGitInfo::test_git_info_from_nonexistent_path

Why Needed: To test the Report Writer's ability to handle non-git directory paths and return empty values for SHA1 and dirty flags.

Key Assertions:

- {'name': 'assert sha is None', 'expected': {'type': 'NoneType', 'value': ''}, 'message': 'Expected get_git_info() to return None for sha parameter'}
- {'name': 'assert dirty is None', 'expected': {'type': 'NoneType', 'value': ''}, 'message': 'Expected get_git_info() to return None for dirty parameter'}

Confidence: 80%

Tokens: 123 input + 159 output = 282 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 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 4ms 3

AI ASSESSMENT

Scenario: tests/test_report_writer_coverage.py::TestGetGitInfo::test_git_info_from_valid_repo

Why Needed: To ensure that the `get_git_info` function returns a valid Git SHA and/or a string representation of the Git repository.

Key Assertions:

- {'name': 'assert sha is None or isinstance(sha, str)', 'description': 'The test asserts that the returned Git SHA is either None or an instance of str.'}

Confidence: 80%

Tokens: 159 input + 109 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	16 lines (ranges: 67-74, 76-81, 83-84)

PASSED

tests/test_report_writer_coverage.py::TestGetPluginGitInfo::test_plugin_git_info_fallback

1ms



3

AI ASSESSMENT

Scenario: Test falls back to git runtime when _git_info import fails.

Why Needed: To prevent a critical bug where the plugin Git Info cannot be imported and we still rely on it for fallback functionality.

Key Assertions:

- The function `get_plugin_git_info()` returns None or a string.
- The function `get_plugin_git_info()` does not raise an exception when _git_info import fails.
- The assertion `assert sha is None or isinstance(sha, str)` passes even if the fallback via git runtime fails.

Confidence: 80%

Tokens: 253 input + 123 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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	3 lines (ranges: 127-128, 130)

PASSED

tests/test_report_writer_coverage.py::TestGetPluginGitInfo::test_plugin_git_info_returns_values

1ms



AI ASSESSMENT

Scenario: tests/test_report_writer_coverage.py::TestGetPluginGitInfo

Why Needed: To ensure that the plugin's Git information can be retrieved and used correctly.

Key Assertions:

- {'expected_type': 'str', 'actual_type': 'None'}
- {'expected_type': 'str', 'actual_type': 'is None or str'}

Confidence: 80%

Tokens: 141 input + 94 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	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

6ms



AI ASSESSMENT

Scenario: Test atomic write fallback**Why Needed:** To ensure the ReportWriter can handle unexpected errors during atomic writes.**Key Assertions:**

- {'name': 'Report writer should create a backup report when writing fails', 'description': 'The backup report is created even if the direct write operation fails.'}

Confidence: 80%**Tokens:** 181 input + 79 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/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 114ms 6

AI ASSESSMENT

Scenario: Test PDF generation when playwright raises exception (lines 424-432).**Why Needed:** Prevents bug where the report writer fails to generate a PDF due to playwright exceptions.**Key Assertions:**

- The `writer.warnings` list should contain 'W201' warnings for each failed playwright context.
- The `writer.warnings` list should contain 'W202' warnings for each successful playwright context.
- The `report.pdf` file should be present in the test directory with a PDF extension.
- The `report.pdf` file should have a warning code of W201 or W202.
- The `report.pdf` file should not be empty.
- The `report.pdf` file should be a valid PDF document (not corrupted).
- The `writer.write_pdf` method should raise an exception when playwright raises an exception.

Confidence: 80%**Tokens:** 356 input + 190 output = 546 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	26 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65-67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	28 lines (ranges: 156-158, 408, 417, 419, 421-423, 431-436, 439, 441-442, 455, 460, 462, 465-469, 477-478)

PASSED

tests/test_report_writer_coverage.py::TestReportWriterPDF::test_pdf_playwright_not_installed

1ms



4

AI ASSESSMENT

Scenario: Test PDF generation when playwright is not installed.

Why Needed: Prevents a potential bug where the report writer fails to create a PDF file due to playwright being missing.

Key Assertions:

- The 'W204' warning should be present in the output of the `writer.write_pdf()` method.
- The PDF file 'report.pdf' should not exist after the test is completed.

Confidence: 80%

Tokens: 293 input + 94 output = 387 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	9 lines (ranges: 156-158, 408-412, 415)

PASSED

tests/test_report_writer_coverage.py::TestReportWriterPDF::test_resolve_html_sourceCreatesTemp 33ms 6

AI ASSESSMENT

Scenario: Test _resolve_pdf_html_source creates temp file when no HTML source exists.

Why Needed: Prevents a potential bug where the test fails if there is no HTML source provided.

Key Assertions:

- The report writer should create a temporary HTML file.
- The temporary HTML file should have the correct suffix (.html).
- The path to the temporary HTML file should exist and be accessible.
- The suffix of the temporary HTML file should match the expected value (HTML).

Confidence: 80%

Tokens: 265 input + 115 output = 380 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	26 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65-67, 79-85, 87, 99, 101-102, 107)
src/pytest_llm_report/report_writer.py	11 lines (ranges: 156-158, 455, 460, 462, 465-469)

PASSED

tests/test_report_writer_coverage.py::TestReportWriterPDF::test_resolve_html_source_missing_html_file

33ms



AI ASSESSMENT

Scenario: Test _resolve_pdf_html_source when configured HTML doesn't exist.**Why Needed:** Prevents a bug where the test fails due to an unresolvable HTML source.**Key Assertions:**

- The `report_html` configuration should be able to resolve the HTML source even if it's missing.
- The resolved path for the HTML file should exist and not be a temporary file.
- The test should still pass even when the HTML file is nonexistent.

Confidence: 80%**Tokens:** 270 input + 108 output = 378 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/render.py	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



AI ASSESSMENT

Scenario: Verify that the `_resolve_pdf_html_source` method uses an existing HTML file as required.

Why Needed: This test prevents a potential bug where the report writer does not use an existing HTML file, potentially leading to incorrect or missing information in the report.

Key Assertions:

- The path of the resolved PDF is set to the specified HTML file.
- The report is not temporary (`is_temp=False`).
- The resolution method uses an existing HTML file instead of creating a new one.

Confidence: 80%

Tokens: 266 input + 115 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/options.py	2 lines (ranges: 123, 171)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/report_writer.py	7 lines (ranges: 156-158, 455-458)

tests/test_schemas.py

2 tests

PASSED

tests/test_schemas.py::TestAnnotationSchema::test_from_dict_full

1ms



AI ASSESSMENT

Scenario: Test that `AnnotationSchema.from_dict` creates a valid annotation from a dictionary with all required fields.

Why Needed: Prevents regression in the case where an invalid input is provided, causing the test to fail and potentially introducing bugs.

Key Assertions:

- assert schema.scenario == "Verify login",
- assert schema.why_needed == "Catch auth bugs",
- assert schema.key_assertions == ["assert 200", "assert token"]
- # Correctly asserts all required fields

Confidence: 80%

Tokens: 276 input + 120 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/llm/schemas.py	5 lines (ranges: 77-81)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_schemas.py::TestAnnotationSchema::test_to_dict_full

1ms



AI ASSESSMENT

Scenario: Should convert to dictionary with all fields.

Why Needed: Prevent regression in authentication logic, ensuring correct data is returned for login scenarios.

Key Assertions:

- assert 'Verify login' in data['scenario']
- assert 'Catch auth bugs' in data['why_needed']
- assert all([item in data['key_assertions'] for item in ['assert 200', 'assert token']])
- assert data['confidence'] == 0.95

Confidence: 80%

Tokens: 273 input + 111 output = 384 total

COVERAGE

src/pytest_llm_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 86ms 8

AI ASSESSMENT

Scenario: The HTML report is generated correctly.**Why Needed:** This test prevents a regression where the report generation fails due to missing dependencies.**Key Assertions:**

- The report file exists at the specified path.
- The report contains the expected test name.
- The report includes all necessary HTML tags.

Confidence: 80%**Tokens:** 264 input + 76 output = 340 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

123ms



8

AI ASSESSMENT

Scenario: test_html_summary_counts_all_statuses verifies that the HTML summary counts include all statuses.

Why Needed: This test prevents regression where the report does not include all statuses, which can make it difficult to diagnose issues.

Key Assertions:

- asserts that the 'Total Tests' label is present and has a count of 6.
- asserts that the 'Passed' label has a count of 1.
- asserts that the 'Failed' label has a count of 1.
- asserts that the 'Skipped' label has a count of 1.
- asserts that the 'XFailed' label has a count of 1.
- asserts that the 'XPassed' label has a count of 1.
- asserts that the 'Errors' and 'Error' labels have counts equal to 1.

Confidence: 80%

Tokens: 621 input + 191 output = 812 total

COVERAGE

src/pytest_llm_report/collector.py	69 lines (ranges: 78-79, 90, 93-94, 96, 99-104, 106-107, 109-112, 114-119, 121-122, 124, 127, 132-133, 140-141, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 212-214, 216, 227-228, 230-236, 250-251, 261, 264, 268, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482-484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555,

561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)

src/pytest_llm_report/plugin.py

288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 399, 403, 407, 410, 429-430, 437-438, 441-442, 444-445, 448-452, 454, 457-458, 460, 463-464, 485, 491-492, 534-544, 558-559, 562, 566-568, 579, 583, 602, 606-608, 619, 623, 626, 628-629)

src/pytest_llm_report/render.py

25 lines (ranges: 30-31, 40, 42-46, 50-51, 53, 65, 67, 79-85, 87, 99, 101-102, 107)

src/pytest_llm_report/report_writer.py

116 lines (ranges: 55, 67-73, 85-86, 98-100, 127-128, 130, 156-158, 186, 192-193, 197-198, 202, 211-218, 222, 226-227, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 302-303, 305-307, 319, 321-322, 324-335, 337, 383, 385-386, 389, 392, 395, 398-402, 477-478, 502, 504, 506-508, 510, 513)

PASSED

tests/test_smoke_pytester.py::TestBasicReportGeneration::test_json_report_created 63ms 7

AI ASSESSMENT

Scenario: The JSON report is created and its existence and content are verified.

Why Needed: This test prevents a potential issue where the report generation process fails to create or write the expected JSON file.

Key Assertions:

- report_path.exists()
- data['schema_version']
- data['summary']['total'] == 2
- data['summary']['passed'] == 1
- data['summary']['failed'] == 1

Confidence: 80%

Tokens: 295 input + 108 output = 403 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 55ms 14

AI ASSESSMENT

Scenario: Test that LLM annotations are included in the report when a provider is enabled.**Why Needed:** Prevents regressions by ensuring LLM annotations are present in the report.**Key Assertions:**

- The test passes without any errors or warnings.
- The 'LLM annotations' key is present in the report.
- The 'LLM annotations' value is 'True'.
- The 'why_needed' value is 'Prevents regressions'.
- The 'key_assertions' list contains the expected assertions.
- The 'choices' list within the 'payload' dictionary contains a valid LLM annotation.
- The 'message' key within each 'choice' dictionary has a valid message.
- The 'content' value of each 'message' dictionary is a valid JSON string.

Confidence: 80%**Tokens:** 385 input + 181 output = 566 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 89ms 14

AI ASSESSMENT

Scenario: Verify that LLM errors are surfaced in HTML output.**Why Needed:** Prevent regression where LLM errors are not reported correctly.**Key Assertions:**

- The function `test_pass()` should raise an error with a meaningful message.
- The function `pytestLLMReport` should be able to detect and report the error.
- The HTML output of the test should contain the error message.
- The error message should be in the correct format (e.g. 'LLM errors are surfaced in HTML output.')
- The error message should not be a simple string but rather an actual error message from `litellm.completion`.
- The function `pytestLLMReport` should be able to handle different types of LLM errors (e.g. syntax errors, semantic errors).

Confidence: 80%**Tokens:** 313 input + 181 output = 494 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: Verify that the LLM opt-out marker is correctly recorded and reported.

Why Needed: This test prevents a regression where the LLM opt-out marker is not properly recorded or reported, potentially leading to incorrect analysis results.

Key Assertions:

- The 'llm_opt_out' marker should be present in the test list.
- The 'llm_opt_out' marker should have an associated boolean value of True.
- The number of tests with the 'llm_opt_out' marker should be exactly 1.
- The first test in the list should have a 'llm_opt_out' marker.
- The 'llm_opt_out' marker should not be present for any other tests.

Confidence: 80%

Tokens: 290 input + 159 output = 449 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: Verify requirement marker is recorded and correctly identified in report.

Why Needed: This test prevents a regression where the requirement marker might not be properly recorded or identified in the report.

Key Assertions:

- The 'requirement' keyword is used to mark requirements with specific names (REQ-001, REQ-002).
- The report generated by pytester includes all tests that have been marked as requiring a specific requirement.
- The test asserts that there is exactly one test that has been marked with both 'requirement' and 'REQ-001'.
- The test asserts that the required requirements are included in the list of tests.
- The test verifies that the 'REQ-001' and 'REQ-002' requirements are correctly identified in the report.

Confidence: 80%

Tokens: 307 input + 172 output = 479 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 60ms 7

AI ASSESSMENT

Scenario: Verifies that multiple xfailed tests are recorded in the report.**Why Needed:** Prevents regression where multiple tests fail due to a common cause, such as a test suite being updated or refactored.**Key Assertions:**

- The number of xfailed tests is correctly reported.
- Each xfailed test has an outcome of 'xfailed'.
- No other outcomes are recorded in the report for this test suite.

Confidence: 80%**Tokens:** 317 input + 103 output = 420 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

53ms  7

AI ASSESSMENT

Scenario: Test that skipped tests are recorded and reported correctly.**Why Needed:** This test prevents regression in the reporting of skipped tests, ensuring they are accurately counted and displayed in the report.**Key Assertions:**

- The 'skipped' key in the report data should contain a value equal to 1.
- The 'summary' section of the report data should contain an entry for 'skipped'.
- The number of skipped tests should be exactly 1 when reported in the JSON file.
- The 'report.json' file should contain a single key-value pair with the key 'skipped' and value 1.

Confidence: 80%**Tokens:** 264 input + 143 output = 407 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

58ms  7

AI ASSESSMENT

Scenario: Verify that xfailed tests are correctly recorded in the report.**Why Needed:** This test prevents regression where only failed tests are reported.**Key Assertions:**

- The 'summary' key under 'xfailed' is present and contains a value of 1.
- The number of failed tests is exactly 1 as expected by the test.
- The xfailed count matches the actual number of failed tests in the report.

Confidence: 80%**Tokens:** 264 input + 102 output = 366 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

57ms



7

AI ASSESSMENT

Scenario: Testing parameterized tests to ensure correct reporting.**Why Needed:** This test prevents regression in the reporting system by verifying that all parameterized tests are recorded correctly.**Key Assertions:**

- The total number of successful parameterized tests is 3.
- All parameterized tests are recorded separately and have a 'passed' status.
- The report.json file contains accurate data about the test runs.

Confidence: 80%**Tokens:** 290 input + 96 output = 386 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

48ms



AI ASSESSMENT

Scenario: tests/test_smoke_pytester.py::TestPluginRegistration::test_help_contains_examples

Why Needed: The test is necessary to ensure the CLI help text includes usage examples.

Key Assertions:

- {'name': 'stdout', 'description': 'The test checks if the stdout matches the expected output.'}

Confidence: 80%

Tokens: 123 input + 81 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	89 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)

src/pytest_llm_report/plugin.py

240 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 558-559, 562, 566-568)

PASSED

tests/test_smoke_pytester.py::TestPluginRegistration::test_markers_registered 42ms 3

AI ASSESSMENT

Scenario: tests/test_smoke_pytester.py::TestPluginRegistration::test_markers_registered

Why Needed: The test is necessary to ensure that the LLM markers are registered correctly.

Key Assertions:

- {'message': 'LLM markers are not registered. Expected: True', 'expected': True}
- {'message': 'LLM context marker is not registered. Expected: True', 'expected': True}
- {'message': 'requirement marker is not registered. Expected: True', 'expected': True}

Confidence: 80%

Tokens: 142 input + 133 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	89 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	240 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349,

PASSED

tests/test_smoke_pytester.py::TestPluginRegistration::test_plugin_registered 49ms 3

AI ASSESSMENT

Scenario: TestPluginRegistration**Why Needed:** To verify that the plugin is successfully registered with pytest.**Key Assertions:**

- {'name': 'The plugin is registered', 'message': 'Plugin is registered via pytest11'}

Confidence: 80%**Tokens:** 118 input + 64 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	89 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482, 484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	240 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159, 161-165, 169-174, 176-181, 185-190, 192-197, 199-204, 208-213, 215-219, 223-227, 229-233, 235-239, 241-245, 247-252, 254-258, 260-264, 268-272, 274-279, 283-287, 289-293, 297-302, 304-309, 311-315, 328-330, 332-334, 336-338, 342, 346-347, 349, 351, 354-355, 362, 371-373, 558-559, 562, 566-568)

PASSED

tests/test_smoke_pytester.py::TestSpecialCharacters::test_special_characters_in_nodeid

91ms



8

AI ASSESSMENT

Scenario: Test verifies that special characters in nodeid are handled correctly.**Why Needed:** This test prevents a potential crash and ensures the HTML is valid.**Key Assertions:**

- The presence of special characters in the nodeid is detected.
- The HTML report generated by pytester is valid.
- The " tag is present in the report.

Confidence: 80%**Tokens:** 288 input + 86 output = 374 total

COVERAGE

src/pytest_llm_report/collector.py	40 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-115, 124, 127, 132-133, 140, 155-159, 163-164, 167-169, 171, 181, 185-186, 198-199, 209-210, 277, 285)
src/pytest_llm_report/coverage_map.py	12 lines (ranges: 44-45, 58-60, 72-73, 83, 86, 88-90)
src/pytest_llm_report/errors.py	4 lines (ranges: 142-145)
src/pytest_llm_report/models.py	1 lines (ranges: 190)
src/pytest_llm_report/options.py	90 lines (ranges: 123, 171, 199, 202-203, 209-210, 217-218, 225-226, 233-234, 241, 245, 247, 249, 251, 253, 257-258, 265-266, 271, 273, 276, 284, 308, 311-312, 320-322, 460, 463, 466, 470, 472-473, 476-477, 482-484, 486, 488, 490, 492, 494, 499-500, 504-505, 511-512, 516-517, 521-522, 528-529, 534, 537-538, 542-543, 547-548, 554-555, 561-562, 566-567, 572, 575-576, 581, 583, 588-589, 593-594, 599, 601, 603, 605, 607, 611, 613)
src/pytest_llm_report/plugin.py	288 lines (ranges: 41, 44-48, 50-54, 56-60, 62-66, 68-72, 74-79, 81-86, 90-94, 96-100, 102-106, 108-112, 114-118, 122-126, 128-132, 134-138, 142-146, 148-153, 155-159,

161-165, 169-174, 176-181,
185-190, 192-197, 199-204,
208-213, 215-219, 223-227,
229-233, 235-239, 241-245,
247-252, 254-258, 260-264,
268-272, 274-279, 283-287,
289-293, 297-302, 304-309,
311-315, 328-330, 332-334,
336-338, 342, 346-347, 349,
351, 354-355, 362, 371-373,
399, 403, 407, 410, 429-430,
437-438, 441-442, 444-445,
448-452, 454, 457-458, 460,
463-464, 485, 491-492, 534-
544, 558-559, 562, 566-568,
579, 583, 602, 606-608, 619,
623, 626, 628-629)

src/pytest_llm_report/render.py

25 lines (ranges: 30-31, 40,
42-46, 50-51, 53, 65, 67, 79-
85, 87, 99, 101-102, 107)

src/pytest_llm_report/report_writer.py

106 lines (ranges: 55, 67-73,
85-86, 98-100, 127-128, 130,
156-158, 186, 192-193, 197-
198, 202, 211-218, 222, 226-
227, 230, 233, 254, 256-259,
262-264, 266, 268-275, 277-
278, 280-289, 291-294, 296-
297, 299-300, 302-303, 305-
307, 319, 321-322, 324-325,
337, 383, 385-386, 389, 392,
395, 398-402, 477-478, 502,
504, 506-508, 510, 513)

 tests/test_time.py

15 tests

PASSED

tests/test_time.py::TestFormatDuration::test_boundary_one_minute

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_boundary_one_minute

Why Needed: To ensure that the `format_duration` function correctly formats a duration of exactly one minute.

Key Assertions:

- {'name': 'result', 'expected_value': '1m 0.0s'}

Confidence: 80%

Tokens: 106 input + 80 output = 186 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	6 lines (ranges: 39, 41, 43, 46-48)

PASSED

tests/test_time.py::TestFormatDuration::test_microseconds_format

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_microseconds_format

Why Needed: To ensure 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 + 80 output = 201 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	2 lines (ranges: 39-40)

PASSED

tests/test_time.py::TestFormatDuration::test_milliseconds_format

1ms



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:**

- assert 'ms' in result
- assert result == '500.0ms'

Confidence: 80%**Tokens:** 119 input + 70 output = 189 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	3 lines (ranges: 39, 41-42)

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 that the `format_duration` function correctly formats durations over a minute, including displaying minutes and seconds.**Key Assertions:**

- {'name': "assert 'm' in result", 'expected': "'m'", 'actual': '1m 30.5s'}

Confidence: 80%**Tokens:** 124 input + 94 output = 218 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	6 lines (ranges: 39, 41, 43, 46-48)

PASSED

tests/test_time.py::TestFormatDuration::test_multiple_minutes

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_multiple_minutes**Why Needed:** To ensure the `format_duration` function correctly formats multiple minutes into a human-readable string.**Key Assertions:**

- {'expected': '3m 5.0s', 'actual': '3m 5.0s'}

Confidence: 80%**Tokens:** 112 input + 79 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_one_second

1ms



AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_one_second**Why Needed:** To ensure the `format_duration` function correctly formats a duration of exactly one second as '1.00s'.**Key Assertions:**

- {'expected': '1.00s', 'actual': '1.0'}

Confidence: 80%**Tokens:** 101 input + 83 output = 184 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	4 lines (ranges: 39, 41, 43-44)

PASSED

tests/test_time.py::TestFormatDuration::test_seconds_format

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_seconds_format

Why Needed: To ensure that the `format_duration` function correctly formats seconds under a minute.

Key Assertions:

- {'name': "result contains 's'", 'expected': 'True'}
- {'name': "result equals '5.50s'", 'expected': 'True'}

Confidence: 80%

Tokens: 110 input + 97 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	4 lines (ranges: 39, 41, 43-44)

PASSED

tests/test_time.py::TestFormatDuration::test_small_milliseconds

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestFormatDuration::test_small_milliseconds

Why Needed: To ensure the `format_duration` function correctly formats small millisecond durations.

Key Assertions:

- {'expected': 1.0, 'actual': '1.0ms'}

Confidence: 80%

Tokens: 111 input + 75 output = 186 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	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 test the functionality of formatting very small durations as microseconds.

Key Assertions:

- {'expected': '1μs', 'actual': '1'}

Confidence: 80%

Tokens: 116 input + 71 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	2 lines (ranges: 39-40)

PASSED

tests/test_time.py::TestIsoFormat::test_formats_datetime_with_utc

1ms  3

AI ASSESSMENT

Scenario: Test ISO Format with UTC**Why Needed:** To ensure the correct formatting of datetime objects in UTC timezone.**Key Assertions:**

- {'name': 'Expected result', 'value': '2024-01-15T10:30:45+00:00'}

Confidence: 80%**Tokens:** 143 input + 75 output = 218 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	1 lines (ranges: 27)

PASSED

tests/test_time.py::TestIsoFormat::test_formats_naive_datetime

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestIsoFormat::test_formats_naive_datetime**Why Needed:** To ensure that the naive datetime format is correct and consistent across different timezones.**Key Assertions:**

- {'name': 'Expected result', 'value': '2024-06-20T14:00:00'}

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/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 formatting with microseconds**Why Needed:** To ensure datetime objects are correctly formatted with microseconds.**Key Assertions:**

- {'assertion': 'The result is a string containing the correct microsecond value.', 'expected_result': '123456'}

Confidence: 80%**Tokens:** 133 input + 70 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: 27)

PASSED

tests/test_time.py::TestUtcNow::test_has_utc_timezone

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestUtcNow::test_has_utc_timezone**Why Needed:** To ensure that the `utc_now()` function returns a datetime object with an associated timezone.**Key Assertions:**

- {'name': 'assert result.tzinfo is not None', 'expected_result': "datetime.timezone('UTC')", 'actual_result': 'None'}
- {'name': 'assert result.tzinfo == UTC', 'expected_result': "datetime.timezone('UTC')", 'actual_result': 'None'}

Confidence: 80%**Tokens:** 109 input + 130 output = 239 total

COVERAGE

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

Why Needed: The test is necessary to ensure that the function correctly returns a JSON response with the expected structure.

Key Assertions:

- {'name': 'Expected JSON structure', 'description': 'The returned JSON should have the following structure: `{'}'}

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/plugin.py	6 lines (ranges: 558-559, 562, 566-568)
src/pytest_llm_report/util/time.py	1 lines (ranges: 15)

PASSED

tests/test_time.py::TestUtcNow::test_returns_datetime

1ms  3

AI ASSESSMENT

Scenario: tests/test_time.py::TestUtcNow::test_returns_datetime

Why Needed: To ensure that the `utc_now()` function returns a datetime object.

Key Assertions:

- {'name': 'isinstance(result, datetime)', 'expected': True}

Confidence: 80%

Tokens: 94 input + 71 output = 165 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 bug where the TokenRefresher class does not raise an exception when the get-token command fails, potentially causing unexpected behavior or errors in subsequent operations.

Key Assertions:

- The 'exit 1' status code is expected to be returned by subprocess.CompletedProcess().
- The error message 'Authentication failed' is expected to be included in the error message returned by pytest.raises(TokenRefreshError).

Confidence: 80%**Tokens:** 310 input + 119 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	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: Test that TokenRefresher raises an error when given an empty output.

Why Needed: To prevent 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 set to an empty string.
- An exception of type `TokenRefreshError` is raised with the message 'empty output'.
- The assertion passes if the output is indeed empty.

Confidence: 80%

Tokens: 297 input + 110 output = 407 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	20 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101, 107-109, 113, 115)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

AI ASSESSMENT

Scenario: Test that 'force_refresh' bypasses cache and returns a new token.

Why Needed: To ensure the TokenRefresher function behaves as expected when the 'refresh_interval' is set to a value other than 3600 (default),

Key Assertions:

- The function correctly returns a new token after force refresh.
- The function does not return the cached token.
- The function increments the call count correctly.
- The function sets the 'call_count' variable correctly.
- The function updates the 'stdout' and 'stderr' attributes of the subprocess result correctly.
- The function returns a CompletedProcess object with an empty stdout and stderr attribute.
- The function does not raise any exceptions when force refreshing a token.

Confidence: 80%

Tokens: 346 input + 167 output = 513 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/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 of the `TokenRefresher` constructor is set to `

Confidence: 80%

Tokens: 303 input + 117 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/token_refresh.py	29 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132-135, 139, 143-144, 148)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh.py::TestTokenRefresher::test_get_token_json_format

1ms



3

AI ASSESSMENT

Scenario: The test verifies that the `TokenRefresher` extracts a JSON object from the output of the `get-token` command.

Why Needed: This test prevents a potential bug where the extracted token is not in the expected JSON format, potentially leading to incorrect usage or errors.

Key Assertions:

- The output of the `get-token` command should be a JSON object with a single key-value pair: `token: 'json-token-value'` and an optional `expires_in` value.
- The extracted token should have the correct type (string) and value ('json-token-value').
- The presence of an `expires_in` value is optional and should be ignored if present.
- The JSON output should not contain any other keys or values that are not relevant to the token extraction process.
- The JSON output should be a valid JSON string, with no syntax errors or unexpected characters.
- The extracted token should have the correct format (e.g., 'Bearer ') if present in the output.
- The `json_key` parameter is set correctly and does not affect the expected output.

Confidence: 80%

Tokens: 308 input + 244 output = 552 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	29 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132-135, 139, 143-144, 148)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh.py::TestTokenRefresher::test_get_token_text_format

1ms



3

AI ASSESSMENT

Scenario: Test that the `TokenRefresher` extracts the correct text format from the output of the `get-token` command.

Why Needed: This test prevents a potential bug where the extracted token is not in the expected text format, potentially leading to incorrect usage or interpretation of the token.

Key Assertions:

- The output of the `get-token` command contains the string 'my-secret-token'.
- The output does not contain any non-text characters (e.g., lines starting with '#', etc.).
- The extracted token is in the same case as it appears in the original text.
- The extracted token has a length of at least 3 characters.
- The extracted token contains only alphanumeric characters and underscores.
- The extracted token does not contain any whitespace characters.
- The output contains no non-ASCII characters (e.g., emojis, etc.).
- The output is a single line of text.

Confidence: 80%

Tokens: 298 input + 206 output = 504 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	25 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh.py::TestTokenRefresher::test_invalid_json

1ms



AI ASSESSMENT

Scenario: Test that `TokenRefresher` raises an error when given invalid JSON input.

Why Needed: This test prevents a potential bug where the TokenRefresher class incorrectly handles or raises an exception on receiving invalid JSON data.

Key Assertions:

- The function `get_token()` in `TokenRefresher` should raise a `TokenRefreshError` when given invalid JSON input.
- The error message returned by `get_token()` should contain the string 'json'.
- The test should fail if the `get_token()` method is called with an invalid JSON argument.

Confidence: 80%

Tokens: 299 input + 131 output = 430 total

COVERAGE

src/pytest_llm_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 returns the expected tokens after a refresh.

Why Needed: This test prevents a potential bug where the TokenRefresher does not clear its cache after a refresh, leading to stale token values being returned.

Key Assertions:

- The function `get_token()` of the `TokenRefresher` instance should return two different tokens after a 1-hour refresh.
- The function `invalidate()` of the `TokenRefresher` instance should clear its cache and return 2.
- The call count variable should be incremented by 2 when calling `invalidate()`.
- The returned token from `get_token()` should not be equal to the first token before a refresh.
- The returned token from `get_token()` should be different from the second token after a refresh.
- The output of `run()` should contain two tokens separated by a space.
- The error message from `run()` should be empty.

Confidence: 80%

Tokens: 340 input + 211 output = 551 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	28 lines (ranges: 59-60, 63, 69-72, 83, 85-86, 90, 93-98, 101, 107-108, 111, 132, 153-154, 156, 160-162)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh.py::TestTokenRefresher::test_missing_json_key 1ms 3

AI ASSESSMENT

Scenario: Test that TokenRefresher raises an error when JSON key is missing.

Why Needed: To prevent a potential bug where the token refresh fails due to a missing required JSON key.

Key Assertions:

- The `get_token` method of the `TokenRefresher` class should raise a `TokenRefreshError` with a message indicating that the 'token' key is not found.
- The error message should be in lowercase and contain the word 'not'.
- The error message should indicate that the token was not found.
- The error message should specify the required JSON key as 'token'.

Confidence: 80%

Tokens: 325 input + 144 output = 469 total

COVERAGE

src/pytest_llm_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: Testing thread safety of TokenRefresher by verifying that it returns the same token across multiple threads.

Why Needed: This test prevents a potential bug where multiple threads accessing the TokenRefresher instance concurrently could result in different tokens being returned, leading to inconsistencies in the data.

Key Assertions:

- All threads should get the same token (first one to acquire lock)
- The length of the set of results should be equal to 1
- The first element of the results list should match 'token-1'
- Multiple threads accessing the TokenRefresher instance concurrently should not return different tokens
- The output format of the returned token should be consistent across all threads

Confidence: 80%

Tokens: 427 input + 154 output = 581 total

COVERAGE

src/pytest_llm_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.

Why Needed: This test prevents a potential bug where the token refresh process times out and fails without providing any meaningful error message.

Key Assertions:

- A `TimeoutExpired` exception is raised with a message indicating that the command timed out.
- The `get_token()` method of the `TokenRefresher` instance raises an exception with a string containing 'timed out'.
- The test asserts that the error message contains the word 'timed out' in lowercase.

Confidence: 80%

Tokens: 279 input + 127 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/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: Verify that TokenRefresher caches tokens and doesn't call command again.

Why Needed: This test prevents a potential bug where the TokenRefresher calls the command multiple times due to caching.

Key Assertions:

- The function `get_token()` returns the same token when called with the same arguments.
- The `TokenRefresher` instance is only called once, even if it's called multiple times in a short period of time.
- No additional output or error messages are produced for repeated calls to `get_token()`.

Confidence: 80%

Tokens: 353 input + 123 output = 476 total

COVERAGE

src/pytest_llm_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



AI ASSESSMENT

Scenario: Test the TokenRefresher's behavior when a command fails with no stderr output.**Why Needed:** This test prevents regression where the TokenRefresher incorrectly handles commands that do not produce any stderr output.**Key Assertions:**

- The `get_token` method of the `TokenRefresher` class raises a `TokenRefreshError` when the command fails with no stderr output.
- The error message returned by the `get_token` method includes 'exit 1' as part of it.
- The error message also explicitly states that there is no error output produced by the command.

Confidence: 80%**Tokens:** 322 input + 137 output = 459 total

COVERAGE

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

Why Needed: Test handling of empty command string.

Key Assertions:

- {'assertion': "The function should raise a TokenRefreshError with the message 'empty' when given an empty command string.", 'expected_value': 'empty'}

Confidence: 80%

Tokens: 151 input + 75 output = 226 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	11 lines (ranges: 59-60, 63, 69, 83, 85-86, 90-91, 113, 115)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_invalid_command_string

1ms



AI ASSESSMENT

Scenario: Test the `test_invalid_command_string` function to verify it handles an invalid command string (shlex parse error).

Why Needed: Prevent a `TokenRefreshError` when encountering an invalid command string during token refresh.

Key Assertions:

- The `'TokenRefresher'` constructor should raise a `'TokenRefreshError'` when given an invalid shell syntax in the `'command'` argument.
- The `'get_token()'` method of the `'TokenRefresher'` instance should return a `'TokenRefreshError'` exception when encountering an invalid command string.
- The error message returned by the `'get_token()'` method should contain the phrase 'Invalid command string'.
- The test should fail with a `'TokenRefreshError'` exception when calling `'refresher.get_token()'`.
- The `'refresher'` instance is expected to be in an invalid state after encountering an invalid command string.
- The `'refresh_interval'` and `'output_format'` arguments are not affected by the presence of an invalid command string.

Confidence: 80%

Tokens: 251 input + 218 output = 469 total

COVERAGE

src/pytest_llm_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 that the TokenRefresher raises a TokenRefreshError when receiving non-dict JSON output.

Why Needed: This test prevents regression where the TokenRefresher incorrectly handles JSON output as a list instead of a dict.

Key Assertions:

- The function `get_token()` should raise a `TokenRefreshError` with an error message indicating that the output is not a JSON object.
- The function `get_token()` should assert that the error message contains the string 'Expected JSON object'.
- The function `get_token()` should assert that the error message contains the string 'list'.

Confidence: 80%

Tokens: 328 input + 137 output = 465 total

COVERAGE

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



AI ASSESSMENT

Scenario: Test handling when token value is an empty string.**Why Needed:** Prevents regression where the TokenRefresher incorrectly handles empty or non-string token values.**Key Assertions:**

- The 'token' key in the response should be empty or a string.
- The 'token' key in the response should not contain any whitespace characters.
- The error message should indicate that the token value is either empty or not a string.
- The output of the subprocess command should include an empty or non-string token value.
- The output of the subprocess command should not include any whitespace characters in the 'token' key.

Confidence: 80%**Tokens:** 324 input + 144 output = 468 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	30 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101, 107-108, 111, 113, 115, 132-135, 139, 143-146, 149)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_json_token_not_string

1ms



3

AI ASSESSMENT

Scenario: Test that the TokenRefresher handles non-string token values correctly.

Why Needed: Prevents a potential bug where the TokenRefresher incorrectly handles tokens with non-string values, potentially leading to unexpected behavior or errors.

Key Assertions:

- The `get_token` method of the `TokenRefresher` instance raises a `TokenRefreshError` when it encounters an empty string or a non-string token value.
- The error message returned by the `get_token` method includes the phrase 'empty or not a string'.
- When a non-string token value is passed to the `get_token` method, the `json.dumps` function is used to convert it to JSON, which may result in an empty string if the token value cannot be converted.
- The `json_key` parameter of the `TokenRefresher` instance is set to 'token', which means that any non-string token values will be ignored and not returned by the `get_token` method.
- The `output_format` parameter of the `TokenRefresher` instance is set to 'json', which indicates that only JSON output should be produced. If a non-string token value is passed, it will not be included in the output.
- The `json_key` and `output_format` parameters are used together to ensure that only valid token values are returned by the `get_token` method.
- If a non-string token value is encountered, the `TokenRefreshError` exception will be raised with an error message indicating that the token value was empty or not a string.
- The test verifies that the `TokenRefresher` instance correctly handles non-string tokens by asserting that it raises a `TokenRefreshError` exception when such a value is passed to the `get_token` method.

Confidence: 80%

Tokens: 326 input + 385 output = 711 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	30 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 101, 107-108, 111, 113, 115, 132-135, 139, 143-146, 149)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_oserror_on_execution

1ms



AI ASSESSMENT

Scenario: Test the TokenRefresher's handling of OSError when executing a command.

Why Needed: Prevents the test from passing if an OSError is raised during execution, which could indicate a bug or regression.

Key Assertions:

- The `get_token()` method raises a `TokenRefreshError` with a message indicating that the command was not found.
- The error message includes the string 'Failed to execute'.
- The `pytest.raises(TokenRefreshError)` assertion checks if the `TokenRefreshError` is raised correctly.

Confidence: 80%

Tokens: 280 input + 122 output = 402 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	19 lines (ranges: 59-60, 63, 69, 83, 85-86, 90, 93-98, 113, 115-118)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_text_only_whitespace_lines

1ms



3

AI ASSESSMENT

Scenario: Test handling when text output has only whitespace lines after initial strip.

Why Needed: Prevents TokenRefreshError due to incorrect parsing of text with only blank lines after the initial strip.

Key Assertions:

- The `_parse_output` method should raise a `TokenRefreshError` when given input that has only non-empty lines after the initial strip.
- The output wrapper should contain non-whitespace content but only whitespace lines in the parsed output.
- The parsing method should correctly handle text with only blank lines after the initial strip and return an error.
- The test should fail when given input that meets these conditions to ensure the correct behavior of `TokenRefresher`.
- The output wrapper should contain non-whitespace content but only whitespace lines in the parsed output.
- The parsing method should correctly handle text with only blank lines after the initial strip and return an error.
- The test should pass when given input that meets these conditions to ensure the correct behavior of `TokenRefresher`.

Confidence: 80%

Tokens: 376 input + 213 output = 589 total

COVERAGE

src/pytest_llm_report/collector.py	14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)
src/pytest_llm_report/llm/token_refresh.py	4 lines (ranges: 132, 153-155)
src/pytest_llm_report/plugin.py	6 lines (ranges: 558-559, 562, 566-568)

PASSED

tests/test_token_refresh_coverage.py::TestTokenRefresherEdgeCases::test_whitespace_only_command

1ms



3

AI ASSESSMENT

Scenario: Test the test_whitespace_only_command to ensure it correctly raises a TokenRefreshError for an empty command string.

Why Needed: To prevent a potential bug where the TokenRefresher is unable to handle whitespace-only commands that do not contain any tokens.

Key Assertions:

- The function `get_token()` should raise a `TokenRefreshError` when given an empty command string.
- The error message should include the word 'empty' and be case-insensitive (e.g., 'empty token').
- The test assertion should verify that the error message contains the word 'empty'.
- The function should not raise a `TokenRefreshError` when given a non-empty command string.
- The test assertion should verify that no error is raised for a non-empty command string.
- The function should return an empty token (or None) when given a non-empty command string.
- The test assertion should verify that the returned token is indeed empty (or None).
- The function should handle whitespace-only commands correctly and raise the expected `TokenRefreshError` instance.

Confidence: 80%

Tokens: 236 input + 237 output = 473 total

COVERAGE

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

4ms



AI ASSESSMENT

Scenario: Test token usage aggregation with mock stash and pytest config to prevent early binding issues.

Why Needed: This test prevents regression in token usage aggregation when using a mock stash and pytest config to avoid early binding issues.

Key Assertions:

- Verify that the total input tokens, total output tokens, and annotations count are correctly calculated for each test.
- Verify that the terminal summary is run with the correct information.
- Verify that the llm_info dictionary contains the expected values for total_input_tokens, total_output_tokens, and annotations_count.

Confidence: 80%

Tokens: 775 input + 124 output = 899 total

COVERAGE

src/pytest_llm_report/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)