# Test Report

Run ID: 20981777763-py3.12 • Generated: 2026-01-14 04:03:28 • Duration: 286.53s
**Plugin:** v0.1.0 (2f498263985a34902252c53c11fb820445bd8f21) [dirty]
**Repo:** v0.1.1 (b7a157f6cb9189cc50a17c846484c8454deeac61)
**LLM:** ollama / llama3.2:1b (minimal context, 386 annotated)

91.09%
Total Coverage

| | | | |
|---|---|---|---|
| 387<br>TOTAL TESTS | 387<br>PASSED | 0<br>FAILED | 0<br>SKIPPED |
| 0<br>XFAILED | 0<br>XPASSED | 0<br>ERRORS | |

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

AI ASSESSMENT

**Scenario:** Test the aggregation function with all policy to aggregate all reports.

**Why Needed:** This test prevents a potential regression where only one report is aggregated and the other is discarded.

**Key Assertions:**

- The 'all' policy should be applied to both retained reports.
- Both retained reports should have the same outcome (in this case, passed).
- The aggregate function should return at least two tests.
- No test should be skipped or lost during aggregation.
- All retained reports should be included in the final result.
- The 'all' policy should not affect the order of the aggregated reports.
- The aggregate function should handle duplicate node IDs without issues.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/aggregation.py` | 69 lines (ranges: 52, 55-56, 59, 61-63, 73-74, 77-80, 84, 87-89, 93-100, 109-110, 113-117, 119, 125, 127-128, 130-131, 134, 141, 146, 148-153, 155, 157-159, 170, 217, 219-223, 235, 245, 248-249, 251, 253, 276-279, 281) |
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Verify that the aggregate function does not throw an exception when a non-existent directory is provided.

**Why Needed:** Prevents a potential bug where the aggregate function throws an exception or returns incorrect results when a non-existent directory is passed as input.

**Key Assertions:**

- The `aggregate` method should return `None` when a non-existent directory is provided.
- The `aggregate` method should not throw an exception when a non-existent directory is provided.
- The `aggregate` method should return the correct results for a non-existent directory.
- The `aggregate` method should raise an error with a descriptive message when a non-existent directory is provided.
- The `aggregate` method should handle non-existent directories correctly and do not throw exceptions or return incorrect results.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/aggregation.py | 7 lines (ranges: 52, 55-57, 109-111) |
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that the `aggregate` function picks the latest policy for a given test case and report.

**Why Needed:** This test prevents regression where the aggregate function fails to pick the correct latest policy when comparing different runs of the same test with the same configuration.

**Key Assertions:**

- The `aggregate` function should return the latest policy for the given test case and report.
- The number of tests in the result should be 1.
- The outcome of the first test in the result should be 'passed'.
- The `run_meta` object should have a `is_aggregated` attribute set to True.
- The `run_meta.run_count` attribute should be equal to 2.
- The `summary.passed` attribute should be equal to 1 (i.e., the first test passed).
- The `summary.failed` attribute should be equal to 0 (i.e., no tests failed).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/aggregation.py | 77 lines (ranges: 52, 55-56, 59, 64, 69, 73-74, 77-80, 84, 87-89, 93-100, 109-110, 113-117, 119, 125, 127-128, 130-131, 134, 141, 146, 148-153, 155, 157-159, 170, 182, 184-188, 190-191, 194, 217, 219-223, 235, 245, 248-249, 251, 253, 276-279, 281) |
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that aggregate function returns None when no directory configuration is provided.

**Why Needed:** Prevents regression in case the user forgets to configure an aggregation directory.

**Key Assertions:**

- agg.aggregate() should return None if mock_config.aggregate_dir is None.
- agg.aggregate() should not raise an error or any other exception when called with a None aggregate_dir.
- mock_config.aggregate_dir should be set to None before calling agg.aggregate().
- The aggregate function should not perform any aggregation operation when the directory is not configured.
- No error message or warning should be printed when aggregating without a directory configuration.
- The test should fail if mock_config.aggregate_dir is not None but agg.aggregate() is called with a None argument.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/aggregation.py` | `3 lines (ranges: 44, 52-53)` |
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

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

AI ASSESSMENT

**Scenario:** Verify that the aggregate function returns None when no reports exist or are not found.

**Why Needed:** This test prevents a potential bug where the aggregate function throws an exception when it cannot find any reports.

**Key Assertions:**

- The `aggregate()` method should return `None` when there are no reports to aggregate.
- A report is expected to be present in the directory before calling `aggregate()`.
- The `glob()` function should not return any results when searching for reports.
- No exception should be thrown by the `aggregate()` method when it cannot find any reports.
- The `pathlib.Path.exists()` function should return `True` when there are no reports to aggregate.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/aggregation.py` | `9 lines (ranges: 52, 55-57, 109-110, 113-114, 170)` |
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_aggregation.py::TestAggregator::test_aggregate_with_coverage_and_llm_annotations` 2ms 🛡️ 4

AI ASSESSMENT

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

**Why Needed:** Prevents regression in core functionality by ensuring accurate coverage and LLM annotation deserialization.

**Key Assertions:**

- coverage is correctly deserialized with the expected file paths and line ranges.
- LLM annotation is correctly deserialized with the expected scenario, why needed, and key assertions.
- The aggregated report can be re-serialized without issues.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/aggregation.py` | 81 lines (ranges: 52, 55-56, 59, 64, 69, 73-74, 77-80, 84, 87-89, 93-100, 109-110, 113-117, 119, 125, 127-128, 130-131, 134-137, 141-144, 146, 148-153, 155, 157-159, 170, 182, 184-188, 194, 217, 219-223, 235, 245, 248-249, 251, 253, 276-279, 281) |
| `src/pytest_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` | 32 lines (ranges: 40-43, 104-107, 109-111, 113, 115, 161-165, 167, 169, 171, 173, 176-180, 182, 184, 186, 188, 190) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that the `aggregate` function correctly aggregates source coverage for a single report.

**Why Needed:** This test prevents regression where the aggregated source coverage is not accurately calculated due to missing or incomplete reports.

**Key Assertions:**

- The `source_coverage` attribute of each `SourceCoverageEntry` in the result should be an instance of `SourceCoverageEntry`.
- Each `SourceCoverageEntry` in the result should have a `file_path` attribute matching the expected value.
- All statements in the source code should be covered by at least 83.33% of the total coverage.
- The number of missed statements should be less than or equal to the number of statements that are not covered.
- Each range of missing coverage should have a corresponding range of covered coverage.
- All ranges of covered and missed coverage should sum up to 100% of the total coverage.
- The `source_coverage` attribute of each `SourceCoverageEntry` in the result should be an instance of `SourceCoverageEntry` with the correct file path.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/aggregation.py` | 66 lines (ranges: 52, 55-56, 59, 64, 69, 73-74, 77-80, 84, 87-89, 93-100, 109-110, 113-117, 119, 125, 127-128, 148-155, 157-159, 170, 182, 184-186, 194, 217, 219-220, 235, 245, 248-249, 251, 253, 276-279, 281) |
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

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

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

**Key Assertions:**

- Verify that calling _load_coverage_from_source() returns None when llm_coverage_source is not set.
- Verify that calling _load_coverage_from_source() raises a UserWarning when llm_coverage_source does not exist.
- Verify that calling _load_coverage_from_source() correctly loads coverage from the configured source file (mocking coverage.py).
- Verify that the mock cov.report() returns the expected percentage value.
- Verify that the mock mapper.map_source_coverage() method is called with the correct entry.
- Verify that the mock cov.load() method is called once to load the data.
- Verify that the mock cov.report() method is called once to verify the coverage report.
- Verify that the result of _load_coverage_from_source() is not None and contains exactly one entry.
- Verify that the percentage value returned by _load_coverage_from_source() matches the expected value (80.0).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/aggregation.py | 19 lines (ranges: 245-246, 248-249, 251, 253-257, 259, 262-263, 265-266, 269, 271-272, 274) |
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

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

**Why Needed:** To prevent regression in the case of failed or skipped tests, where the total duration and coverage percentage may not be accurately reflected.

**Key Assertions:**

- The total number of tests is updated correctly to reflect the new count.
- The passed count is updated correctly to reflect the new count.
- The failed count is updated correctly to reflect the new count.
- The skipped count is updated correctly to reflect the new count.
- The xfailed count is updated correctly to reflect the new count.
- The xpassed count is updated correctly to reflect the new count.
- The error count is updated correctly to reflect the new count.
- The coverage percentage is preserved and accurately reflects the total number of tests.
- The total duration is updated correctly to reflect the time elapsed since the last summary update.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/aggregation.py` | 17 lines (ranges: 217, 219-233, 235) |
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test verifies that the test_skips_invalid_json function prevents skipping of reports with non-JSON files.

**Why Needed:** This test ensures that the aggregation function correctly handles invalid JSON reports and skips them, preventing potential data loss or inconsistencies.

**Key Assertions:**

- The `aggregate` function should not count any report as valid when it contains a non-JSON file.
- The `aggregate` function should raise a warning when it encounters a non-JSON report file.
- The test should fail when the `aggregate` function is called with an invalid JSON report file.
- The test should only count the first valid report in the aggregation result.
- The test should not count any reports that contain missing fields.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/aggregation.py | 71 lines (ranges: 52, 55-56, 59, 64, 69, 73-74, 77-80, 84, 87-89, 93-100, 109-110, 113-117, 119-120, 125, 127-128, 148-153, 155, 157-159, 162, 164-166, 168, 170, 182, 184-186, 194, 217, 219-220, 235, 245, 248-249, 251, 253, 276-279, 281) |
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

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

**Why Needed:** This test prevents a regression where the summary coverage is not calculated correctly for cases with multiple failed tests.

**Key Assertions:**

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

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/aggregation.py | 10 lines (ranges: 44, 217, 219-225, 235) |
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_annotator.py::TestAnnotateTests::test_cached_tests_are_skipped 2ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Verify that cached tests are skipped when annotating with a mock provider, cache and assembler.

**Why Needed:** This test prevents regression in the annotator's behavior when using mocks for providers, caches, or assemblers.

**Key Assertions:**

- The test verifies that the annotator skips caching of tests when using mocks for providers, caches, or assemblers.
- The test checks if the annotator correctly handles mocking of provider, cache and assembler objects.
- The test ensures that the annotator does not re-run cached tests when using mocks.
- The test verifies that the annotator skips caching of tests with mock providers, caches and assemblers.
- The test checks for any exceptions raised during caching of tests with mocks.
- The test verifies that the annotator correctly handles mocking of provider, cache and assembler objects in a test context.
- The test ensures that the annotator does not re-run cached tests when using mocks.

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 | 68 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-67, 71-72, 74-81, 87-92, 97-98, 100, 102, 104, 115-122, 129-135, 137, 139, 165-168, 170-171, 173-174, 176, 178, 180, 185-190, 192, 198, 203) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_annotator.py::TestAnnotateTests::test_concurrent_annotation 3ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Testing concurrent annotation with multiple providers and caches.

**Why Needed:** Prevents a potential memory leak by ensuring that annotations are not created concurrently.

**Key Assertions:**

- Verify that the annotator does not create new annotations while caching is in use.
- Ensure that cache hits are minimized when using multiple providers.
- Verify that the annotator does not attempt to annotate with a provider that has already been cached.
- Check for any unexpected annotation creation due to concurrent access.
- Verify that the annotator properly cleans up resources when caching is complete.
- Test that the annotator correctly handles cases where multiple annotations are created concurrently.

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 | 64 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-67, 71-72, 74-78, 87-92, 97-98, 100, 102, 104-112, 129-135, 137, 139, 229-232, 234, 236-237, 239, 245-246, 248-253, 255, 261, 266) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** The annotator handles failures when multiple annotations are performed concurrently.

**Why Needed:** This test prevents a potential bug where the annotator fails to handle concurrent annotation requests, leading to unexpected behavior or errors.

**Key Assertions:**

- mock_provider.assert_called_once_with('annotation', mock_assembler, mock_cache)
- mock_assembler.assert_called_once_with(mock_provider, 'annotation', mock_cache)
- mock_cache.assert_called_once_with('annotation')
- assert mock_provider.return_value.annotation == 'annotation'
- assert mock_assembler.return_value.annotation == 'annotation'
- assert mock_cache.return_value.annotation == 'annotation'
- assert mock_provider.return_value.cache == {}
- assert mock_assembler.return_value.cache == {}
- assert mock_cache.return_value.cache == {}

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 | 68 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-67, 71-72, 74-78, 87-92, 97-98, 100, 102, 104-112, 129-135, 137-139, 229-232, 234, 236-237, 239, 245-246, 248-253, 255, 261-264, 266) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_annotator.py::TestAnnotateTests::test_progress_reporting 2ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The `test_progress_reporting` function is used to verify the progress reporting mechanism of the annotator.

**Why Needed:** This test prevents regression in the annotator's progress reporting functionality.

**Key Assertions:**

- Verify that the `mock_provider`, `mock_cache`, and `mock_assembler` are properly mocked and not called.
- Check if the `progress_reporting` method of the `annotator` class is correctly implemented.
- Ensure that the `reportProgress` method is called with the expected arguments (e.g. provider, cache, assembler) during testing.
- Verify that the progress reporting is updated correctly in the UI after each iteration.
- Check if any exceptions are raised when attempting to report progress.
- Verify that the annotator's progress bar updates correctly on the dashboard.
- Ensure that the progress reporting is accurate and reflects the actual work being done by the annotator.

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 | 69 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-68, 71-72, 74-78, 87-92, 97-98, 100, 102, 104, 115-122, 129-135, 137, 139, 165-168, 170-171, 173-174, 176, 178, 180, 185-190, 192-195, 198, 203) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_annotator.py::TestAnnotateTests::test_sequential_annotation` 12.00s 🛡️ 5

AI ASSESSMENT

**Scenario:** The `test_sequential_annotation` function is being tested to verify its ability to annotate sequential data.

**Why Needed:** This test prevents regression in the sequential annotation functionality when using multiple annotators or assemblers concurrently.

**Key Assertions:**

- mock_provider, mock_cache and mock_assembler are all instances of `MagicMock` objects.
- The `test_sequential_annotation` function is being called with at least three arguments.
- The `test_sequential_annotation` function is checking the state of each argument before calling it.
- The `test_sequential_annotation` function is asserting that each argument has a specific value or behavior.
- The `test_sequential_annotation` function is using assertions to verify the correctness of its inputs.
- The `test_sequential_annotation` function is testing for potential side effects or unexpected behavior in its arguments.

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 | 68 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-67, 71-72, 74-78, 87-92, 97-98, 100, 102, 104, 115-122, 129-135, 137, 139, 165-168, 170-171, 173-174, 176, 178, 180-183, 185-190, 192, 198, 203) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Testing the `test_skips_if_disabled` function to ensure it skips tests when LLM is disabled.

**Why Needed:** This test prevents regression in the annotator's behavior when the Large Language Model (LLM) is not enabled.

**Key Assertions:**

- The function `annotate_tests([], config)` should be called without any arguments, indicating that no annotation should occur.
- The configuration object `config` should have a 'provider' key set to 'none', indicating that the LLM is disabled.
- The `annotate_tests` function should not take any arguments, as specified in the test description.
- The `annotate_tests` function should not modify the test results or output.
- The `Config` class should have a `provider` attribute set to 'none' when creating an instance with `None` as the provider argument.
- The `test_skips_if_disabled` function should be able to verify that the annotator skips tests based on its configuration.
- The test should fail if the LLM is enabled and the `test_skips_if_disabled` function is called without any arguments, indicating a regression in the annotator's behavior.

COVERAGE

| | |
|---|---|
| src/pytest_llm_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: 45-46) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** The annotator should skip annotation if the provider is unavailable.

**Why Needed:** This test prevents a regression where the annotator fails to skip annotations when the provider is not available.

**Key Assertions:**

- Mocking `mock_provider` with an unavailable provider will prevent the annotator from skipping annotations.
- The `skip` method of the annotated object will be called if the provider is unavailable.
- The annotation process will still complete successfully even though the provider is not available.
- The annotator will skip annotations when the provider is unavailable, as expected.
- The error message for an unavailable provider will be logged to the console.
- The `mock_provider` object will have a `__call__` method that returns a mock response.
- The `mock_provider` object will not raise any exceptions when called with an unavailable provider.
- The annotator's behavior will remain consistent even when the provider is unavailable.

COVERAGE

| | |
|---|---|
| src/pytest_llm_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: 45, 48-52, 54) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_annotator_maximal.py::TestAnnotatorAdvanced::test_annotate_concurrent_with_progress_and_errors` 2ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test that annotator reports progress and first error when annotated concurrently with errors.

**Why Needed:** Prevents regression of annotator's behavior when handling concurrent annotations with errors.

**Key Assertions:**

- Verify that the annotator correctly reports a progress message for each task in the list.
- Ensure that the first annotation result contains an error message as expected.
- Confirm that the annotator appends 'first error' to the progress messages list.
- Verify that all tasks are annotated and failures are reported correctly.
- Check if any of the progress messages contain 'LLM annotation'.
- Verify that no other annotations were made before the first error is reported.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_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` | `28 lines (ranges: 229-232, 234, 236-237, 239-242, 245-246, 248-253, 255-258, 261-264, 266)` |
| `src/pytest_llm_report/options.py` | `2 lines (ranges: 107, 147)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_annotator_maximal.py::TestAnnotatorAdvanced::test_annotate_sequential_rate_limit_wait` 2ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Should wait if rate limit interval has not elapsed.

**Why Needed:** To prevent a potential issue where the annotator does not wait for the rate limit interval to elapse before proceeding with annotation tasks.

**Key Assertions:**

- The time.sleep function was called.
- The mock_time object's side_effect was set to [100.0, 100.1, 100.2, 100.3, 100.4]
- The time.sleep function did not call itself multiple times within the given interval.

COVERAGE

| | |
|---|---|
| src/pytest_llm_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 | 23 lines (ranges: 165-168, 170-171, 173-174, 176, 178, 180-183, 185-190, 192, 198, 203) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_annotator_maximal.py::TestAnnotatorAdvanced::test_annotate_tests_cached_progress` 2ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Should report progress for cached tests when annotating tests with maximal caching.

**Why Needed:** This test prevents regression where the annotator fails to report progress for cached tests, potentially leading to incorrect results or missed opportunities for optimization.

**Key Assertions:**

- The `get_provider` method of `LlmCache` returns a mock provider instance when called.
- The `assemble` method of `ContextAssembler` is called with the correct arguments (`src`, None) when called.
- Any message containing '(cache): test_cached' is appended to the `progress_msgs` list.

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 | 37 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-68, 71-72, 74-84, 97-98, 100, 127, 129-135, 137, 139) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_annotator_maximal.py::TestAnnotatorAdvanced::test_annotate_tests_provider_unavailable` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test that the annotator does not attempt to annotate tests when the provider is unavailable.

**Why Needed:** To prevent a potential error where the annotator tries to annotate tests without a valid provider.

**Key Assertions:**

- The function `annotate_tests` should return an empty list of results.
- The function `annotate_tests` should not attempt to print any messages or capture output when the provider is unavailable.
- The function `annotate_tests` should correctly skip annotations for tests that are not available.
- The function `annotate_tests` should handle the case where `is_available` returns False without raising an exception.
- The function `annotate_tests` should return a result with the correct outcome (in this case, 'passed') when the provider is unavailable.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_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: 45, 48-52, 54) |
| `src/pytest_llm_report/options.py` | 3 lines (ranges: 107, 147, 224) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** The test verifies that the `test_base_parse_response_malformed_json_after_extract` function will fail when an invalid JSON string is provided.

**Why Needed:** This test prevents a bug where the function incorrectly assumes valid JSON content, leading to incorrect error handling and potential crashes or unexpected behavior.

**Key Assertions:**

- The `annotation.error` attribute will be set to 'Failed to parse LLM response as JSON'.
- The `provider._parse_response(response)` call will raise a `JSONDecodeError` exception with the message 'Invalid JSON: invalid content'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 186-187, 190-191, 194-195, 220-221) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Tests that the `test_base_parse_response_non_string_fields` function handles non-string fields in the response data correctly.

**Why Needed:** This test prevents a potential bug where the function incorrectly assumes all fields are strings and throws an error for non-string values.

**Key Assertions:**

- The function should be able to parse the `scenario` field as an integer without throwing an error.
- The function should correctly identify the `why_needed` list containing only 'list'.
- The function should be able to extract the correct key from the response data, which is 'a' in this case.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 186-187, 190-191, 194-195, 198-200, 203-207, 212, 214-218) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Verify that the `get_gemini_provider` function returns a valid instance of `GeminiProvider`.

**Why Needed:** This test prevents a potential bug where the `get_gemini_provider` function may return an incorrect or None value if the Gemini provider is not configured correctly.

**Key Assertions:**

- The returned instance should be an instance of `GeminiProvider`.
- The returned instance should have the correct attributes (e.g. `name`, `url`, etc.).
- The returned instance should implement the `GeminiProvider` interface or inherit from it.
- The function should not return None or an incorrect value if the Gemini provider is configured correctly.
- The function should raise a suitable exception (e.g. `ValueError`) if the Gemini provider configuration is invalid.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 245, 247, 249, 252, 257, 262-263, 265) |
| src/pytest_llm_report/llm/gemini.py | 7 lines (ranges: 134, 136-139, 141-142) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Tests the `get_provider` function when an unknown LLM provider is specified.

**Why Needed:** This test prevents a ValueError from being raised when an invalid provider is provided to the `get_provider` function.

**Key Assertions:**

- The `get_provider` function should raise a `ValueError` with a message indicating that the specified provider is unknown.
- The error message should include the string 'Unknown LLM provider: invalid'.
- When an invalid provider is passed to `get_provider`, it should not return any value (i.e., it should be a no-op).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 245, 247, 249, 252, 257, 262, 267) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_base_maximal.py::TestGetProvider::test_get_litellm_provider 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Verify that the `get_litellm_provider` function returns a valid instance of `LiteLLMProvider`.

**Why Needed:** This test prevents a potential bug where the provider is not correctly initialized with the correct configuration.

**Key Assertions:**

- The returned value is an instance of `LiteLLMProvider`.
- The `provider` attribute of the returned value has the expected type and value.
- The `config` object passed to `get_provider` has a valid `provider` key with the correct value.
- The `LiteLLMProvider` class is correctly instantiated from the provider configuration.
- The provider's `name` attribute matches the expected value.
- The provider's `max_length` attribute is set to the expected value.
- The provider's `batch_size` attribute is set to the expected value.
- The provider's `num_workers` attribute is set to the expected value.
- The provider's `device` attribute matches the expected device type (e.g., CPU, GPU).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 245, 247, 249, 252, 257-258, 260) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Tests the `get_noop_provider` method with a configuration that returns a NoopProvider.

**Why Needed:** This test prevents a potential regression where a NoopProvider is returned unexpectedly when no provider is specified in the config.

**Key Assertions:**

- The function `get_provider(config)` should return an instance of `NoopProvider`.
- The `provider` attribute of the `NoopProvider` instance should be `None`.
- The `config` object passed to `get_provider(config)` should not have a `provider` attribute.
- The `get_provider(config)` function should raise a `ValueError` when given an invalid configuration.
- The `get_provider(config)` function should return the correct instance of `NoopProvider` for valid configurations.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 245, 247, 249-250) |
| src/pytest_llm_report/llm/noop.py | 1 lines (ranges: 32) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Verify that the `get_ollama_provider` method returns an instance of `OllamaProvider` when a valid provider is provided.

**Why Needed:** This test prevents a potential bug where the `get_ollama_provider` method does not return an instance of `OllamaProvider` when a valid provider is passed.

**Key Assertions:**

- The function should return an instance of `OllamaProvider`.
- The function should be able to successfully retrieve an Ollama provider instance from the configuration.
- The function should not raise any exceptions if a valid provider is provided.
- The function should correctly handle cases where the provider name is invalid or missing.
- The function should return `None` when no valid provider is found in the configuration.
- The function should be able to retrieve an Ollama provider instance with a specific name.
- The function should not throw any exceptions if the provider name is changed during execution.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 245, 247, 249, 252-253, 255) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Verify that the LLM Provider Defaults test case checks for available caches correctly.

**Why Needed:** This test prevents a regression where the LLM Provider Defaults test case fails to detect when the cache is not available.

**Key Assertions:**

- The `is_available()` method returns True for both cache availability and non-cache availability scenarios.
- The `checks` attribute of the provider instance is incremented correctly after calling `_check_availability()`.
- The `is_available()` method returns False when the cache is not available.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 107-108, 110-111) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** The `get_model_name()` method of the `ConcreteProvider` class should return the model name specified in the configuration.

**Why Needed:** Without this test, a bug or regression could occur where the default model name is not correctly set to the one provided in the configuration.

**Key Assertions:**

- The `get_model_name()` method of the `ConcreteProvider` class should return 'test-model'.
- The `model` attribute of the `Config` object passed to `get_model_name()` should be equal to 'test-model'.
- The `provider` object created with the `config` should have a `get_model_name()` method that returns 'test-model'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 136) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** The test verifies that the `get_rate_limits` method of a `ConcreteProvider` instance returns `None` when no rate limits are specified.

**Why Needed:** This test prevents a regression where the default rate limit for LLM providers is not correctly set to None.

**Key Assertions:**

- The `provider.get_rate_limits()` call should return `None`.
- The `rate_limit` attribute of the provider instance should be `None`.
- The `max_rate` and `min_rate` attributes of the provider instance should not be set to any value.
- The `default_rate_limit` attribute of the provider instance should be `None`.
- The `provider.get_rate_limits()` call should raise an exception with a meaningful error message if rate limits are specified.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.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: 52-53, 128) |
| `src/pytest_llm_report/options.py` | 2 lines (ranges: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Verifies that `is_local()` returns `False` when the LLM defaults to local mode.

**Why Needed:** Prevents a bug where the default LLM configuration does not set `is_local()` to `False`.

**Key Assertions:**

- config.is_local() is False
- provider.is_local() is False
- provider.is_local() == False
- not provider.is_local() == True
- provider.is_local() != Config.is_local()
- not provider.is_local() != False

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 147) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Testing the consistency of a hash function for the same source code.

**Why Needed:** Prevents a potential bug where different inputs to a hash function could produce different hashes, leading to unexpected behavior or data corruption.

**Key Assertions:**

- The hash of the input `source` should be equal to itself.
- The hash of the input `source` should not change even if it is modified (e.g., by adding or removing code).

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: 380-381, 384, 388-390)` |

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

AI ASSESSMENT

**Scenario:** The test verifies that the hash of two different functions with the same name but different implementations produces a different hash.

**Why Needed:** This test prevents a potential bug where two functions with the same name but different implementations produce the same hash, potentially leading to unexpected behavior or incorrect results in certain scenarios.

**Key Assertions:**

- The function `hash_source` should return a different hash for two different source strings.
- The function `hash_source` should not return the same hash when given the same input but with different implementations.
- The hash of `hash_source(test_a)` should be different from the hash of `hash_source(test_b)` even if `test_a` and `test_b` have the same implementation.
- If two functions `func1` and `func2` are defined as `def func1(): pass` and `def func2(): pass`, then `hash_source(func1)` should be different from `hash_source(func2).
- The hash of `hash_source(test_a())` should be different from the hash of `hash_source(test_b())` even if `test_a()` and `test_b()` have the same implementation.
- If two functions `func1` and `func2` are defined as `def func1(): pass`, then `hash_source(func1)` should not produce a hash that can be used to identify `func1` even after calling it multiple times.
- The function `hash_source(test_a)` should raise an exception when given invalid input, such as an empty string or a non-string value.

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: 380-381, 384, 388-390)` |

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

AI ASSESSMENT

**Scenario:** Verify the length of the hash generated by the HashSource.

**Why Needed:** Prevents a potential issue where the hash length is not consistent across different inputs.

**Key Assertions:**

- The hash should be at least 16 characters long.
- The hash should be no longer than 32 characters long.
- The hash should have a consistent length across different inputs.
- The hash should not be empty.
- The hash should contain only ASCII characters.
- The hash should not contain any non-ASCII characters.
- The hash should not contain any whitespace characters.

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: 380-381, 384, 388-390)` |

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

AI ASSESSMENT

**Scenario:** Test clears all cache entries after adding some initial values.

**Why Needed:** Prevents a regression where the test fails to clear cache due to an inconsistent state.

**Key Assertions:**

- Ensure that clearing the cache removes all existing entries.
- Verify that the cache is empty after clearing.
- Confirm that attempting to retrieve an entry from the cache returns None.
- Check that adding a new annotation after clearing does not create any additional entries.
- Verify that the cache's internal state is consistent before and after clearing.
- Ensure that the cache's size remains accurate after clearing.

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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that annotations with errors are not cached.

**Why Needed:** Prevents a potential regression where error annotations are cached and cause issues later.

**Key Assertions:**

- The cache should not store the annotation for 'test::foo' with key 'abc123' when it contains an error.
- The cache should return None for the retrieved annotation.
- The cache should not store any annotations with errors in the specified configuration.
- Error annotations are not cached in the test environment.
- No caching of error annotations is performed by default.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that `get` method returns `None` for missing entries in the cache.

**Why Needed:** Prevents a potential bug where the test fails when an entry is missing from the cache.

**Key Assertions:**

- The function should return `None` when trying to retrieve a non-existent key.
- The function should not raise any exceptions for missing keys.
- The function should handle the case where the cache directory does not exist.
- The function should ignore the cache directory if it is empty.
- The function should use the provided configuration to determine the cache directory.
- The function should handle cases where the key is not present in the cache.
- The function should return `None` instead of raising an exception for missing keys.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test the ability to store and retrieve annotations from the cache.

**Why Needed:** This test prevents a potential bypass attack by ensuring that the cache stores and retrieves annotations in a consistent manner.

**Key Assertions:**

- Verify that the annotation is stored correctly in the cache.
- Check if the annotation's status matches the expected value.
- Ensure that the confidence level of the retrieved annotation matches the original value.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test verifies that CollectionError has the correct nodeID and message.

**Why Needed:** Prevents a potential bug where CollectionError is incorrectly constructed with incorrect nodeID or message.

**Key Assertions:**

- assert error.nodeid == 'test_bad.py'
- assert error.message == 'SyntaxError'

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test verifies that an initial empty collection is returned when no errors are present.

**Why Needed:** Prevents a potential error where an empty collection is returned without any errors being detected.

**Key Assertions:**

- The function `get_collection_errors()` returns an empty list.
- No exceptions are raised or thrown by the `get_collection_errors()` method.
- The `get_collection_errors()` method does not attempt to raise an exception when no errors are present in the collection.
- The `get_collection_errors()` method only checks for the presence of any errors, without attempting to retrieve them.
- No error messages or details about the detected errors are provided by the `get_collection_errors()` method.
- The `get_collection_errors()` method does not attempt to handle cases where multiple errors are present in the collection.
- The `get_collection_errors()` method only checks for a single error type, without considering other potential issues.
- No indication is given that an empty collection indicates no errors or that there are no issues with the data.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test the default value of llm_context_override in TestCollectorMarkerExtraction.

**Why Needed:** This test prevents a regression where the default value of llm_context_override might be set to None unexpectedly.

**Key Assertions:**

- The llm_context_override attribute is not set to None for TestCaseResult instances with nodeid 'test.py::test_foo' and outcome 'passed'.
- llm_context_override is not equal to None for TestCaseResult instances with nodeid 'test.py::test_foo' and outcome 'failed'.
- llm_context_override is not equal to None for TestCaseResult instances with nodeid 'test.py::test_bar' and outcome 'passed'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that the default value of llm_opt_out is set to False.

**Why Needed:** Prevents a regression where the default value of llm_opt_out could be incorrectly set to True.

**Key Assertions:**

- The llm_opt_out attribute of TestCaseResult is set to False.
- The llm_opt_out attribute of TestCaseResult is not set to True.
- The llm_opt_out attribute of TestCaseResult is not set to False when the nodeid does not match 'test.py::test_foo'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** The `capture` feature of the output collector is not enabled by default.

**Why Needed:** This test prevents a regression where the output collector's default behavior is changed without proper notification.

**Key Assertions:**

- assert config.capture_failed_output is False
- assert isinstance(config, Config)
- assert hasattr(config, 'capture') and config.capture is False
- assert not hasattr(config, 'output_capture')
- assert not hasattr(config, 'output_capture_enabled')
- assert not hasattr(config, 'output_capture_type')
- assert not hasattr(config, 'output_capture_enabled')
- assert isinstance(config.output_capture, bool)

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** The test verifies that the default value of `capture_output_max_chars` in the `Config` class is 4000.

**Why Needed:** This test prevents a potential bug where the default max chars is set to an extremely high value (e.g., 10000), which could lead to unexpected behavior or errors when capturing output.

**Key Assertions:**

- assert config.capture_output_max_chars == 4000
- assert isinstance(config.capture_output_max_chars, int)
- config.capture_output_max_chars should be greater than or equal to 1 (default value)
- config.capture_output_max_chars should not exceed 10000 (exceeding default value could lead to unexpected behavior)

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test 'xfail failures should be recorded as xfailed' verifies that xfail failures are correctly reported as xfailed.

**Why Needed:** This test prevents regression where xfail failures are not properly recorded as expected failures.

**Key Assertions:**

- The `results` dictionary contains the correct key for the failed report, 'xfailed'.
- The value of `outcome` in the failed report matches the expected outcome 'xfailed'.
- The `wasxfail` attribute of the failed report correctly indicates that it was an xfail failure.
- The `duration` and `longrepr` attributes are set to correct values for the failed report.
- The `passed`, `skipped`, and `when` attributes are all correctly initialized with default values.
- The `nodeid` attribute matches the expected value 'test_xfail.py::test_expected_fail'.
- The `config` attribute is not explicitly set, but it should be for a valid test collector instance.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** The test verifies that xfail passes are correctly recorded as xpassed.

**Why Needed:** This test prevents regression where an unexpected pass is not properly marked as xpassed.

**Key Assertions:**

- The `when` field of the report should be set to 'call' when an xfail occurs.
- The `passed` field of the report should be set to `True` when an xfail passes.
- The `failed` field of the report should be set to `False` when an xfail fails.
- The `skipped` field of the report should be set to `False` when an xfail is skipped.
- The `duration` field of the report should be set to a value greater than 0 (in this case, 0.01 seconds)
- The `longrepr` field of the report should be empty (in this case, an empty string)
- The `wasxfail` field of the report should match the expected failure message ('expected failure')

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

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

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

**Key Assertions:**

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

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that the `get_results` method returns sorted results by node ID.

**Why Needed:** This test prevents a regression where the order of results is not preserved due to manual sorting.

**Key Assertions:**

- The list of node IDs in the results should be in ascending order.
- The list of node IDs in the results should contain only 'a_test.py::test_a' and 'z_test.py::test_z'.
- The list of node IDs in the results should not contain any other nodes than 'a_test.py::test_a' and 'z_test.py::test_z'.

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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that the `handle_collection_finish` method correctly tracks collected and deselected counts.

**Why Needed:** This test prevents a regression where the count of collected items is not updated correctly when an item is deselected.

**Key Assertions:**

- The `collected_count` attribute should be set to 3 after calling `handle_collection_finish` with 3 collected items and 1 deselected item.
- The `deselected_count` attribute should be set to 1 after calling `handle_collection_finish` with 3 collected items and 1 deselected item.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that `capture_output` is disabled when `config.capture_failed_output=False` and `handle_runtest_logreport` is used.

**Why Needed:** To prevent capturing of output in test cases where `capture_output` is not enabled (e.g., integration tests via handle_runtest_logreport).

**Key Assertions:**

- The collector should not capture the output of the test.
- The report nodeid should be set to 't'.
- The outcome of the report should be 'failed'.
- The when field of the report should be set to 'call'.
- The passed field of the report should be False.
- The failed field of the report should be True.
- The skipped field of the report should be False.
- The capstdout field of the report should be set to 'output'.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that the `TestCollector` captures stderr correctly when `capture_failed_output=True`.

**Why Needed:** This test prevents a potential bug where the `TestCollector` does not capture stderr, potentially leading to silent failures or incorrect reporting.

**Key Assertions:**

- The output of `report.capstderr` is set to 'Some error'.
- The `captured_stderr` attribute of the `result` object is set to 'Some error'.

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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test the `test_capture_output_stdout` method to ensure it captures stdout correctly.

**Why Needed:** This test prevents a potential bug where the collector does not capture stdout and reports an empty string instead.

**Key Assertions:**

- The `captured_stdout` attribute of the `TestCaseResult` object is set to 'Some output'.
- The `report.capstdout` method was called with the argument 'Some output'.
- The `report.capstderr` method was not called (i.e., it returns an empty string).

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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that the `test_capture_output_truncated` test verifies that output is truncated when it exceeds the maximum allowed characters.

**Why Needed:** This test prevents a potential bug where the collector fails to truncate output exceeding the max chars limit, causing unexpected behavior or errors.

**Key Assertions:**

- The captured stdout should be truncated to 10 characters (or less) if it exceeds this limit.
- The captured stderr should not be affected by this truncation.
- The `test_capture_output_truncated` test should pass without any exceptions when the output is truncated.
- The collector's behavior should not change even after exceeding the max chars limit.
- The collector's error message should indicate that the output was truncated.
- The captured stdout and stderr should be consistent with this truncation.
- The `TestCollector` class should correctly implement the `capture_output_max_chars` attribute.

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: 107, 147)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

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

AI ASSESSMENT

**Scenario:** Test that the collector extracts item markers correctly for create_result with item_markers.

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

**Key Assertions:**

- item.callspec.id should be set to 'param1' when calling get_closest_marker('llm_opt_out')
- item.get_closest_marker('llm_context') should return a MagicMock object with args ['complete']
- result.param_id should match the value of item.callspec.id
- result.llm_opt_out should be set to True based on the marker 'llm_opt_out'
- result.llm_context_override should be set to 'complete' based on the marker 'llm_context'
- result.requirements should contain the values 'REQ-1', 'REQ-2'

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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that the `collectors` module correctly handles ReprFileLocation cases when creating a crash report.

**Why Needed:** This test prevents a potential regression where the `collectors` module might not handle ReprFileLocation correctly, potentially causing unexpected behavior or errors in crash reports.

**Key Assertions:**

- The `_extract_error` method of the `TestCollector` class should return the expected error message when a `ReprFileLocation` is used.
- The `longrepr` attribute of the `report` object should be set to the expected string value.
- The `__str__` method of the `longrepr` object should return the expected string value.
- The `_extract_error` method should not raise an exception when a `ReprFileLocation` is used.
- The `collectors` module should correctly handle ReprFileLocation cases without crashing or producing unexpected results.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test the `_extract_error` method of `TestCollector` to ensure it returns a string that matches the expected longrepr.

**Why Needed:** This test prevents a potential regression where the extracted error string is not correctly formatted, potentially leading to incorrect reporting or further errors.

**Key Assertions:**

- The value returned by `_extract_error(report)` is equal to `Some error occurred`.
- The type of the value returned by `_extract_error(report)` is `str`.
- The value returned by `_extract_error(report)` contains the string `'Some error occurred'` exactly.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test the `_extract_skip_reason` method of `TestCollector` when no longrepr is provided.

**Why Needed:** To prevent a potential bug where the method returns `None` unexpectedly when no longrepr is available.

**Key Assertions:**

- The `_extract_skip_reason` method should return `None` if `report.longrepr` is `None`.
- The `_extract_skip_reason` method should not raise an exception or throw any error when `report.longrepr` is `None`.
- The `_extract_skip_reason` method should correctly handle the case where `report.longrepr` is `None` and return `None` without any side effects.
- The `_extract_skip_reason` method should not modify the original `report` object in place when `report.longrepr` is `None`.
- The `_extract_skip_reason` method should preserve the original behavior of returning `None` when `report.longrepr` is `None` from previous test cases.
- The `_extract_skip_reason` method should not throw any warning or error message when `report.longrepr` is `None`.
- The `_extract_skip_reason` method should correctly handle the case where `report.longrepr` is `None` and return `None` without raising an exception.
- The `_extract_skip_reason` method should be able to handle different types of report objects, including those with or without longrepr.

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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test the `_extract_skip_reason` method of `TestCollector`.

**Why Needed:** The test prevents a potential bug where the `longrepr` attribute is not correctly extracted from the report.

**Key Assertions:**

- The `report.longrepr` attribute should be set to 'Just skipped'.
- The `collector._extract_skip_reason(report)` method should return 'Just skipped' as the skip reason string.
- The `_extract_skip_reason` method should correctly handle cases where the `longrepr` attribute is not present in the report.
- The test should fail if the `report.longrepr` attribute is not set or is an empty string.
- The `collector._extract_skip_reason(report)` method should raise a `AssertionError` if the skip reason string is not 'Just skipped'.
- The `_extract_skip_reason` method should correctly handle cases where the report object is missing any attributes, including `longrepr`.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that `collectors._extract_skip_reason` correctly extracts skip message from a tuple containing file, line and message.

**Why Needed:** The test prevents a potential bug where the skip reason is not extracted correctly from tuples containing file, line and message.

**Key Assertions:**

- The tuple `(file, line, message)` should be converted to `('test_file.py', 10, 'Skipped for reason')` before being passed to `_extract_skip_reason`.
- The string representation of the tuple `('test_file.py', 10, 'Skipped for reason')` should match the expected output from `_extract_skip_reason`.
- The `_extract_skip_reason` function should be able to correctly extract the skip message from the tuple containing file, line and message.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

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

**Why Needed:** This test prevents a potential issue where the collector does not handle collection reports correctly and may silently ignore or fail to log these errors.

**Key Assertions:**

- The length of `collector.collection_errors` is set to 1.
- The nodeid in `collector.collection_errors[0]` matches the expected value 'test_broken.py'.
- The message in `collector.collection_errors[0]` matches the expected value 'SyntaxError'.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

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

**Why Needed:** This test prevents regression where a rerun attribute is not handled correctly, potentially leading to incorrect results or failures in subsequent runs.

**Key Assertions:**

- The `rerun_count` of the result 't::r' should be 1 after rerunning.
- The final outcome of the result 't::r' should be 'failed' after rerunning.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test 'handle_runtest_setup_failure' verifies that the TestCollector correctly records setup errors and prevents regression.

**Why Needed:** This test prevents a potential regression in the TestCollector's behavior when runtest setup fails, ensuring consistency with expected output.

**Key Assertions:**

- The report is created with the correct nodeid 't::f' and failed status.
- The report indicates an error phase and a specific error message 'Setup failed'.
- The collector correctly extracts the outcome from the report as 'error'.

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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test verifies that the Collector correctly records an error when teardown fails after a pass.

**Why Needed:** This test prevents regression in the Collector's behavior when it encounters a teardown failure after a successful run.

**Key Assertions:**

- The `teardown` report is not skipped and has a long representation indicating a cleanup failed.
- The error message 'Cleanup failed' is correctly recorded as the outcome of the test.
- The phase of the test is set to 'teardown'.
- The error message is not empty, indicating that an error occurred during teardown.

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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test the parsing of edge cases for Gemini models, including an empty list and 'all' model.

**Why Needed:** This test prevents regression in case the `gemini_model_parsing` method is modified to handle edge cases without proper error handling or assertions.

**Key Assertions:**

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

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.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: 52-53) |
| `src/pytest_llm_report/llm/gemini.py` | 17 lines (ranges: 134, 136-139, 141-142, 385, 387, 417-424) |
| `src/pytest_llm_report/options.py` | 2 lines (ranges: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Verify that the rate limiter does not allow a request when there are no tokens available, and also verify that it allows a request when there are enough tokens to cover the next available time.

**Why Needed:** This test prevents regression in case of an edge case where the rate limiter is under both limits (i.e., no tokens and no requests).

**Key Assertions:**

- The function `next_available_in` should return a non-zero value when there are enough tokens to cover the next available time.
- The function `next_available_in` should not allow a request when there are no tokens available.
- The function `next_available_in` should allow a request when there are enough tokens to cover the next available time and there is at least one token left.
- The function `next_available_in` should return 0 when there are no tokens available.

COVERAGE

| | |
|---|---|
| src/pytest_llm_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: 380-381, 384, 388-390) |

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

AI ASSESSMENT

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

**Why Needed:** This test prevents regression in coverage calculation for models with varying numbers of statements.

**Key Assertions:**

- The 'coverage_percent' key in the returned dictionary is set to 50.0 (or its decimal representation).
- The 'error' key in the returned dictionary matches the expected error message ('timeout').

COVERAGE

| | |
|---|---|
| `src/pytest_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` | 46 lines (ranges: 71-78, 104-107, 109, 111-113, 115, 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Verify that a new `CoverageMapper` instance is created with the provided configuration.

**Why Needed:** Prevents a potential bug where the `CoverageMapper` instance is not properly initialized with the correct configuration.

**Key Assertions:**

- The `config` attribute of the `CoverageMapper` instance should be set to the provided `Config` object.
- The `warnings` attribute of the `CoverageMapper` instance should be an empty list.

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** The `get_warnings` method in the `CoverageMapper` class should be able to return a list of warnings.

**Why Needed:** This test prevents a potential issue where the method might not return any warnings if there are no warnings available.

**Key Assertions:**

- assert isinstance(warnings, list)
- warnings is expected to be a list
- all(warnings) should be true

COVERAGE

| | |
|---|---|
| `src/pytest_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: 107, 147)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_coverage_map.py::TestCoverageMapper::test_map_coverage_no_coverage_file` 1ms 🛡️ 5

AI ASSESSMENT

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

**Why Needed:** Prevents a regression where the test fails due to missing coverage data.

**Key Assertions:**

- The function should return an empty dictionary even if there are no files with coverage information.
- The `map_coverage` method should not raise any exceptions when no coverage file is present.
- No warnings or errors should be raised in this scenario.

COVERAGE

| | |
|---|---|
| `src/pytest_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: 139-142) |
| `src/pytest_llm_report/options.py` | 2 lines (ranges: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test should extract node ID for all phases when include_phase='all'.

**Why Needed:** This test prevents a regression where the coverage map does not include all phases.

**Key Assertions:**

- The function _extract_nodeid() includes the specified phase in the node ID.
- The function _extract_nodeid() excludes the 'setup' phase from the node ID.
- The function _extract_nodeid() includes the 'teardown' phase in the node ID.
- The function _extract_nodeid() does not include any phases in the node ID when include_phase='all'.
- The function _extract_nodeid() excludes all phases except for the specified one from the node ID.
- The function _extract_nodeid() includes only the specified phase in the node ID, excluding other phases.
- The function _extract_nodeid() correctly handles cases where 'setup' or 'teardown' is not present in the code.
- The function _extract_nodeid() returns the correct node ID for all phases when include_phase='all'.

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test the `extract_nodeid` method with an empty context.

**Why Needed:** Prevents a potential bug where the method returns `None` when given an empty context.

**Key Assertions:**

- The `_extract_nodeid` method should return `None` for an empty string.
- The `_extract_nodeid` method should return `None` for an empty integer.
- The `_extract_nodeid` method should not raise any exceptions when given an empty context.

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Verify that the `test_extract_nodeid_filters_setup` test verifies that setting `include_phase` to 'run' prevents filtering out node IDs during context extraction.

**Why Needed:** This test prevents a regression where the node ID filter does not extract node IDs from setup phases when `include_phase` is set to 'run'.

**Key Assertions:**

- The function `_extract_nodeid` in the `CoverageMapper` class should return None for the given input.
- The `nodeid` variable should be None after calling `_extract_nodeid` on the given input.
- The `nodeid` variable should not be a string.

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Verify that the `ExtractNodeID` method correctly extracts node ID from a run phase context.

**Why Needed:** This test prevents a potential bug where the node ID is not extracted correctly when running in the 'run' phase.

**Key Assertions:**

- The `_extract_nodeid` method of the `CoverageMapper` class should extract the node ID from the provided string.
- The extracted node ID should match the expected value (`test.py::test_foo`).
- The test should fail if the node ID extraction fails or is incorrect.

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_extract_contexts_full_logic` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Test that the mapper extracts all contexts for full logic coverage.

**Why Needed:** This test prevents regression in coverage analysis when dealing with complex codebases where some files may not be fully covered by tests.

**Key Assertions:**

- The function `_extract_contexts` should return a list of contexts containing 'test_one' and 'test_two' for file 'app.py'.
- Each context in the result should have exactly two lines (lines 1 and 2).
- The line count of each context should be equal to its corresponding file path. For example, 'test_app.py::test_one' has a line count of 2.
- The function should correctly handle mock data where some files are not fully covered by tests.

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/util/fs.py | 17 lines (ranges: 30, 33, 36, 39, 42, 55, 58-60, 63-64, 100, 103, 111-112, 116, 123) |
| src/pytest_llm_report/util/ranges.py | 13 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65-67) |

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

AI ASSESSMENT

**Scenario:** Test that `CoverageMapper._extract_contexts` handles data with no test contexts correctly.

**Why Needed:** Prevents regression in coverage reporting when there are no test contexts.

**Key Assertions:**

- mock_data.measured_files.return_value should be an empty list.
- mock_data.contexts_by_lineno.return_value should be an empty dictionary.
- result should be an empty dictionary.

COVERAGE

| | |
|---|---|
| src/pytest_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: 139-142) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test Extract Node ID Variants with different phases and contexts.

**Why Needed:** This test prevents regression in coverage mapping when the phase is not specified or when a context without a phase is used.

**Key Assertions:**

- The mapper correctly extracts node IDs for each phase.
- It returns None for run phase when no context is provided.
- It correctly handles contexts without phases (e.g., test.py::test_no_phase).
- The mapper does not return any results for nodes with missing phase information.
- It preserves the original node ID in cases where a context is used but the phase is not specified.
- The mapper returns None when no context is provided and the phase is run.
- It correctly handles contexts without phases (e.g., test.py::test_no_phase).
- The mapper does not return any results for nodes with missing phase information.

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test should fail when no coverage files exist in the test directory.

**Why Needed:** To prevent a `RuntimeError` caused by attempting to load coverage data from an empty directory.

**Key Assertions:**

- The function `_load_coverage_data()` returns None.
- The number of warnings is equal to 1.
- The first warning has the code 'W001'.

COVERAGE

| | |
|---|---|
| src/pytest_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: 139-142) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test ensures that the CoverageMapper can handle errors when loading coverage data from a corrupted file.

**Why Needed:** This test prevents a regression where the CoverageMapper fails to detect and report errors in coverage data due to a corrupt or incomplete file.

**Key Assertions:**

- The function `mapper._load_coverage_data()` should return `None` when an error occurs while reading the coverage data.
- Any warnings generated by the `CoverageMapper` should contain the string 'Failed to read coverage data' in their message.
- The `warnings` attribute of the `CoverageMapper` should be populated with these warnings.
- The function `os.chdir()` is called twice, once inside a `try/finally` block and once outside it. The second call should not affect the first one.
- The temporary directory created by `tempfile.TemporaryDirectory()` is deleted after the test finishes.
- The `CoverageData` class is patched with a mock implementation that raises an exception when read from.
- The `return_value` attribute of the mocked `CoverageData` instance is set to return a mock object instead of raising an exception.

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_coverage_map_maximal.py::TestCoverageMapperMaximal::test_load_coverage_data_with_parallel_files` 3ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test should handle parallel coverage files from xdist and verify that the CoverageMapper correctly updates its internal state to reflect changes in parallel coverage data.

**Why Needed:** This test prevents regression of a bug where the CoverageMapper does not update its internal state when handling parallel coverage files from xdist, potentially leading to incorrect coverage data being reported.

**Key Assertions:**

- The mock instances returned by `CoverageData` are different for each call to `_load_coverage_data()`
- The `update` method of the mock `CoverageData` instance is called at least twice during the test execution
- The `update` method of the mock `CoverageData` instance is not called when no parallel coverage files are present in the temporary directory

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test should handle case where _load_coverage_data returns None and return an empty dictionary.

**Why Needed:** This test prevents a potential bug in the CoverageMapper class, which may lead to incorrect results or errors if it tries to access data that is not loaded.

**Key Assertions:**

- The function mapper._load_coverage_data() should be called with no arguments and return None.
- The function mapper.map_coverage() should be called without any arguments and return an empty dictionary.

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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

AI ASSESSMENT

**Scenario:** Test that the CoverageMapper handles errors during analysis and skips corresponding files.

**Why Needed:** To prevent regression in coverage analysis when an error occurs, this test verifies that the CoverageMapper correctly skips files with errors.

**Key Assertions:**

- MockAnalysis2 should be called with an exception as its side effect.
- mock_cov.analysis2 should raise an Exception.
- mock_data.measured_files.return_value should not contain 'app.py'.
- entries should have a length of 0 after calling map_source_coverage.
- assert len(entries) == 0 should pass without raising an AssertionError.

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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 that the test_map_source_coverage_comprehensive function exercises all paths in map_source_coverage.

**Why Needed:** This test prevents regression by ensuring that the coverage of all source files is comprehensive.

**Key Assertions:**

- The function should return a list containing exactly one entry with file path 'app.py',
- the number of statements in the entry should be 3,
- the coverage percentage for this entry should be 66.67% (i.e., 2 out of 3 statements are covered),

COVERAGE

| | |
|---|---|
| src/pytest_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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the `make_warning` factory function to ensure it creates a Warning with the correct code and message.

**Why Needed:** The test prevents a potential bug where the `make_warning` function returns an incorrect Warning object without providing any meaningful information.

**Key Assertions:**

- w.code == WarningCode.W001_NO_COVERAGE
- assert "No .coverage file found" in w.message
- assert w.detail == 'test-detail'

COVERAGE

| | |
|---|---|
| `src/pytest_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: 139-142) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_errors.py::test_warning_code_values` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test that warning codes have correct values.

**Why Needed:** Prevents a potential bug where the warning code values are incorrect, potentially leading to unexpected behavior or errors in the application.

**Key Assertions:**

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

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_errors.py::test_warning_to_dict 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the `to_dict()` method of `Warning` class to ensure it correctly converts warnings into dictionaries.

**Why Needed:** This test prevents a potential bug where the `Warning.to_dict()` method does not correctly convert warnings with detailed messages.

**Key Assertions:**

- The warning code and message are extracted from the dictionary correctly.
- The detail field is included in the dictionary if present.
- The expected output matches the actual output for both test cases.
- The `to_dict()` method handles warnings without detailed messages by returning a default value.
- The `to_dict()` method handles warnings with detailed messages by including the exact details in the dictionary.

COVERAGE

| | |
|---|---|
| src/pytest_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 | 6 lines (ranges: 70-72, 74-76) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_known_code` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test verifies that a warning with the correct code and message is created when known code is used.

**Why Needed:** To prevent a regression where warnings are not correctly triggered for known code.

**Key Assertions:**

- The function `make_warning` returns an instance of `WarningCode.W101_LLM_ENABLED` with the correct value.
- The message returned by `make_warning` is set to `WARNING_MESSAGES[WarningCode.W101_LLM_ENABLED]`.
- The detail attribute of the warning is set to `None`, indicating that no additional information is available.

COVERAGE

| | |
|---|---|
| src/pytest_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: 139-142) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_unknown_code` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test Make Warning: Unknown Code

**Why Needed:** Prevents a potential bug where the 'make_warning' function throws an exception when given unknown code.

**Key Assertions:**

- The function `make_warning(missing_code)` should return a warning message.
- The function `make_warning(missing_code)` should set the warning message to 'Unknown warning.'
- The function `make_warning(missing_code)` should restore the original warning message after use.
- The exception thrown by `make_warning(missing_code)` is not related to unknown code.

COVERAGE

| | |
|---|---|
| `src/pytest_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: 139-142) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_errors_maximal.py::TestMakeWarning::test_make_warning_with_detail` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'test_make_warning_with_detail' verifies that a warning is created with the correct code and detail.

**Why Needed:** This test prevents a potential regression where warnings are not correctly propagated when using the `detail` parameter.

**Key Assertions:**

- The function `make_warning` creates a warning object with the specified `WarningCode.W301_INVALID_CONFIG` code and 'Bad value' as its detail.
- The assertion `assert w.code == WarningCode.W301_INVALID_CONFIG` checks that the created warning has the correct code.
- The assertion `assert w.detail == 'Bad value'` checks that the created warning has the correct detail.

COVERAGE

| | |
|---|---|
| src/pytest_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: 139-142) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_errors_maximal.py::TestWarningCodes::test_codes_are_strings` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Ensures that all enum values are strings and start with 'W' to prevent WarningsCodes from being set incorrectly.

**Why Needed:** This test prevents the setting of WWarningsCodes when an invalid enum value is encountered, which could lead to unexpected warnings or errors in the application.

**Key Assertions:**

- assert isinstance(code.value, str) checks if the value of each code is indeed a string.
- assert code.value.startswith('W') checks if the value starts with 'W' as expected for WarningCode enum values.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_errors_maximal.py::TestWarningDataClass::test_warning_to_dict_no_detail` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the warning to dictionary conversion without detail.

**Why Needed:** Prevents a potential bug where warnings are not properly serialized to dictionaries.

**Key Assertions:**

- The warning object is converted into a dictionary with 'code' and 'message' keys.
- The 'code' key contains the warning code.
- The 'message' key contains the warning message.
- The dictionary has the correct structure for warnings.
- The warning data does not contain any additional detail.
- The warning object is properly converted into a dictionary without any extra information.

COVERAGE

| | |
|---|---|
| src/pytest_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 | 5 lines (ranges: 70-72, 74, 76) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_errors_maximal.py::TestWarningDataClass::test_warning_to_dict_with_detail` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the warning_to_dict method with detailed information.

**Why Needed:** This test prevents a potential bug where warnings are not properly serialized to dictionaries with detail.

**Key Assertions:**

- The 'to_dict' method of Warning class should return a dictionary with the correct keys and values.
- The 'code' key in the returned dictionary should match the warning code.
- The 'message' key in the returned dictionary should match the warning message.
- The 'detail' key in the returned dictionary should match the warning detail.
- The 'warning_code' attribute of the Warning object should be accessible through the 'to_dict' method.
- The 'setup_check' attribute of the Warning object should also be accessible through the 'to_dict' method.

COVERAGE

| | |
|---|---|
| `src/pytest_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` | `6 lines (ranges: 70-72, 74-76)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_fs.py::TestIsPythonFile::test_non_python_file` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verifies that the `is_python_file` function returns False for non-.py files.

**Why Needed:** Prevents a potential bug where the function incorrectly identifies Python files as non-Python files, potentially leading to incorrect file type detection in other parts of the codebase.

**Key Assertions:**

- The function should return `False` for files without a `.py` extension (e.g., `foo/bar.txt`).
- The function should return `False` for files with a `.pyc` extension (e.g., `foo/bar.pyc`).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** Verifies that the function `is_python_file` returns True for a .py file.

**Why Needed:** Prevents a potential bug where the function does not correctly identify .py files.

**Key Assertions:**

- The function should return `True` for a file named `foo/bar.py`.
- The function should raise an error or return False for a non-python file.
- The function should handle cases where the file name is missing or empty.
- The function should not incorrectly identify files with different extensions (e.g. `.txt`, `.java`).
- The function should correctly handle files with relative paths (e.g. `./foo/bar.py`).
- The function should raise an error for a non-existent file.
- The function should return False for a file named `non_python_file.py` or any other invalid filename.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** Test makes absolute path relative by creating a subdirectory and then making the file within it relative to the test directory.

**Why Needed:** This test prevents regression where the `make_relative` function does not correctly make paths relative when working with subdirectories.

**Key Assertions:**

- The `file_path.parent` should be created if it does not exist, and then a file named `file.py` should be created within that parent directory.
- The `file_path.touch()` call should create the file `file.py` in the specified path.
- After creating the subdirectory and file, the `make_relative(file_path, tmp_path)` function should return the expected relative path.
- If the test directory does not exist, `make_relative(file_path, tmp_path)` should raise an exception or handle it correctly.
- The `file_path.parent.mkdir(parents=True, exist_ok=True)` call should create the parent directories if they do not already exist.
- The `file_path.touch()` call should update the file's last modification time to reflect its new location within the subdirectory.
- After creating and modifying the file, the relative path returned by `make_relative(file_path, tmp_path)` should be correct.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** Verifies that the `make_relative` function returns a normalized path when there is no base.

**Why Needed:** Prevents a potential bug where an absolute path would be returned instead of a relative one.

**Key Assertions:**

- The result of calling `make_relative('foo/bar')` should be 'foo/bar'.
- The function should not return an absolute path ('/foo/bar') when there is no base.
- The function should handle cases where the input path has a trailing slash correctly.
- The function should not modify the original file system in any way.
- The function should throw an error if the input path is empty or null.
- The function should not return a relative path with a leading dot (..) when there is no base.
- The function should handle cases where the input path has multiple parents correctly.
- The function should not return an absolute path that points to a non-existent file.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** The test verifies that the `normalize_path` function correctly handles already-normalized paths.

**Why Needed:** This test prevents a potential bug where an already normalized path would be incorrectly normalized back to its original form.

**Key Assertions:**

- assert normalize_path('foo/bar') == 'foo/bar'
- assert normalize_path('/foo/bar') == '/foo/bar'
- assert normalize_path('foo//bar') == 'foo//bar'
- assert normalize_path('foo/./bar') == 'foo/./bar'
- assert normalize_path('foo/../bar') == 'foo/bar'
- assert normalize_path('foo/../../bar') == 'foo/bar'
- assert normalize_path('/home/foo/bar') == '/home/foo/bar'

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `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:** The 'test_forward_slashes' test verifies that the `normalize_path` function correctly converts backslashes to forward slashes when a single slash is present.

**Why Needed:** This test prevents a potential bug where the function does not handle cases with multiple consecutive backslashes correctly, potentially leading to incorrect path normalization.

**Key Assertions:**

- The normalized path should contain exactly one forward slash.
- The path should be in the format 'path/to/file'.
- The directory part of the path should have a forward slash at the beginning if it exists.
- The file part of the path should not start with a forward slash.
- The path should end with a forward slash if necessary.
- The function should correctly handle cases where there are no backslashes in the input string.
- The function should throw an error when given invalid input, such as a string containing only forward slashes.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/util/fs.py` | 5 lines (ranges: 30, 33, 36, 39, 42) |

PASSED `tests/test_fs.py::TestNormalizePath::test_strips_trailing_slash` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Tests the `normalize_path` function to remove trailing slashes from paths.

**Why Needed:** Prevents a potential bug where a path with a trailing slash is not correctly normalized.

**Key Assertions:**

- The input path should be stripped of any trailing slashes.
- The resulting path should only contain one slash.
- Any leading or trailing whitespace in the original path should be preserved in the output.
- The function should handle paths with multiple consecutive slashes correctly.
- The function should not modify the original path if it already had no trailing slashes.
- The function should raise an error for invalid input (e.g. a non-string path).
- Any exceptions raised during normalization should be caught and propagated to the caller.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/util/fs.py` | 5 lines (ranges: 30, 33, 36, 39, 42) |

PASSED `tests/test_fs.py::TestShouldSkipPath::test_custom_exclude_patterns` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test verifies whether a path matches custom exclusion patterns.

**Why Needed:** Prevents regression by ensuring that paths matching custom patterns are correctly excluded.

**Key Assertions:**

- The function `should_skip_path` should return True for the given path and exclude patterns.
- The function `should_skip_path` should return False for the given path and exclude patterns.
- The function `should_skip_path` should exclude the specified paths from the test directory.
- The function `should_skip_path` should not include the excluded paths in the list of skipped files.
- The function `should_skip_path` should handle cases where the exclude pattern is empty or contains only '*' characters.
- The function `should_skip_path` should correctly handle cases where the path does not match any exclusion patterns.
- The function `should_skip_path` should raise an error if the excluded patterns are invalid or malformed.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** Verifies that the `should_skip_path` function does not return True for normal paths.

**Why Needed:** Prevents a regression where the test incorrectly skips normal paths.

**Key Assertions:**

- The function should return `False` for paths like 'src/module.py'.
- The function should not be able to determine whether a path is normal or not without additional context.
- If the path contains a module name, it should still return `False`.
- If the path does not contain a module name, it should also return `False`.
- The test should cover all possible paths in the project.
- The function should be able to handle large projects with many files.
- The function should not skip normal paths when they are part of a package.
- If the path is relative, it should still return `False`.
- If the path is absolute, it should return `True`.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** Verifies that the `should_skip_path` function correctly identifies `.git` directories.

**Why Needed:** Prevents a potential issue where the test incorrectly skips non-`.git` directories by relying on the presence of `objects/` in their paths.

**Key Assertions:**

- The path to the `.git` directory is `./.git/objects/foo`.
- The function should return `True` for this path because it contains a `.git` directory.
- The function should not return `True` for other non-`.git` directories like `./non_git_directory`.
- The function should raise an error or handle the case where the path is not a `.git` directory correctly.
- The test should be able to reproduce the issue by running the test with different paths.
- The test should fail when the path does not contain a `.git` directory but still return `True` for it.
- The function's behavior should change if the current working directory is outside of the `.git` directory tree.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** The test verifies that the `should_skip_path` function correctly identifies and skips `__pycache__` directories.

**Why Needed:** This test prevents a potential bug where the function incorrectly or unnecessarily includes `__pycache__` paths in the skip list.

**Key Assertions:**

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

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** The test verifies that the `should_skip_path` function correctly identifies venv directories as skip paths.

**Why Needed:** This test prevents a potential issue where the function incorrectly includes non-venv directories in its list of skip paths.

**Key Assertions:**

- assert should_skip_path('venv/lib/python/site.py') is True
- assert should_skip_path('.venv/lib/python/site.py') is True

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `src/pytest_llm_report/util/fs.py` | `10 lines (ranges: 30, 33, 36, 39, 42, 100, 103, 111-113)` |

PASSED `tests/test_gemini_advanced.py::TestGeminiRateLimiter::test_pruning` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that pruning of request times clears the _request_times and _token_usage lists after a past request has been processed.

**Why Needed:** This test prevents a potential bug where requests from the past are not properly cleared after being pruned, leading to incorrect usage statistics.

**Key Assertions:**

- The length of _request_times should be 0 after pruning.
- The length of _token_usage should be 0 after pruning.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_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: 380-381, 384, 388-390) |

PASSED `tests/test_gemini_advanced.py::TestGeminiRateLimiter::test_rpm_limit` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the rate limiter prevents requests from exceeding a certain limit when it is available.

**Why Needed:** This test prevents a potential issue where a request exceeds the allowed rate limit, causing the rate limiter to become unavailable.

**Key Assertions:**

- The `next_available_in` method should return a non-negative value greater than or equal to 0.
- The `wait` assertion should be within the range of [0, 60.0].
- The `limiter.record_request()` call should not block the execution of the test function.

COVERAGE

| | |
|---|---|
| src/pytest_llm_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 | 26 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: 380-381, 384, 388-390) |

PASSED `tests/test_gemini_advanced.py::TestGeminiRateLimiter::test_tpm_limit` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the rate limiter prevents a regression when the token limit is exceeded.

**Why Needed:** This test verifies that the rate limiter correctly prevents excessive token usage and updates the state accordingly.

**Key Assertions:**

- The next available time point should be greater than 0.
- The total number of tokens used so far (90) should not exceed the limit (100).
- The _token_usage list should contain exactly two elements after updating with new tokens (10).

COVERAGE

| | |
|---|---|
| `src/pytest_llm_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` | 33 lines (ranges: 39-42, 45-46, 48, 52-54, 66, 68-70, 81-82, 84, 87-88, 92-94, 100-101, 103, 105, 107-108, 110-114) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_gemini_advanced.py::TestGeminiRateLimiter::test_wait_for_slot 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the `wait_for_slot` method sleeps for a specified amount of time.

**Why Needed:** This test prevents a potential issue where the rate limiter does not sleep when it should, potentially causing unexpected behavior or performance issues.

**Key Assertions:**

- The `wait_for_slot` method is called with the correct argument (1)
- The `time.sleep` function is called with the correct argument (1)
- The `assert` statement checks for the correct condition (the `mock_sleep` object was called)

COVERAGE

| | |
|---|---|
| src/pytest_llm_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 | 31 lines (ranges: 39-42, 45-46, 48, 52-54, 58-59, 61-63, 73, 76-78, 81-82, 84, 87-88, 92-93, 95, 97, 100-102) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_gemini_coverage_v2.py::test_gemini_limiter_record_zero_tokens` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the rate limiter does not attempt to record tokens when there are zero tokens available.

**Why Needed:** This test prevents a potential bug where the rate limiter attempts to record tokens early, potentially leading to incorrect usage statistics.

**Key Assertions:**

- The length of `_token_usage` is equal to 0 after calling `record_tokens(0)`.
- The value of `_token_usage` is an empty list.
- _token_usage contains no elements.
- The number of tokens in `_token_usage` is 0.
- The rate limiter's usage statistics are correct.

COVERAGE

| | |
|---|---|
| src/pytest_llm_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: 380-381, 384, 388-390) |

PASSED `tests/test_gemini_coverage_v2.py::test_gemini_limiter_requests_per_day_exhaustion` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the test raises an error when exceeding daily rate limit.

**Why Needed:** Prevents a potential rate limiting exceeded error in the Gemini RateLimiter.

**Key Assertions:**

- The function `wait_for_slot` should raise `_GeminiRateLimitExceeded` with a message 'requests_per_day'.
- The function `wait_for_slot` should raise `_GeminiRateLimitExceeded` with a message 'requests_per_day' after calling `record_request`.
- The error message should contain the string 'requests_per_day'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_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: 380-381, 384, 388-390) |

PASSED `tests/test_gemini_coverage_v2.py::test_gemini_limiter_tpm_fallback_wait` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the TPM fallback wait time is correctly calculated when filling up TPM with tokens.

**Why Needed:** The test prevents a potential regression where the TPM fallback wait time is too short, potentially leading to unnecessary rate limiting.

**Key Assertions:**

- assert wait > 0
- assert (tokens_used + request_tokens) > limit AND token_usage is not empty
- assert tokens_used + request_tokens >= limit
- assert request_tokens == 10
- assert tokens_used == 20
- assert request_tokens == 10
- assert token_usage != []

COVERAGE

| | |
|---|---|
| src/pytest_llm_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: 39-42, 66, 68-70, 81-82, 84, 87-88, 100-101, 103, 105, 107-108, 110-114) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_gemini_coverage_v2.py::test_gemini_provider_rpm_cooldow
n 581ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Test that RPM rate limit cooldown handling is correctly implemented.

**Why Needed:** This test prevents a potential bug where the RPM rate limit cooldown is not properly set, leading to unexpected behavior when hitting the limit on subsequent calls.

**Key Assertions:**

- The 'models/gemini-pro' model should be present in the provider's cooldowns.
- The cooldown for 'models/gemini-pro' should be greater than 1000.0 seconds (1 minute).
- The provider's cooldowns should contain 'models/gemini-pro'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| src/pytest_llm_report/llm/gemini.py | 117 lines (ranges: 32-34, 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, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-187, 200-202, 206-208, 210, 213-215, 217-223, 225-227, 233-234, 238-240, 242-243, 274-277, 280, 282-290, 292-295, 297-298, 300-301, 346, 348-350, 352-353, 381-382, 385-386) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_gemini_provider.py::TestGeminiProvider::test_annotate_rate_limit_retry 4ms 🛡️ 4

AI ASSESSMENT

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

**Why Needed:** This test prevents regression where the provider fails to annotate a rate limit retry scenario due to an incorrect or missing Retry-After header in the response.

**Key Assertions:**

- The annotation should have the correct scenario 'Recovered Scenario' when the first call to _annotate_internal fails with a 429 status code.
- The second call to _annotate_internal should succeed with the correct scenario 'Recovered Scenario' and no error.
- The number of calls to mock_post should be exactly 2, one for each call to _annotate_internal.
- The annotation should not have an error when the first call to _annotate_internal fails with a 429 status code.
- The annotation should not have an error when the second call to _annotate_internal succeeds with the correct scenario 'Recovered Scenario'.
- The mock_post.call_count attribute should be set to 2 after the test.
- The mock_parse.return_value.json.return_value should contain a key named 'scenario' with value 'Recovered Scenario'.
- The mock_parse.return_value.error attribute should be None.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53) |
| src/pytest_llm_report/llm/gemini.py | 181 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, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-187, 200-202, 206-208, 210, 213-215, 217-222, 225-227, 233-234, 238-240, 242-243, 274-277, 280-283, 286-290, 292-295, 297-298, 300-301, 315, 317-320, 322-325, 327-328, 330-333, 335-341, 343, 346, 348-350, 352-355, 360-363, 374-377, 381-382, 385-387, 391-392, 396-399, 401-402, 405, 408-410, 412-414, 417, 419, 421-424, 428, 430- |

|  |  |
|---|---|
|  | 434, 437-440, 442-443, 445-447) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_gemini_provider.py::TestGeminiProvider::test_annotate_success 5ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test that _annotate_success correctly annotates success scenarios with the expected LlmAnnotation and no errors.

**Why Needed:** This test prevents a potential regression where the GeminiProvider incorrectly annotates failure scenarios as successes.

**Key Assertions:**

- The annotation returned by _annotate_internal has the correct scenario 'Success Scenario'.
- The annotation does not have an error attribute.
- The annotation's scenario matches the expected value 'Success Scenario'.
- _parse_response returns a Mock object with the correct scenario and no error.
- The annotation's error attribute is None.
- The _build_prompt function is mocked to avoid complex dependencies, which is necessary for this test.
- The _call_gemini method correctly calls _parse_response where it expects text and tokens.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.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: 52-53) |
| `src/pytest_llm_report/llm/gemini.py` | 173 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, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-187, 200-202, 206-208, 210, 213-214, 217-223, 225-227, 274-277, 280-283, 286-290, 292-295, 297-298, 300-301, 315, 317-320, 322-325, 327-328, 330, 335-343, 346, 348-350, 352-355, 360-363, 374-377, 381-382, 385-387, 391-392, 396-399, 401-402, 405, 408-410, 412-414, 417, 419, 421-424, 428, 430-434, 437-440, 442-443, 445-447) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_gemini_provider.py::TestGeminiProvider::test_availability` 3ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test that the availability of a Gemini provider is correctly checked and updated accordingly.

**Why Needed:** This test prevents a potential bug where the availability status of a Gemini provider is not properly updated when the environment variables are cleared or modified.

**Key Assertions:**

- The function _check_availability() in the GeminiProvider class returns False if the provider is set to 'gemini'.
- The function _check_availability() in the GeminiProvider class returns True if the provider is set to 'gemini' and the GEMINI_API_TOKEN environment variable is provided.
- The function _check_availability() in the GeminiProvider class checks for a specific error message when the availability status is False, indicating that the provider has been successfully checked and updated.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53) |
| src/pytest_llm_report/llm/gemini.py | 10 lines (ranges: 134, 136-139, 141-142, 266-267, 269) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_gemini_provider.py::TestGeminiRateLimiter::test_rpd_limit` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the rate limiter prevents exceeding the daily limit of 1 request per day.

**Why Needed:** This test prevents a potential bug where the rate limiter allows more than one request to be processed in a single day.

**Key Assertions:**

- The next_available_in method returns None when there are no available slots for requests.
- The record_request method increments the request count and updates the limiter's state accordingly.
- The limiter's next_available_in method checks if there are any available slots before returning None.
- The limiter's next_available_in method does not return a value when all slots are filled (i.e., it returns None).
- The record_request method increments the request count and updates the limiter's state correctly.
- The limiter's next_available_in method should be able to handle cases where there are multiple requests in the queue.
- The limiter's next_available_in method should return None when all slots are filled, indicating that no more requests can be processed.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_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: 380-381, 384, 388-390) |

PASSED `tests/test_gemini_provider.py::TestGeminiRateLimiter::test_rpm_limit` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that the rate limiter does not block requests until it has available slots.

**Why Needed:** This test prevents a potential issue where multiple requests are blocked due to insufficient available slots in the rate limiter.

**Key Assertions:**

- The next_available_in method returns 0.0 after two requests have been recorded.
- The next_available_in method returns 0.0 after three more requests have been recorded.
- wait is greater than 0 and less than or equal to 60.0 when the next_available_in method is called.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_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: 380-381, 384, 388-390)` |

PASSED `tests/test_hashing.py::TestComputeConfigHash::test_different_config` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test that different configurations of the same provider produce different hashes.

**Why Needed:** This test prevents a bug where two instances with the same configuration but different providers produce the same hash, potentially leading to incorrect configuration identification.

**Key Assertions:**

- The function `compute_config_hash` should return a different hash for two different configurations of the same provider.
- The function `compute_config_hash` should not return the same hash when comparing two instances with the same configuration but different providers.
- The function `compute_config_hash` should raise an error if it is unable to compute the hash for one or both of the given configurations.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `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:** Verifies the length of the computed hash is exactly 16 characters.

**Why Needed:** This test prevents a potential issue where the hash length could be longer than expected, potentially leading to incorrect comparisons or data corruption.

**Key Assertions:**

- The length of the computed hash should not exceed 15 characters.
- The first character of the hash should be '0'.
- The second character of the hash should be '1'.
- The third character of the hash should be '2'.
- The fourth character of the hash should be '3'.
- The fifth character of the hash should be '4'.
- The sixth character of the hash should be '5'.
- The seventh character of the hash should be '6'.
- The eighth character of the hash should be '7'.
- The ninth character of the hash should be '8'.
- The tenth character of the hash should be '9'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/util/hashing.py | 8 lines (ranges: 96-101, 103-104) |

PASSED `tests/test_hashing.py::TestComputeFileSha256::test_consistent_with_bytes` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the computed SHA-256 hash of a file matches its content hash when the same file is used with different input.

**Why Needed:** This test prevents regression in cases where the input file's contents are modified or deleted, causing the file hash to change unexpectedly.

**Key Assertions:**

- The computed SHA-256 hash of the file `test.txt` should be equal to its content hash `test content`.
- The computed SHA-256 hash of the file `test.txt` should not be affected by any modifications or deletions of the file's contents.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/util/hashing.py` | 6 lines (ranges: 32, 44-48) |

PASSED `tests/test_hashing.py::TestComputeFileSha256::test_hashes_file` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verifies the correctness of hashing a file using SHA-256.

**Why Needed:** Prevents a potential bug where the hash length is not correctly calculated.

**Key Assertions:**

- The length of the computed hash should be exactly 64 bytes.
- The hash value should match the expected output from a SHA-256 algorithm.
- Any errors or exceptions during the hashing process should be properly handled and reported.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** Test that different keys produce different signatures.

**Why Needed:** Prevents a potential bug where the same key could produce the same signature, potentially leading to unexpected behavior or security vulnerabilities.

**Key Assertions:**

- The HMAC of the content with the first key is not equal to the HMAC of the content with the second key.
- The HMAC of the content with the first key is different from the expected value (which would be the HMAC of the content with the second key).
- The signature produced by the first key does not match the expected signature for the same input and key.
- The HMAC of the content with the first key is not a valid signature for the given content and key.
- The HMAC of the content with the second key is different from the expected value (which would be the HMAC of the content with the first key).
- The signature produced by the second key does not match the expected signature for the same input and key.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** Verifies the computation of HMAC using a secret key.

**Why Needed:** Prevents potential security vulnerabilities by ensuring the integrity and authenticity of the computed signature.

**Key Assertions:**

- The length of the computed HMAC should be exactly 64 bytes.
- The computed HMAC should not contain any padding characters (if present).
- The computed HMAC should only include the secret key as its payload, without any additional data.
- Any non-secret-key input should result in an empty or invalid signature.
- The computed HMAC should match the expected output for a given input and secret key.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/util/hashing.py` | 1 lines (ranges: 61) |

PASSED `tests/test_hashing.py::TestComputeSha256::test_consistent` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'same_content_should_produce_same_hash' verifies that the SHA-256 hash of a given content is consistent.

**Why Needed:** This test prevents a potential bug where different inputs to the `compute_sha256` function produce different hashes, potentially leading to inconsistent results.

**Key Assertions:**

- The two input strings should have the same length and only contain ASCII characters.
- The two input strings should be identical in content (i.e., they should not differ by any non-ASCII character).
- If a non-ASCII character is present, it should appear exactly once in each string.
- If a non-ASCII character appears more than once in one string but not the other, it should appear only once in the string with fewer occurrences.
- The two input strings should be identical when compared using `hash()`.
- The two input strings should have the same hash value when computed using `compute_sha256()`.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `src/pytest_llm_report/util/hashing.py` | `1 lines (ranges: 32)` |

PASSED `tests/test_hashing.py::TestComputeSha256::test_length` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify the length of the computed SHA-256 hash.

**Why Needed:** Prevents a potential issue where the hash length is not as expected due to incorrect input data.

**Key Assertions:**

- The length of the output should be exactly 64 characters.
- The length of the output should not exceed 63 characters.
- The length of the output should not be less than 0 characters.
- The length of the output should be an integer value.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `src/pytest_llm_report/util/hashing.py` | `1 lines (ranges: 32)` |

PASSED `tests/test_hashing.py::TestGetDependencySnapshot::test_includes_pytest` 82ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verifies that the `get_dependency_snapshot` function includes the 'pytest' package.

**Why Needed:** This test prevents a regression where the 'pytest' package is not included in the dependency snapshot.

**Key Assertions:**

- The 'pytest' package should be present in the snapshot.
- The 'pytest' package should be listed as an item in the snapshot.
- The presence of 'pytest' in the snapshot indicates that it is a required dependency.
- The absence of 'pytest' in the snapshot indicates that it is not a required dependency.
- The inclusion of 'pytest' in the snapshot ensures that all dependencies are accounted for.
- The exclusion of 'pytest' from the snapshot may indicate a missing or incorrect dependency.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/util/hashing.py | 8 lines (ranges: 113-114, 116-121) |

PASSED `tests/test_hashing.py::TestGetDependencySnapshot::test_returns_dict` 83ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that the `get_dependency_snapshot` function returns a dictionary.

**Why Needed:** This test prevents a potential bug where the function might not return a dictionary or may return incorrect data.

**Key Assertions:**

- snapshot is an instance of dict
- snapshot has a non-empty keys attribute
- snapshot does not have any duplicate keys
- snapshot has all required packages
- snapshot has no missing packages
- snapshot has correct package ordering

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** Test that the `load_hmac_key` function correctly loads a key from a file.

**Why Needed:** This test prevents a potential bug where the HMAC key is not loaded correctly if the file does not exist or cannot be read.

**Key Assertions:**

- The output of the `load_hmac_key(config)` function should be equal to the expected value `b'my-secret-key'`.
- The `config.hmac_key_file` attribute should point to the file path `str(key_file)`.
- The `load_hmac_key(config)` function should successfully load the key from the file and return it as a bytes object `b'my-secret-key'`.
- The `key` variable should be assigned the correct value `my-secret-key` after calling `load_hmac_key(config)`.
- The `assert` statement should raise an AssertionError with a message indicating that the key was not loaded correctly.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/util/hashing.py | 5 lines (ranges: 73, 76-77, 80-81) |

PASSED tests/test_hashing.py::TestLoadHmacKey::test_missing_key_file 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test that `load_hmac_key` returns `None` when the HMAC key file does not exist.

**Why Needed:** Prevents a potential bug where the function `load_hmac_key` fails to load an HMAC key due to a missing or invalid key file.

**Key Assertions:**

- The `hmac_key_file` parameter is set to a string representing the path to a non-existent key file.
- The `load_hmac_key` function attempts to load the HMAC key from the specified file.
- An assertion error occurs when the `load_hmac_key` function fails to load the HMAC key due to a missing or invalid key file.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** Verifies that the `load_hmac_key` function returns `None` when no key file is specified.

**Why Needed:** Prevents a potential bug where the function does not handle cases without a configured key file.

**Key Assertions:**

- The `load_hmac_key` function should return `None` if no key file is provided.
- The `load_hmac_key` function should raise an error or return an appropriate value when no key file is specified.
- The test should verify that the expected behavior (i.e., returning `None`) occurs in all cases.
- The test should also verify that the function raises an error or returns an appropriate value when no key file is provided.
- The test should include a clear and concise description of the scenario being tested.
- The test should use a valid configuration object to simulate a case without a key file.
- The test should avoid using assertions that rely on external state (e.g., `key` variable).

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/util/hashing.py` | 2 lines (ranges: 73-74) |

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_aggregation_defaults` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the default aggregation configuration.

**Why Needed:** Prevents a potential bug where aggregation defaults are not set correctly, leading to unexpected behavior in the integration gate.

**Key Assertions:**

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

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147, 233) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_capture_failed_output_default_false` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the `capture_failed_output` default value is set to `False` for the integration gate.

**Why Needed:** This test prevents a potential bug where the `capture_failed_output` option is not correctly configured, leading to unexpected behavior or errors.

**Key Assertions:**

- config.capture_failed_output should be `None` (or an empty boolean value) when `integration_gate` is disabled.
- config.capture_failed_output should be `False` when `integration_gate` is set to `True`.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147, 233) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_integration_gate.py::TestConfigDefaults::test_context_mode_default_minimal 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Tests the default context mode for integration gate.

**Why Needed:** Prevents a potential bug where the context mode is not set to 'minimal' by default.

**Key Assertions:**

- The function `get_default_config()` returns an instance of `Config` with a `llm_context_mode` attribute set to 'minimal'.
- The value of `config.llm_context_mode` is equal to 'minimal'.
- The context mode is not set to 'minimal' by default.
- The function `get_default_config()` does not return an instance of `Config` with a `llm_context_mode` attribute set to 'minimal'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147, 233) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_llm_not_enabled_by_default` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the LLM is not enabled by default in the configuration.

**Why Needed:** The test prevents a regression where the LLM might be enabled by default, potentially causing unexpected behavior or errors.

**Key Assertions:**

- config.is_llm_enabled() == False
- config.get_llm_enabled_value() == False
- get_default_config().llm_enabled() == False
- get_default_config().get_llm_enabled_value() == False

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147, 224, 233) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_integration_gate.py::TestConfigDefaults::test_omit_tests_default_true 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the `TestConfigDefaults` class returns `True` when omitting tests from coverage by default.

**Why Needed:** This test prevents a regression where the `TestConfigDefaults` class does not correctly handle the case where `omit_tests_from_coverage` is set to `False`.

**Key Assertions:**

- config.omit_tests_from_coverage should be set to `True` when `omit_tests_from_coverage` is `False`
- config.omit_tests_from_coverage should be a boolean value
- The `TestConfigDefaults` class should correctly handle the case where `omit_tests_from_coverage` is `False`
- When `omit_tests_from_coverage` is `True`, `config.omit_tests_from_coverage` should be `True`
- When `omit_tests_from_coverage` is `False`, `config.omit_tests_from_coverage` should be a boolean value

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147, 233) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_provider_default_none` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Tests the default provider setting when it is set to 'none'.

**Why Needed:** This test prevents a potential bug where the provider is not set to 'none' by default.

**Key Assertions:**

- The `provider` attribute of the configuration object should be equal to 'none'.
- The `provider` attribute of the configuration object should not be equal to any other value (e.g., 'google', 'facebook').

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147, 233) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_integration_gate.py::TestConfigDefaults::test_secret_exclude_globs` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that secret files are excluded by default from the LLM context.

**Why Needed:** This test prevents a potential bug where non-secret files are inadvertently included in the LLM context.

**Key Assertions:**

- The 'secret' keyword is present in any globs listed as excluding them.
- The '.env' file is excluded from being processed by the LLM.
- Any other secret files or directories not explicitly excluded are also excluded.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147, 233) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_integration_gate.py::TestFullPipeline::test_deterministic_output` 6ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test that the deterministic output of the integration gate is reported correctly.

**Why Needed:** This test prevents a regression where the deterministic output may not be reported correctly due to a change in the sorting logic of the report.

**Key Assertions:**

- The nodeids are sorted correctly before and after writing the report.
- The nodeid 'z_test.py::test_z' is present in the list of nodeids.
- The nodeid 'a_test.py::test_a' is present in the list of nodeids.
- The nodeid 'm_test.py::test_m' is present in the list of nodeids.
- The nodeids are sorted alphabetically (case-insensitive) before and after writing the report.
- No duplicate nodeids are present in the list of nodeids.
- The nodeid 'z_test.py::test_z' appears first in the list of nodeids.
- The nodeid 'a_test.py::test_a' appears second in the list of nodeids.
- The nodeid 'm_test.py::test_m' appears third in the list of nodeids.

COVERAGE

| | |
|---|---|
| src/pytest_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 | 78 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 364-380, 382-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

| | |
|---|---|
| src/pytest_llm_report/report_writer.py | 117 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, 312, 314-315, 317-318, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_integration_gate.py::TestFullPipeline::test_empty_test_suite` 6ms 🛡️ 5

AI ASSESSMENT

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

**Why Needed:** To prevent regression in case of an empty test suite, where the report may not contain any summary information.

**Key Assertions:**

- The total count of tests is zero.
- All metrics are set to zero.
- No summary data is present in the report.
- There are no test runs included in the report.
- The report does not contain any test execution details.
- Test suite is empty, hence no test results can be reported.

COVERAGE

| | |
|---|---|
| src/pytest_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 | 67 lines (ranges: 229-231, 233, 235, 364-380, 382-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510-512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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, 202-206, 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, 312, 314, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED tests/test_integration_gate.py::TestFullPipeline::test_html_report_generation 31ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test that the Full pipeline generates an HTML report.

**Why Needed:** This test prevents a regression where the HTML report is not generated correctly due to a mismatch between the expected and actual file paths.

**Key Assertions:**

- The 'report.html' file exists at the specified path.
- The '' tag is present in the contents of the 'report.html' file.
- The 'test_pass' string is included in the contents of the 'report.html' file.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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 | 113 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, 312, 314-315, 317-318, 330, 376, 378-379, 382, 385, 388, 391-395, 470-471, 495, 497, 499-501, 503, 506) |

PASSED tests/test_integration_gate.py::TestFullPipeline::test_json_report_generation 55ms 🛡️ 7

AI ASSESSMENT

**Scenario:** Test that the full pipeline generates a valid JSON report.

**Why Needed:** This test prevents regression in the integration gate, where it's possible for the pipeline to generate incorrect or missing reports.

**Key Assertions:**

- The report is generated with the correct schema version and summary statistics.
- The total count of tests passed, failed, skipped, and pending is accurate.
- The number of tests that passed, failed, and skipped are correctly reported in the 'summary' section.
- The test results are stored in the 'report.json' file as expected.
- The report is written to a valid JSON file with the correct path.
- The data loaded from the JSON file contains the expected schema version, total count of tests, and summary statistics.
- The number of tests that passed, failed, and skipped are correctly reported in the 'summary' section.

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 | 78 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 364-380, 382-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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 | 133 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, 312, 314-315, 317-322, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 376, 378-379, 382, 385, 388, 391-395, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_integration_gate.py::TestSchemaCompatibility::test_report_root_has_required_fields` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that the ReportRoot has required fields.

**Why Needed:** This test prevents a bug where the report root is missing required fields, potentially causing errors during validation or reporting.

**Key Assertions:**

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

COVERAGE

| | |
|---|---|
| src/pytest_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: 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_integration_gate.py::TestSchemaCompatibility::test_run_meta_has_aggregation_fields` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that `RunMeta` has an 'aggregation_fields' key.

**Why Needed:** Prevents regression where the schema does not have any aggregated fields.

**Key Assertions:**

- is_aggregated should be present in the data.
- run_count should also be present in the data.

COVERAGE

| | |
|---|---|
| `src/pytest_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: 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_integration_gate.py::TestSchemaCompatibility::test_run_meta_has_status_fields` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that RunMeta has run status fields.

**Why Needed:** Prevents regression where the 'RunMeta' object does not have a 'status' field, potentially causing unexpected behavior or errors when trying to access it.

**Key Assertions:**

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

COVERAGE

| | |
|---|---|
| src/pytest_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: 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_integration_gate.py::TestSchemaCompatibility::test_schema_version_defined` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Tests that the schema version is defined and matches a semver-like format.

**Why Needed:** Prevents regression where the schema version is not defined or does not match a valid semver-like format.

**Key Assertions:**

- The schema version should be present in the test.
- The schema version should be a string that can be compared to a valid semver-like format (e.g., '1.2.3').
- The schema version should not be empty or an empty string.
- The schema version should contain at least one dot (.) character.
- The schema version should match the expected format (e.g., '1.2.3') exactly.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_integration_gate.py::TestSchemaCompatibility::test_test_case_has_required_fields` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'test_case_has_required_fields' verifies that the TestCaseResult object has all required fields.

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

**Key Assertions:**

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

COVERAGE

| | |
|---|---|
| src/pytest_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 | 17 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm.py::TestGetProvider::test_gemini_returns_provider` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the `get_provider` function returns an instance of `GeminiProvider` when the 'provider' parameter is set to 'gemini'.

**Why Needed:** This test prevents a potential bug where the `get_provider` function may return an incorrect provider type if the 'provider' parameter is not set to 'gemini'.

**Key Assertions:**

- The method name of the returned provider instance should be 'GeminiProvider'.
- The class name of the returned provider instance should be 'GeminiProvider'.
- The attribute name of the returned provider instance should be 'provider'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 245, 247, 249, 252, 257, 262-263, 265) |
| src/pytest_llm_report/llm/gemini.py | 7 lines (ranges: 134, 136-139, 141-142) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm.py::TestGetProvider::test_litellm_returns_provider` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** The test verifies that the `get_provider` function returns a correct instance of LiteLLMProvider when provided with the 'litellm' configuration.

**Why Needed:** This test prevents regression where the `get_provider` function may return an incorrect provider type or class name for the 'litellm' configuration.

**Key Assertions:**

- The `provider` attribute of the returned instance should be set to `LiteLLMProvider`.
- The `__class__.__name__` attribute of the returned instance should match `LiteLLMProvider`.
- The `provider` attribute should have a value of `'litellm'`.
- The `get_provider` function should return an instance of `LiteLLMProvider` for the given configuration.
- The `__class__.__name__` attribute of the returned provider should be `LiteLLMProvider`.
- The `provider` attribute of the returned provider should have a value of `'litellm'`.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.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: 52-53, 245, 247, 249, 252, 257-258, 260) |
| `src/pytest_llm_report/options.py` | 2 lines (ranges: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm.py::TestGetProvider::test_none_returns_noop` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test that the `None` provider returns a `NoopProvider` instance.

**Why Needed:** This test prevents regression where the LLM model is not correctly handling cases with no provider specified.

**Key Assertions:**

- The function `get_provider(config)` should return an instance of `NoopProvider` when `provider='none'`.
- The variable `provider` should be set to `None` after calling `get_provider(config)`.
- The type of the returned `provider` should be `NoopProvider`.
- An exception of type `ValueError` should not be raised if `provider='none'`.
- A NoopProvider instance should be created when `provider='none'`.
- The `get_provider(config)` function should return a `None` value for other provider types.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 245, 247, 249-250) |
| src/pytest_llm_report/llm/noop.py | 1 lines (ranges: 32) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm.py::TestGetProvider::test_ollama_returns_provider` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** The test verifies that OllamaProvider is returned when 'provider='ollama' in the config.

**Why Needed:** This test prevents a potential bug where the correct provider type (OllamaProvider) is not returned if the configuration contains an invalid or missing 'httpx' library.

**Key Assertions:**

- The function get_provider() returns an instance of OllamaProvider.
- The class name of the returned provider is OllamaProvider.
- The provider type is correctly identified as OllamaProvider even if httpx library is not available.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.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: 52-53, 245, 247, 249, 252-253, 255)` |
| `src/pytest_llm_report/options.py` | `2 lines (ranges: 107, 147)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

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

AI ASSESSMENT

**Scenario:** Test that an unknown provider raises a ValueError when trying to get a provider.

**Why Needed:** This test prevents the UnknownProviderError from being raised, ensuring the function behaves correctly when encountering an unknown provider.

**Key Assertions:**

- The function `get_provider()` should raise a ValueError with message 'unknown' when called with an unknown provider.
- The error message of the ValueError should contain the string 'unknown'.
- The error message of the ValueError should be in lowercase.
- The function `get_provider()` should not throw any other exception when called with an unknown provider.
- The function `get_provider()` should not raise a different type of exception (e.g., TypeError) when called with an unknown provider.
- The function `get_provider()` should handle the unknown provider case correctly and return None or another appropriate value.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 245, 247, 249, 252, 257, 262, 267) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm.py::TestLlmProviderContract::test_noop_implements_interface` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test that NoopProvider implements all required interface methods.

**Why Needed:** This test prevents a potential regression where the NoopProvider class is not properly implemented.

**Key Assertions:**

- The provider should have annotate method
- The provider should have is_available method
- The provider should have get_model_name method
- The provider should have config attribute

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53) |
| src/pytest_llm_report/llm/noop.py | 1 lines (ranges: 32) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_llm.py::TestNoopProvider::test_annotate_returns_empty 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the NoopProvider returns an empty annotation when no annotation is provided.

**Why Needed:** This test prevents a regression where the LlmAnnotation class does not handle cases without any annotations correctly.

**Key Assertions:**

- annotation is of type LlmAnnotation
- annotation scenario is an empty string
- annotation why_needed is an empty string
- annotation key_assertions are an empty list

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80) |
| src/pytest_llm_report/llm/noop.py | 2 lines (ranges: 32, 50) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm.py::TestNoopProvider::test_get_model_name_empty` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the `get_model_name` method of the `NoopProvider` class returns an empty string when given an empty configuration.

**Why Needed:** This test prevents a potential bug where an empty configuration causes the provider to fail or produce incorrect results.

**Key Assertions:**

- assert provider.get_model_name() == ''
- assert isinstance(provider.get_model_name(), str)
- assert provider.get_model_name() != 'noop' # This should never be true in a valid case

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53) |
| src/pytest_llm_report/llm/noop.py | 2 lines (ranges: 32, 66) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm.py::TestNoopProvider::test_is_available` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Verify that the NoopProvider instance is available.

**Why Needed:** Prevents a potential bug where the provider might not be available due to configuration issues or other internal errors.

**Key Assertions:**

- The `provider.is_available()` method should return True.
- The `provider` object should have an `is_available()` attribute that returns True.
- No exception should be raised when calling `provider.is_available()`.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 107, 110-111) |
| src/pytest_llm_report/llm/noop.py | 2 lines (ranges: 32, 58) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_annotator.py::test_annotate_tests_emits_summary` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** The test verifies that the annotation summary is printed when annotations run.

**Why Needed:** This test prevents a regression where the annotation summary is not printed for annotated tests.

**Key Assertions:**

- The function `annotate_tests` should print 'Annotated X test(s) via litellm' to the console.
- The function `annotate_tests` should capture and return the captured output.
- The function `annotate_tests` should call the `get_provider` method on the `FakeProvider` instance with a valid configuration.
- The annotation summary should be printed before any test runs.
- The annotation summary should not be printed for annotated tests that do not have an outcome of 'passed'.

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 | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/llm/annotator.py | 65 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-67, 71-72, 74-78, 87-92, 97-98, 100, 102, 104, 115-122, 129-135, 137, 139, 165-168, 170-171, 173-174, 176, 178, 180, 185-190, 192, 198, 203) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/prompts.py | 12 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-84) |

PASSED `tests/test_llm_annotator.py::test_annotate_tests_reports_progress` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Test that LLM annotation progress is reported via callback.

**Why Needed:** Prevents regression where the test case does not receive any LLM annotations.

**Key Assertions:**

- The expected message should be 'pytest-llm-report: Starting LLM annotations for 1 test(s)',
- The expected message should contain the name of the test case being annotated (tests/test_progress.py::test_case).
- The progress callback should append a new message to the list.
- The first message in the list should be 'pytest-llm-report: Starting LLM annotations for 1 test(s)',
- The second message in the list should contain the name of the test case being annotated (tests/test_progress.py::test_case).

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 | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/llm/annotator.py | 69 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-68, 71-72, 74-78, 87-92, 97-98, 100, 102, 104, 115-122, 129-135, 137, 139, 165-168, 170-171, 173-174, 176, 178, 180, 185-190, 192-195, 198, 203) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/prompts.py | 12 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-84) |

PASSED tests/test_llm_annotator.py::test_annotate_tests_respects_opt_out_and_limit 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Tests should respect opt-out and limit for LLM annotations.

**Why Needed:** This test prevents regression by ensuring that LLM annotations do not skip opt-out tests or exceed the maximum number of tests.

**Key Assertions:**

- Verify that only 'tests/test_a.py::test_a' is called when LLM annotations are enabled.
- Verify that no LLM annotation is returned for the second test.
- Verify that no LLM annotation is returned for the third test.
- Verify that the maximum number of tests (1) is respected by calling provider with llm_max_tests=1.
- Verify that the opt-out test ('tests/test_b.py::test_b') returns a non-LLM annotation.
- Verify that the limit of 1 LLM annotations is not exceeded.
- Verify that no other LLM annotations are returned for tests beyond the first one.

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 | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/llm/annotator.py | 65 lines (ranges: 45, 48-49, 56-57, 59, 61-62, 66-67, 71-72, 74-78, 87-92, 97-98, 100, 102, 104, 115-122, 129-135, 137, 139, 165-168, 170-171, 173-174, 176, 178, 180, 185-190, 192, 198, 203) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/prompts.py | 12 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-84) |

PASSED tests/test_llm_annotator.py::test_annotate_tests_respects_rate_limit 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Test that LLM annotations respect the requests-per-minute rate limit.

**Why Needed:** This test prevents a potential regression where the annotator does not respect the rate limit and makes unnecessary calls to the provider.

**Key Assertions:**

- The provider's `calls` attribute should contain a list of node IDs annotated by the LLM.
- The provider's `calls` attribute should contain two node IDs annotated by the LLM.
- The `sleep_calls` list should contain one value equal to 2.0 seconds.
- The `provider.calls` attribute should be equal to ['tests/test_a.py::test_a', 'tests/test_b.py::test_b']
- The `sleep_calls` list should not contain any values greater than or equal to 2.0 seconds.
- No other assertions are needed as the test verifies that the LLM annotations respect the rate limit.

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 | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/llm/annotator.py | 68 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-67, 71-72, 74-78, 87-92, 97-98, 100, 102, 104, 115-122, 129-135, 137, 139, 165-168, 170-173, 176, 178, 180-183, 185-190, 192, 198, 203) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/prompts.py | 12 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-84) |

PASSED `tests/test_llm_annotator.py::test_annotate_tests_skips_unavailable_provider` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test that annotation with unavailable providers is skipped.

**Why Needed:** To prevent skipping of tests due to unavailable LLM providers.

**Key Assertions:**

- The test verifies that the annotation process skips the unavailable provider.
- The test verifies that the annotation process displays a message indicating that the provider is not available.
- The test verifies that the annotation process does not skip all tests when an unavailable provider is detected.
- The test verifies that the annotation process logs the message in the captured output.
- The test verifies that the annotation process displays the message before skipping the test.
- The test verifies that the annotation process does not log any other messages related to the unavailable provider.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_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: 45, 48-52, 54)` |
| `src/pytest_llm_report/options.py` | `3 lines (ranges: 107, 147, 224)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_llm_annotator.py::test_annotate_tests_uses_cache` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Test that annotations are cached between runs and that the annotation function is called when needed.

**Why Needed:** This test prevents regression by ensuring that annotations are properly cached, which can cause issues if the test suite is run multiple times in a row without re-running the annotator.

**Key Assertions:**

- provider.calls == ['tests/test_sample.py::test_case']
- test.llm_annotation is not None
- test.llm_annotation.scenario == 'cached'

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/cache.py | 30 lines (ranges: 39-41, 53, 55-56, 58, 60-62, 68-73, 86, 90, 92, 94, 97-101, 103, 118-119, 121, 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 | 69 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-67, 71-72, 74-81, 87-92, 97-98, 100, 102, 104, 115-122, 127, 129-135, 137, 139, 165-168, 170-171, 173-174, 176, 178, 180, 185-190, 192, 198, 203) |
| src/pytest_llm_report/options.py | 3 lines (ranges: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/prompts.py | 12 lines (ranges: 33, 49, 52, 55, 58-59, 65, 78-79, 82-84) |

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_required_fields` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** The test verifies that the schema requires both 'scenario' and 'why_needed' fields.

**Why Needed:** This test prevents a regression where the schema is not enforced correctly, potentially leading to invalid data being accepted.

**Key Assertions:**

- assert 'scenario' in required
- assert 'why_needed' in required

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_schema_from_dict` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that the AnnotationSchema.from_dict function correctly parses a dictionary into an instance of AnnotationSchema.

**Why Needed:** This test prevents potential bugs or regressions in the AnnotationSchema class where it may not be able to parse dictionaries correctly, potentially leading to incorrect schema instances being created.

**Key Assertions:**

- checks password
- checks username
- the length of key_assertions is 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/llm/schemas.py` | 5 lines (ranges: 77-81) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_schema_handles_empty` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that the AnnotationSchema class can handle an empty input.

**Why Needed:** This test prevents a potential bug where the schema is not validated correctly for invalid or missing inputs.

**Key Assertions:**

- schema.scenario = "" (empty string)
- schema.why_needed = "" (no specific reason, but it's an important validation check)
- assert schema.scenario == "" (checks that the scenario attribute is set to an empty string)
- assert schema.why_needed == "" (checks that the why_needed attribute is set to an empty string)

COVERAGE

| | |
|---|---|
| `src/pytest_llm_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: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_schema_handles_partial` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that the `AnnotationSchema` class correctly handles partial input scenarios.

**Why Needed:** This test prevents a potential bug where the `AnnotationSchema` class does not handle partial input correctly, leading to incorrect validation or errors.

**Key Assertions:**

- The `schema.scenario` attribute is set to 'Partial only'.
- The `schema.why_needed` attribute is empty, indicating no specific issue with handling partial input.

COVERAGE

| | |
|---|---|
| src/pytest_llm_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: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_schema_has_required_fields` 1ms 🛡️ 2

AI ASSESSMENT

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

**Why Needed:** This test prevents a potential bug where the annotation schema does not have all necessary fields, potentially causing errors when validating annotations.

**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']

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestAnnotationSchema::test_schema_to_dict` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** TestAnnotationSchema::test_schema_to_dict verifies that the AnnotationSchema instance correctly serializes to a dictionary.

**Why Needed:** This test prevents regression by ensuring that the AnnotationSchema class properly converts its internal state into a dictionary representation.

**Key Assertions:**

- assertion 1: The 'scenario' key in the resulting dictionary matches the provided scenario string.
- assertion 2: The 'why_needed' key in the resulting dictionary matches the provided why_needed string.
- assertion 3: The 'key_assertions' key in the resulting dictionary contains all expected assertion strings.

COVERAGE

| | |
|---|---|
| src/pytest_llm_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: 380-381, 384, 388-390) |

PASSED tests/test_llm_contract.py::TestNoopProvider::test_noop_from_factory 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The factory function should return a NoopProvider instance when the provider is set to 'none'.

**Why Needed:** This test prevents a potential bug where the NoopProvider instance is not created for providers that are not explicitly specified.

**Key Assertions:**

- the `provider` attribute of the returned `NoopProvider` instance should be `None`.
- the `provider` attribute of the returned `NoopProvider` instance should be `None`.
- the `provider` attribute of the returned `NoopProvider` instance should be `None`.
- the `provider` attribute of the returned `NoopProvider` instance should be `None`.
- the `provider` attribute of the returned `NoopProvider` instance should be `None`.
- the `provider` attribute of the returned `NoopProvider` instance should be `None`.
- the `provider` attribute of the returned `NoopProvider` instance should be `None`.
- the `provider` attribute of the returned `NoopProvider` instance should be `None`.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 245, 247, 249-250) |
| src/pytest_llm_report/llm/noop.py | 1 lines (ranges: 32) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestNoopProvider::test_noop_is_llm_provider` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The `test_noop_is_llm_provider` test verifies that the `NoopProvider` class correctly returns an instance of `LlmProvider`.

**Why Needed:** This test prevents a potential bug where the `NoopProvider` class is incorrectly implemented as an LLM provider, leading to unexpected behavior or errors.

**Key Assertions:**

- The `provider` variable should be an instance of `LlmProvider`.
- The `provider` variable should have the correct type hint.
- The `provider` variable should not be a subclass of `LLMProvider`.
- The `provider` variable should not have any additional attributes or methods.
- The `provider` variable should only contain the necessary instance variables.
- The `provider` variable should not have any inherited properties from other classes.
- The `provider` variable should not be a mock object created with a different implementation.
- The `provider` variable should not have any side effects or external dependencies.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.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: 52-53) |
| `src/pytest_llm_report/llm/noop.py` | 1 lines (ranges: 32) |
| `src/pytest_llm_report/options.py` | 2 lines (ranges: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestNoopProvider::test_noop_returns_empty_annotation` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The NoopProvider returns an empty annotation for a function with no side effects or dependencies.

**Why Needed:** This test prevents regression in the case where a function has no side effects or dependencies, as it ensures that the annotation is correctly set to an empty string.

**Key Assertions:**

- The annotation returned by `provider.annotate` is not equal to `None`.
- The annotation returned by `provider.annotate` does not contain any of the expected keys.
- The annotation returned by `provider.annotate` has a value that is not empty.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80) |
| src/pytest_llm_report/llm/noop.py | 2 lines (ranges: 32, 50) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestProviderContract::test_annotate_returns_annotation` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the `provider.annotate` method returns an instance of LlmAnnotation-like object.

**Why Needed:** This test prevents a potential regression where the annotation result is not properly populated with expected attributes.

**Key Assertions:**

- assert hasattr(result, 'scenario')
- assert hasattr(result, 'why_needed')
- assert hasattr(result, 'key_assertions')

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80) |
| src/pytest_llm_report/llm/noop.py | 2 lines (ranges: 32, 50) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestProviderContract::test_provider_handles_empty_code` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the ProviderContract handles an empty code by passing it through without any issues.

**Why Needed:** This test prevents a potential bug where the contract might throw an error or raise an exception when given an empty code.

**Key Assertions:**

- The provider should not return None for the provided test.
- The provider should handle the empty code gracefully by returning a valid result.
- The provider should not throw any errors or exceptions when given an empty code.
- The provider's annotation process should be able to identify and handle the empty code correctly.
- The test should fail if the provider returns None for the provided test.
- The test should pass if the provider handles the empty code correctly.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80) |
| src/pytest_llm_report/llm/noop.py | 2 lines (ranges: 32, 50) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestProviderContract::test_provider_handles_none_context` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the `provider` function handles a `None` context for the `annotate` method correctly.

**Why Needed:** This test prevents potential bugs or regressions where the `provider` function may throw an exception or return incorrect results when given a `None` context.

**Key Assertions:**

- The `provider.annotate` method is called with a `None` value for the `code` parameter.
- An instance of `TestCaseResult` is returned from the `provider.annotate` method.
- The `result` variable is not `None` after calling `provider.annotate`.
- The `provider.annotate` method does not throw an exception or raise an error when given a `None` context.
- The `provider.annotate` method returns a valid instance of `TestCaseResult` even when given a `None` context.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80) |
| src/pytest_llm_report/llm/noop.py | 2 lines (ranges: 32, 50) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_contract.py::TestProviderContract::test_provider_has_annotate_method` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Test that all providers have an annotate method.

**Why Needed:** Prevents regression in LLM contract where some providers may not be able to annotate.

**Key Assertions:**

- The provider has an 'annotate' attribute.
- The 'annotate' function is callable.
- All providers are tested for this functionality.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 245, 247, 249-250, 252-253, 255, 257-258, 260, 262-263, 265) |
| src/pytest_llm_report/llm/gemini.py | 7 lines (ranges: 134, 136-139, 141-142) |
| src/pytest_llm_report/llm/noop.py | 1 lines (ranges: 32) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_handles_context_too_large` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The `annotate` method of the `GeminiProvider` class is being tested when it handles a context that is too large.

**Why Needed:** This test prevents a potential issue where the `annotate` method may throw an exception or behave unexpectedly due to the size of the context.

**Key Assertions:**

- The `context_size` attribute of the `GeminiProvider` instance should be less than 1000000.
- The `annotate` method should not raise a `MemoryError` when processing a context with a large size.
- The `context` variable should have a value that is within the expected range for the `GeminiProvider` class.
- No exception should be raised when calling `annotate` on a `GeminiProvider` instance with a context that is too large.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75-76, 80, 165, 167, 175) |
| src/pytest_llm_report/llm/gemini.py | 155 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-187, 200-202, 206-208, 210, 213-214, 217-221, 233, 245-248, 274-277, 280-283, 286-290, 292-295, 297-298, 300-301, 315, 317-320, 322-325, 327-328, 346, 348-350, 352-355, 360-363, 374-377, 381-382, 385-387, 391-392, 396-399, 401-402, 405, 408-410, 412-414, 417-418, 428, 430-434, 437-440, 442-443, 445-447) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_llm_providers.py::TestGeminiProvider::test_annotate_missing_dependency 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test that LiteLLMProvider annotates missing dependencies correctly.

**Why Needed:** This test prevents a potential bug where the provider does not report missing dependencies.

**Key Assertions:**

- The annotation error message is correct and includes the name of the missing dependency.
- The annotation error message is informative and provides necessary instructions to install the required package.
- The annotation error message is consistent across different environments (e.g., local, remote).
- The provider does not report missing dependencies when they are installed via pip.
- The provider reports missing dependencies cleanly without any unnecessary or misleading information.
- The test case outcome is correctly set to 'passed' even if the dependency is not installed.
- The annotation error message includes the correct package name (litellm) and a clear installation instruction.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80) |
| src/pytest_llm_report/llm/gemini.py | 12 lines (ranges: 134, 136-139, 141-142, 160-164) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_missing_token` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test that a GeminiProvider annotation fails when API token is missing.

**Why Needed:** To prevent the assertion error 'GEMINI_API_TOKEN is not set' when the API token is not provided.

**Key Assertions:**

- The function `annotation.error` should return the string "GEMINI_API_TOKEN is not set".
- The function `provider.annotate()` should raise an exception with the message "GEMINI_API_TOKEN is not set".
- The function `test_case()` should throw a `AssertionError` with the message "GEMINI_API_TOKEN is not set".

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80) |
| src/pytest_llm_report/llm/gemini.py | 12 lines (ranges: 134, 136-139, 141-142, 160-161, 167-169) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_records_tokens` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Test that tokens recorded on limiter are verified correctly.

**Why Needed:** Prevents regressions where tokens are not recorded or reported correctly.

**Key Assertions:**

- The provider annotated the test function with a valid annotation.
- The provider's rate limits logic ran without error and recorded the correct number of tokens.
- The limiter is not None and has at least one token usage entry.
- Each token usage entry has a total count equal to 123.
- The limiter uses the 'gemini-1.5-pro' model for token recording.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/llm/base.py` | 26 lines (ranges: 52-53, 72, 75, 80, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| `src/pytest_llm_report/llm/gemini.py` | 183 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, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-187, 200-202, 206-208, 210, 213-214, 217-223, 225-227, 274-277, 280-283, 286-290, 292-295, 297-298, 300-301, 315, 317-320, 322-325, 327-328, 330, 335-343, 346, 348-350, 352-355, 360-366, 368, 370-371, 374-377, 381-382, 385-387, 391-392, 396-399, 401-402, 405, 408-410, 412-414, 417, 419, 421-424, 428, 430-434, 437-440, 442-443, 445-447) |
| `src/pytest_llm_report/llm/schemas.py` | 7 lines (ranges: 38, 42-43, 50-53) |
| `src/pytest_llm_report/options.py` | 2 lines (ranges: 107, 147) |

| | |
|---|---|
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_retries_on_rate_limit` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Verify that the LLM provider annotates retries on rate limit violations

**Why Needed:** This test prevents a potential regression where the LLM provider does not retry when rate limiting occurs.

**Key Assertions:**

- The function calls `self._llm.annotate_retries()` after each API call to retry if rate limiting is exceeded
- The function logs an error message indicating that retries are being attempted due to rate limit violations
- The function checks the LLM's internal state to determine whether a retry attempt has already been made
- The function increments the retry count for the LLM before attempting another API call
- The function calls `self._llm.annotate_retries()` after each API call to retry if rate limiting is exceeded
- The function logs an error message indicating that retries are being attempted due to rate limit violations

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/llm/base.py | 26 lines (ranges: 52-53, 72, 75, 80, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| src/pytest_llm_report/llm/gemini.py | 181 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, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-187, 200-202, 206-208, 210, 213-214, 217-222, 225-227, 233-234, 238-240, 242-243, 274-277, 280-283, 286, 292-295, 297-298, 300-301, 315, 317-320, 322-325, 327-328, 330-333, 335-341, 343, 346, 348-350, 352-355, 360-366, 368-369, 374-377, 381-382, 385-387, 391-392, 396-399, 401-402, 405, 408-410, 412-414, 417, 419, 421-424, 428, |

| | |
|---|---|
| | 430-434, 437-440, 442-443, 445-447) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_rotates_models_on_daily_limit` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** The `annotate` method of the `GeminiProvider` class rotates models on a daily limit when annotating with `rotate_models_on_daily_limit=True`.

**Why Needed:** This test prevents regression where the model rotation is not applied correctly due to an incorrect implementation.

**Key Assertions:**

- The `annotate` method of the `GeminiProvider` class rotates models on the specified daily limit.
- The `rotate_models_on_daily_limit=True` parameter has no effect if the `annotate` method does not rotate models.
- Models are rotated correctly when the `annotate` method is called with `rotate_models_on_daily_limit=True`.
- The `annotate` method rotates models on the specified daily limit, even if the model size exceeds the limit.
- The `rotate_models_on_daily_limit=False` parameter has no effect on the rotation of models.
- Models are not rotated when the `annotate` method is called with `rotate_models_on_daily_limit=False`.
- The `annotate` method rotates models correctly when the daily limit is exceeded, but the model size does not exceed it.
- The `annotate` method rotates models on the specified daily limit, even if the model size exceeds a certain threshold.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/llm/base.py` | 26 lines (ranges: 52-53, 72, 75, 80, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| `src/pytest_llm_report/llm/gemini.py` | 177 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, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-187, 200-208, 210, 213-214, 217-222, 225-227, 274-277, 280-283, 286-290, 292-295, 297-298, 300-301, 315, 317-320, 322-325, 327-328, 330, |

|  |  |
|---|---|
|  | 335-341, 343, 346, 348-350, 352-355, 360-366, 368, 370, 372-377, 381-382, 385-387, 391-393, 396-399, 401-402, 405, 408-410, 412-414, 417, 419-420, 428, 430-434, 437-440, 442-443, 445-447) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_annotate_skips_on_daily_limit` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** The test verifies that annotating a model with the `GeminiProvider` skips daily limits.

**Why Needed:** This test prevents regression when using the `GeminiProvider` to annotate models without exceeding daily limits.

**Key Assertions:**

- Annotates a model without exceeding daily limit
- Does not skip annotation due to daily limit
- Uses correct `GeminiProvider` instance for annotating
- Exceeds daily limit when annotating with `GeminiProvider`
- Skips annotation due to daily limit when using other provider
- Error message is correct when exceeding daily limit

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/llm/base.py` | 26 lines (ranges: 52-53, 72, 75, 80, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| `src/pytest_llm_report/llm/gemini.py` | 184 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, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-187, 200-208, 210-211, 213-214, 217-222, 225-227, 252-254, 274-277, 280-283, 286-290, 292-295, 297-298, 300-301, 315, 317-320, 322-325, 327-328, 330, 335-341, 343, 346, 348-350, 352-355, 360-366, 368, 370, 372-377, 381-382, 385-387, 391-393, 396-399, 401-402, 405, 408-410, 412-414, 417, 419, 421-424, 428, 430-434, 437-440, 442-443, 445-447) |
| `src/pytest_llm_report/llm/schemas.py` | 7 lines (ranges: 38, 42-43, 50-53) |
| `src/pytest_llm_report/options.py` | 2 lines (ranges: 107, 147) |

| | |
|---|---|
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_llm_providers.py::TestGeminiProvider::test_annotate_success_with_mock_response 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Test that the annotate method of LiteLLMProvider returns an LlmAnnotation object with the correct scenario, why_needed, key_assertions, confidence, model, and messages.

**Why Needed:** Prevents a potential regression where the annotation is not correctly set for a valid response payload.

**Key Assertions:**

- annotation.scenario == 'Checks login'
- annotation.why_needed == 'Stops regressions'
- annotation.key_assertions == ['status ok', 'redirect']
- annotation.confidence == 0.8
- captured['model'] == 'gpt-4o'
- captured['messages'][0]['role'] == 'system'

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/llm/base.py | 26 lines (ranges: 52-53, 72, 75, 80, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| src/pytest_llm_report/llm/gemini.py | 177 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, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-187, 200-202, 206-208, 210, 213-214, 217-222, 225-227, 274-277, 280-283, 286, 292-295, 297-298, 300-301, 315, 317-320, 322-325, 327-328, 330, 335-341, 343, 346, 348-350, 352-355, 360-366, 368-377, 381-382, 385-387, 391-392, 396-399, 401-402, 405, 408-410, 412-414, 417, 419, 421-424, 428, 430-434, 437-440, 442-443, 445-447) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |

| | |
|---|---|
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_exhausted_model_recovers_after_24h` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** The test verifies that the exhausted model recovers after 24 hours.

**Why Needed:** This test prevents a potential regression where the model does not recover from exhaustion within 24 hours.

**Key Assertions:**

- The recovered model should have the same accuracy as before exhaustion.
- The recovered model should have the same number of parameters as before exhaustion.
- The recovered model should be able to make predictions without errors.
- The recovered model's metrics (e.g. precision, recall) should not have changed significantly after 24 hours.
- The recovered model's training time should have decreased by a significant amount within 24 hours.
- The recovered model's memory usage should have decreased significantly within 24 hours.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/llm/base.py` | 26 lines (ranges: 52-53, 72, 75, 80, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| `src/pytest_llm_report/llm/gemini.py` | 190 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, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-188, 190-191, 193-194, 196, 200-208, 210-211, 213-214, 217-222, 225-227, 252-254, 274-277, 280-283, 286-290, 292-295, 297-298, 300-301, 315, 317-320, 322-325, 327-328, 330, 335-341, 343, 346, 348-350, 352-355, 360-366, 368, 370, 372-377, 381-382, 385-387, 391-393, 396-399, 401-402, 405, 408-410, 412-414, 417, 419, 421-424, 428, 430-434, 437-440, 442-443, 445-447) |

| | |
|---|---|
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestGeminiProvider::test_fetch_available_models_error` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The `fetch_available_models` method of a Gemini provider returns an error when no models are available.

**Why Needed:** This test prevents a potential regression where the `fetch_available_models` method fails to return an error when no models are available, potentially causing downstream tests to fail.

**Key Assertions:**

- The `fetch_available_models` method should raise a `GeminiError` with a suitable message.
- The `fetch_available_models` method should not return any models when there are none available.
- A suitable error message should be included in the `GeminiError` instance returned by `fetch_available_models`.
- The `fetch_available_models` method should raise an exception instead of returning a value when no models are available.
- The `fetch_available_models` method should not return any values when there are none available, including None or empty lists.
- A clear and descriptive error message should be included in the `GeminiError` instance returned by `fetch_available_models`.
- The `fetch_available_models` method should raise a `GeminiError` with a specific code (e.g., 404) when no models are available, rather than raising a generic exception.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53) |
| src/pytest_llm_report/llm/gemini.py | 65 lines (ranges: 134, 136-139, 141-142, 280, 282-283, 286-290, 292-295, 297-298, 300-301, 346, 348-350, 352-355, 360-363, 374-377, 385, 387, 391-392, 396-402, 405, 408-410, 412-414, 417-418, 428, 430-432, 435-436) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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 when the LLM provider is used with a refresh interval.

**Why Needed:** This test prevents regression in the LLM provider's behavior when using a refresh interval.

**Key Assertions:**

- The `refresh_interval` attribute of the LLM provider should be updated correctly after each interval.
- The model list should contain all models that were available at the start of the interval.
- No new models should be added to the model list during the interval.
- All existing models should still be present in the model list at the end of the interval.
- The `refresh_interval` attribute should be updated correctly after each interval even if no models are added or removed.
- If a refresh interval is not provided, the LLM provider should use the default interval (e.g. 1 hour).
- The LLM provider's behavior should not cause any unexpected behavior when using a refresh interval.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/llm/base.py` | 26 lines (ranges: 52-53, 72, 75, 80, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| `src/pytest_llm_report/llm/gemini.py` | 169 lines (ranges: 39-42, 45-46, 48, 52-54, 73, 76-78, 81-82, 84, 87-88, 92-93, 95-96, 100-102, 134, 136-139, 141-142, 160-161, 167-168, 171-172, 174, 176-184, 186-187, 200-202, 206-208, 210, 213-214, 217-222, 225-227, 274-277, 280-283, 286, 292-295, 297-298, 300-301, 315, 317-320, 322-325, 327-328, 330, 335-341, 343, 346, 348-350, 352-355, 360-366, 368-369, 374-377, 381-382, 385-387, 391-393, 396-399, 401-402, 405, 408-410, 412-414, 417, |

| | |
|---|---|
| | 419, 421-424, 428, 430-434, 437-440, 442-443, 445-447) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_handles_completion_error` 90.00s 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the LiteLLMProvider annotates completion errors correctly.

**Why Needed:** This test prevents regression where the LiteLLMProvider does not surface completion errors in annotations.

**Key Assertions:**

- The 'boom' error is present in the annotation.
- The 'boom' error is contained within the annotation's error message.
- The LiteLLMProvider correctly surfaces the 'boom' error when it occurs during annotation.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80, 165, 167, 175) |
| src/pytest_llm_report/llm/litellm_provider.py | 22 lines (ranges: 37-38, 44, 46, 49, 51-52, 54-60, 62-63, 78-79, 81-82, 84-85) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_invalid_key_assertions` 90.00s 🛡️ 6

AI ASSESSMENT

**Scenario:** Test that LiteLLMProvider rejects invalid key_assertions payloads.

**Why Needed:** To prevent the provider from incorrectly handling cases with invalid key_assertions payloads.

**Key Assertions:**

- Invalid response: key_assertions must be a list
- Missing or empty list of key_assertions

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207-209) |
| src/pytest_llm_report/llm/litellm_provider.py | 25 lines (ranges: 37-38, 44, 46, 49, 51-52, 54-60, 62-63, 66-67, 69, 73, 76, 81-82, 84-85) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_missing_dependency` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The LiteLLMProvider annotates a missing dependency in the provided test case.

**Why Needed:** This test prevents regression when the 'litellm' library is not installed, ensuring that tests are executed with the correct error message.

**Key Assertions:**

- assert annotation.error == 'litellm not installed. Install with: pip install litellm'
- provider.annotate(test, 'def test_case(): assert True')
- testid = test.nodeid
- outcome = test.outcome
- nodeid = config.provider
- config = Config(provider=nodeid)
- litellmProvider = LiteLLMProvider(config)
- test = CaseResult(nodeid='tests/test_sample.py::test_case', outcome='passed')

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.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: 52-53, 72, 75, 80)` |
| `src/pytest_llm_report/llm/litellm_provider.py` | `5 lines (ranges: 37-41)` |
| `src/pytest_llm_report/options.py` | `2 lines (ranges: 107, 147)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_annotate_success_with_mock_response` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Test that the LiteLLM provider annotates a successful response correctly.

**Why Needed:** Prevents regressions caused by incorrect annotation of mock responses with fake completion functions.

**Key Assertions:**

- The annotation has the correct scenario, why needed message, and key assertions.
- The annotation's confidence level is set to 0.8 as expected.
- The captured model matches the one provided in the configuration.
- The 'tests/test_auth.py::test_login' message is present in the mock response.
- The function being annotated has a role of 'system'.
- The test login function is present in the mock response's content.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/llm/base.py` | 26 lines (ranges: 52-53, 72, 75, 80, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| `src/pytest_llm_report/llm/litellm_provider.py` | 20 lines (ranges: 37-38, 44, 46, 49, 51-52, 54-60, 62-63, 66-67, 69-70) |
| `src/pytest_llm_report/llm/schemas.py` | 7 lines (ranges: 38, 42-43, 50-53) |
| `src/pytest_llm_report/options.py` | 2 lines (ranges: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestLiteLLMProvider::test_is_available_with_module` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Verifies that the LiteLLM provider detects installed modules correctly.

**Why Needed:** This test prevents a bug where the provider does not detect installed modules.

**Key Assertions:**

- The `is_available()` method of the `LiteLLMProvider` class returns True when the module is available.
- The `litellm` module is imported from the system's modules list.
- The `liteellm` attribute of the fake_litellm object is set to the imported module.
- The provider instance has an `is_available()` method that checks for the presence of the module.
- The test passes when the module is installed and available in the system's modules list.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 107, 110-111) |
| src/pytest_llm_report/llm/litellm_provider.py | 3 lines (ranges: 94-95, 97) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_annotate_fallbacks_on_context_length_error` 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** The test verifies that the annotate fallbacks on context length error are handled correctly.

**Why Needed:** This test prevents a regression where the annotation fails when the input context is too long.

**Key Assertions:**

- The function `annotate` should not raise an exception when the input context is too long.
- The function `annotate` should return an error message indicating that the context length exceeds the maximum allowed value.
- The function `annotate` should update the output with a fallback annotation instead of raising an exception.
- The function `annotate` should handle the case where the input context is exactly equal to the maximum allowed value without raising an exception or returning an error message.
- The function `annotate` should not raise an exception when the input context is too short (less than 1 token).
- The function `annotate` should return a fallback annotation with the correct type and format for the context length.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/llm/base.py | 33 lines (ranges: 52-53, 72, 75-76, 78, 165, 167-173, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| src/pytest_llm_report/llm/ollama.py | 15 lines (ranges: 40-41, 47, 50, 52, 54-55, 57-60, 62-63, 66-67) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_annotate_handles_call_error` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test OllamaProvider::test_annotate_handles_call_error verifies that the annotate method returns an error message when a call to _call_ollama raises a RuntimeError.

**Why Needed:** This test prevents regression where the annotation fails with a generic 'Failed after 10 retries. Last error: boom' error message instead of raising a specific exception like RuntimeError.

**Key Assertions:**

- The annotate method should raise a RuntimeError when _call_ollama raises a RuntimeError.
- The annotate method should return the exact same error message as the last error raised by _call_ollama.
- The annotation should include the full stack trace of the error that caused the call to _call_ollama.
- The annotation should not raise a RuntimeError when _call_ollama does not raise a RuntimeError.
- The annotation should return an error message with a specific prefix (e.g. 'Failed after 10 retries.')
- The annotation should include the last error that caused the call to _call_ollama in the error message.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80, 165, 167, 175) |
| src/pytest_llm_report/llm/ollama.py | 16 lines (ranges: 40-41, 47, 50, 52, 54-55, 57-59, 71-72, 74-75, 77-78) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

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 case.

**Why Needed:** This test prevents the provider from incorrectly reporting missing dependencies and causing downstream issues.

**Key Assertions:**

- assert annotation.error == 'httpx not installed. Install with: pip install httpx'
- provider.annotate(test, 'def test_case(): assert True')
- test CaseResult(nodeid='tests/test_sample.py::test_case', outcome='passed')

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 72, 75, 80) |
| src/pytest_llm_report/llm/ollama.py | 5 lines (ranges: 40-44) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_llm_providers.py::TestOllamaProvider::test_annotate_success_full_flow 1ms 🛡️ 6

AI ASSESSMENT

**Scenario:** Test the full annotation flow for Ollama provider with mocked HTTP.

**Why Needed:** Prevents authentication bugs by verifying that the response from the API contains a valid token.

**Key Assertions:**

- check status
- validate token

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/llm/base.py | 26 lines (ranges: 52-53, 72, 75, 80, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| src/pytest_llm_report/llm/ollama.py | 29 lines (ranges: 40-41, 47, 50, 52, 54-55, 57-60, 62-63, 114, 116-123, 127-130, 132, 134-135) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_call_ollama_success` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The Ollama provider makes the correct API call to generate a response.

**Why Needed:** This test prevents regression in case the Ollama provider fails to make the correct API call.

**Key Assertions:**

- The 'url' captured is set to 'http://localhost:11434/api/generate'.
- The 'json' captured contains a 'model' key with value 'llama3.2:1b'.
- The 'json' captured contains a 'prompt' key with value 'test prompt'.
- The 'json' captured contains a 'system' key with value 'system prompt'.
- The 'json' captured does not contain a 'stream' key.
- The 'timeout' captured is set to 60 seconds.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53) |
| src/pytest_llm_report/llm/ollama.py | 16 lines (ranges: 114, 116-123, 127-130, 132, 134-135) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_call_ollama_uses_default_model` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test that the default model is used when not specified for Ollama provider.

**Why Needed:** Prevents a bug where the user has to specify the model manually, making the API more intuitive and easier to use.

**Key Assertions:**

- The captured JSON from the request contains the 'model' key with the default value of 'llama3.2'.
- The provider's _call_ollama method is called with an empty string as the model parameter.
- The captured JSON does not contain any other information that would indicate a different model was used.
- The provider's _call_ollama method raises an exception if the model is specified, but this test prevents it from happening by default.
- The captured JSON contains the 'response' key with the value 'ok', which indicates that the API responded successfully.
- The captured JSON does not contain any other information that would indicate a different response code or status.
- The provider's _call_ollama method is called without any arguments, which means it uses the default model.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.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: 52-53) |
| `src/pytest_llm_report/llm/ollama.py` | 16 lines (ranges: 114, 116-123, 127-130, 132, 134-135) |
| `src/pytest_llm_report/options.py` | 2 lines (ranges: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_check_availability_failure` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the Ollama provider returns False when the server is unavailable.

**Why Needed:** This test prevents a regression where the provider incorrectly reports availability when the server is down.

**Key Assertions:**

- the function _check_availability() of the OllamaProvider instance should raise an exception or return a specific value indicating that the server is unavailable.
- the function _check_availability() of the OllamaProvider instance should not return False by default when the server is available.
- the function _check_availability() of the OllamaProvider instance should raise a ConnectionError with the correct message when the server is down.
- the function _check_availability() of the OllamaProvider instance should be able to distinguish between a normal connection error and a server not running error.
- the function _check_availability() of the OllamaProvider instance should not return False for a known working URL.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53) |
| src/pytest_llm_report/llm/ollama.py | 6 lines (ranges: 87-88, 90-91, 93-94) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_check_availability_non_200` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test that the Ollama provider returns False for non-200 status codes when checking availability.

**Why Needed:** This test prevents a regression where the provider incorrectly returns True for non-200 status codes, potentially leading to unexpected behavior or errors in downstream applications.

**Key Assertions:**

- The method _check_availability() of the OllamaProvider instance is called with no arguments.
- The return value of _check_availability() is set to False.
- The provider._check_availability() method is not called with a valid status code (200 or above).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53) |
| src/pytest_llm_report/llm/ollama.py | 5 lines (ranges: 87-88, 90-92) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_check_availability_success` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test checks availability of Ollama provider via /api/tags endpoint.

**Why Needed:** Prevents a potential bug where the provider does not respond to requests for tags.

**Key Assertions:**

- The provider's _check_availability method should return True when the /api/tags endpoint is available.
- The provider's _check_availability method should raise an exception when the /api/tags endpoint is unavailable.
- The provider's _check_availability method should not throw a TypeError when the /api/tags endpoint is unavailable.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53) |
| src/pytest_llm_report/llm/ollama.py | 5 lines (ranges: 87-88, 90-92) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_llm_providers.py::TestOllamaProvider::test_is_local_returns_true 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The Ollama provider should always return `is_local=True`.

**Why Needed:** This test prevents a potential regression where the provider might not return `is_local=True` when it's supposed to, potentially causing issues with downstream dependencies.

**Key Assertions:**

- provider.is_local() == True
- provider.config.provider == 'ollama'
- config is not None
- provider is an instance of OllamaProvider
- is_local is a property of provider
- is_local is set to True for the given config
- is_local is always returned as True

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53) |
| src/pytest_llm_report/llm/ollama.py | 1 lines (ranges: 102) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_invalid_json 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the `OllamaProvider` class throws an error when parsing a response with invalid JSON.

**Why Needed:** This test prevents a potential bug where the Ollama provider incorrectly interprets valid JSON responses and reports an error instead of attempting to parse it as LLM output.

**Key Assertions:**

- The `annotation.error` attribute is set to 'Failed to parse LLM response as JSON'.
- The `provider._parse_response` method returns a `ConfigError` exception with the message 'Failed to parse LLM response as JSON'.
- The error message indicates that the provided JSON string is not valid.
- The test verifies that the provider correctly raises an exception when encountering invalid JSON.
- The test ensures that the provider does not attempt to parse the invalid JSON as a valid LLM output.
- The `provider._parse_response` method checks for specific error conditions and returns an exception accordingly.
- The `ConfigError` exception is raised with a meaningful error message indicating the problem with the provided JSON string.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 186-187, 190-192) |
| 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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_invalid_key_assertions 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the Ollama provider rejects invalid key_assertions payloads in its _parse_response method.

**Why Needed:** This test prevents regression where the Ollama provider incorrectly handles invalid key_assertions payloads, potentially causing unexpected behavior or errors.

**Key Assertions:**

- The 'key_assertions' field must be a list of strings.
- The 'key_assertions' field should contain at least one string value.
- The 'key_assertions' field should not be empty.
- The 'key_assertions' field should only contain valid key_assertions payloads.
- The 'key_assertions' field should not contain any invalid or malformed key_assertions payloads.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 186-187, 190-191, 194-195, 198-200, 203, 205, 207-209) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_code_fence` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The provided test verifies that the Ollama provider correctly parses a JSON response from a markdown code fence.

**Why Needed:** This test prevents regression in the LLM providers, as it ensures the provider can extract meaningful JSON data from code fences.

**Key Assertions:**

- The response is not empty.
- The response contains valid JSON syntax (i.e., no invalid characters or formatting).
- The response does not contain any extraneous whitespace or line breaks.
- The response does not contain any nested objects or arrays with only one element.
- The response contains a single object with the following properties: `text` and `json`.
- The JSON object has the expected structure (i.e., it is an object with `text` and `json` keys).
- The JSON object does not contain any extraneous or redundant data.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| src/pytest_llm_report/llm/schemas.py | 6 lines (ranges: 38, 42-44, 46-47) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_json_in_plain_fence` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The provided test verifies that the Ollama provider can extract JSON from a plain markdown fence without any language specification.

**Why Needed:** This test prevents regression in the case where the input contains no language, as it ensures the provider correctly extracts the JSON content.

**Key Assertions:**

- The response is not empty.
- The response starts with `"`.
- The response ends with `"}"`.
- The response does not contain any special characters or formatting.
- The response only contains whitespace and no line breaks.
- The response does not contain any JSON syntax (e.g., `{`, `}`, `[`, `]`).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| src/pytest_llm_report/llm/schemas.py | 6 lines (ranges: 38, 42-44, 46-47) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_llm_providers.py::TestOllamaProvider::test_parse_response_success` 1ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test Ollama provider parses valid JSON responses with success scenario.

**Why Needed:** Prevents potential bugs in the LLM providers by ensuring correct parsing of valid JSON responses.

**Key Assertions:**

- assert a is not None
- assert b is not None

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.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: 52-53, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_models.py::TestArtifactEntry::test_to_dict 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that `CoverageEntry.to_dict()` correctly serializes the object.

**Why Needed:** This test prevents a potential bug where the serialized representation of `CoverageEntry` is incorrect, potentially leading to data corruption or loss during storage or transmission.

**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 value.
- The 'line_count' key in the serialized dictionary matches the expected value.

COVERAGE

| | |
|---|---|
| src/pytest_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: 254-257) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestCollectionError::test_to_dict` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Ensures that the `CoverageEntry` class correctly serializes a coverage entry into a dictionary.

**Why Needed:** This test prevents a potential bug where the `CoverageEntry` class does not properly serialize its internal data, causing incorrect results when comparing against expected dictionaries.

**Key Assertions:**

- The 'file_path' key in the serialized dictionary should match the original value.
- The 'line_ranges' key in the serialized dictionary should match the original value.
- The 'line_count' key in the serialized dictionary should match the original value.
- The 'coverage_data' key (if present) should not be included in the serialized dictionary.
- Any additional keys or values in the serialized dictionary should only include those that are expected based on the `CoverageEntry` class's internal data.
- The serialized dictionary should have the same structure and order as the original `CoverageEntry` object.
- Any unexpected keys or values in the serialized dictionary should raise an AssertionError.

COVERAGE

| | |
|---|---|
| src/pytest_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: 207-209) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestCoverageEntry::test_to_dict` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test coverage entry serialization.

**Why Needed:** This test prevents a potential bug where the coverage entry is not correctly serialized to JSON.

**Key Assertions:**

- The 'file_path' key in the dictionary should be equal to 'src/foo.py'.
- The 'line_ranges' key in the dictionary should be equal to '1-3, 5, 10-15'.
- The 'line_count' key in the dictionary should be equal to 10.
- The 'file_path' value is not a string.
- The 'line_ranges' value is not a string or an array of strings.
- The 'line_ranges' value contains non-string values (e.g., integers).
- The 'line_count' value is not an integer.

COVERAGE

| | |
|---|---|
| `src/pytest_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: 40-43)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_models.py::TestLlmAnnotation::test_empty_annotation` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** An empty annotation should be created with default values.

**Why Needed:** The test prevents a potential bug where an empty annotation does not have any key-value pairs or default values.

**Key Assertions:**

- annotation.scenario == "" (empty string)
- annotation.why_needed == "Empty annotation should have default values." (description of why the annotation is expected to be created with default values)
- annotation.key_assertions == [] (expected empty list of key-value pairs)
- assert annotation.confidence is None (expected confidence value to be None for an empty annotation)
- assert annotation.error is None (expected error message to be None for an empty annotation)

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_models.py::TestLlmAnnotation::test_to_dict_minimal` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that the `to_dict` method of `LlmAnnotation` returns a dictionary with required fields.

**Why Needed:** This test prevents a potential bug where the minimal annotation is missing some required fields.

**Key Assertions:**

- required_fields = ['scenario', 'why_needed', 'key_assertions', 'confidence']
- annotation.to_dict() should return a dictionary containing all these keys
- assert 'scenario' in annotation.to_dict()
- assert 'why_needed' in annotation.to_dict()
- assert 'key_assertions' in annotation.to_dict()
- assert 'confidence' not in annotation.to_dict()

COVERAGE

| | |
|---|---|
| `src/pytest_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: 104-107, 109, 111, 113, 115)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_models.py::TestLlmAnnotation::test_to_dict_with_all_fields` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test to dictionary with all fields

**Why Needed:** Prevents incorrect LLM annotation due to missing confidence value.

**Key Assertions:**

- Asserts that the 'confidence' key is present and has a value of 0.95.
- Asserts that the 'context_summary' key contains the expected mode ('minimal') and bytes (1000).
- Asserts that the 'scenario' key matches the expected value ('Tests user login').

COVERAGE

| | |
|---|---|
| src/pytest_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 | 10 lines (ranges: 104-107, 109-111, 113-115) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestReportRoot::test_default_report` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the default report of ReportRoot.

**Why Needed:** Prevents regression when creating a new ReportRoot instance with no tests.

**Key Assertions:**

- The 'schema_version' key should be set to the current schema version.
- The 'tests' key should be an empty list.
- The 'warnings' key should not exist in the report dictionary.
- The 'collection_errors' key should not exist in the report dictionary.

COVERAGE

| | |
|---|---|
| src/pytest_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: 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_models.py::TestReportRoot::test_report_with_collection_errors 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test Report Root: test_report_with_collection_errors verifies that the report includes collection errors.

**Why Needed:** This test prevents a regression where the report does not include all collection errors.

**Key Assertions:**

- The report should contain at least one collection error.
- The first collection error should have a nodeid of 'test_bad.py'.
- All collection errors should be included in the report.

COVERAGE

| | |
|---|---|
| src/pytest_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: 207-209, 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508-510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestReportRoot::test_report_with_warnings` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test reports include warnings as expected.

**Why Needed:** This test prevents a regression where the report does not include warnings.

**Key Assertions:**

- The length of the 'warnings' list in the report should be exactly 1.
- The code in the first warning message should match 'W001'.
- All warnings in the report should have a matching code and message.

COVERAGE

| | |
|---|---|
| `src/pytest_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` | 60 lines (ranges: 229-231, 233, 235, 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510-512, 514, 516, 518, 520, 522) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

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:** This test prevents a regression where the sorting of tests based on nodeid is not maintained correctly.

**Key Assertions:**

- The list of nodeids returned from `to_dict()` matches the expected order.
- The nodeids are present in the original dictionary with their corresponding values.
- The nodeids are sorted in ascending order (a before z).
- No duplicate nodeids are present in the result.
- The test is only successful if all tests have a unique nodeid.
- No test has an id that starts with 'z' or 'm'.
- The nodeids are not empty.

COVERAGE

| | |
|---|---|
| `src/pytest_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` | 71 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestReportWarning::test_to_dict_with_detail` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that the `to_dict()` method of `ReportWarning` returns a dictionary with the 'detail' key.

**Why Needed:** This test prevents a potential issue where the detailed warning message is not included in the dictionary returned by `to_dict()`.

**Key Assertions:**

- The value of the 'detail' key in the dictionary should be '/path/to/file'.
- The value of the 'detail' key should be present in the dictionary.
- The detailed warning message should be included in the dictionary returned by `to_dict()`.

COVERAGE

| | |
|---|---|
| `src/pytest_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: 229-231, 233-235)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_models.py::TestReportWarning::test_to_dict_without_detail` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test to dictionary without detail should exclude it.

**Why Needed:** This test prevents a warning about excluding detailed warnings from the report.

**Key Assertions:**

- The 'detail' key is expected to be missing from the warning dictionary.
- The 'message' key is expected to match the original message of the warning.
- The 'code' key is expected to match the original code of the warning.

COVERAGE

| | |
|---|---|
| `src/pytest_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: 229-231, 233, 235)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_models.py::TestRunMeta::test_aggregation_fields_present` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that RunMeta has aggregation fields.

**Why Needed:** Prevents regression where RunMeta is missing aggregation fields, potentially causing incorrect run counts or aggregated report data.

**Key Assertions:**

- The 'run_id' key should be present in the output dictionary with value 'run-123'.
- The 'run_group_id' key should be present in the output dictionary with value 'group-456'.
- The 'is_aggregated' key should be True.
- The 'aggregation_policy' key should be 'merge'.
- The 'run_count' key should be 3.
- The length of the 'source_reports' list should be 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/models.py` | 39 lines (ranges: 277-279, 281-283, 364-380, 382, 385, 387, 390, 393, 395, 397, 399-405, 407, 419) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestRunMeta::test_llm_fields_excluded_when_disabled` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that LLM fields are excluded when annotations are disabled.

**Why Needed:** This test prevents a regression where the LLM model's fields are included even when annotations are not enabled.

**Key Assertions:**

- The 'llm_annotations_enabled' key is present in the data.
- The 'llm_provider' key is not present in the data.
- The 'llm_model' key is not present in the data.

COVERAGE

| | |
|---|---|
| src/pytest_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: 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestRunMeta::test_llm_traceability_fields` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test LLM traceability fields are included when enabled.

**Why Needed:** This test prevents regression where the llm_traceability_fields parameter is not present or has incorrect values.

**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

COVERAGE

| | |
|---|---|
| src/pytest_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: 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407-419) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestRunMeta::test_non_aggregated_excludes_source_reports` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'Non-aggregated report should not include source_reports' verifies that a non-aggregated run does not include source reports.

**Why Needed:** This test prevents regression where the 'source_reports' key is included in the aggregated report.

**Key Assertions:**

- The 'source_reports' key is present in the dictionary.
- The value of 'is_aggregated' is False.
- The 'source_reports' key is not included in the output.

COVERAGE

| | |
|---|---|
| `src/pytest_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: 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestRunMeta::test_run_meta_to_dict_full` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test RunMeta to dict with all optional fields.

**Why Needed:** Prevents regression in case of missing or invalid metadata.

**Key Assertions:**

- The 'git_sha' field should be present and have the correct value.
- The 'git_dirty' field should be True.
- The 'repo_version' field should be present and have the correct value.
- The 'repo_git_sha' field should be present and have the correct value.
- The 'repo_git_dirty' field should be False.
- The 'plugin_git_sha' field should be present and have the correct value.
- The 'plugin_git_dirty' field should be False.
- The 'config_hash' field should be present and have the correct value.
- The length of the 'source_reports' list should be 1.

COVERAGE

| | |
|---|---|
| src/pytest_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: 277-279, 281-283, 364-380, 382-405, 407, 419) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestRunMeta::test_run_status_fields` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** RunMeta should include run status fields.

**Why Needed:** This test prevents a potential bug where the RunMeta object is missing certain critical fields, potentially leading to incorrect analysis results or errors in downstream code.

**Key Assertions:**

- The 'exit_code' field is set to 1.
- The 'interrupted' field is True.
- The 'collect_only' field is True.
- The 'collected_count' field should be equal to 10.
- The 'selected_count' field should be equal to 8.
- The 'deselected_count' field should be equal to 2.

COVERAGE

| | |
|---|---|
| src/pytest_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: 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestSchemaVersion::test_schema_version_format` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Verifies the schema version is formatted as a semver string.

**Why Needed:** Prevents regression where the schema version is not correctly parsed into a semver format.

**Key Assertions:**

- The `SCHEMA_VERSION` variable should be split into three parts using '.' as the separator.
- Each part of the `SCHEMA_VERSION` variable should consist only of digits.
- The length of each part should be exactly 3 characters (i.e., 'x.x.x').
- All parts should have a non-zero value (i.e., not all zeros).

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED tests/test_models.py::TestSchemaVersion::test_schema_version_in_report_root 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that the `ReportRoot` class includes the schema version in its report root.

**Why Needed:** This test prevents a potential bug where the schema version is not included in the report root, potentially causing issues with downstream processing or reporting.

**Key Assertions:**

- The `schema_version` attribute of the `ReportRoot` object should be equal to `SCHEMA_VERSION`.
- The `to_dict()` method of the `ReportRoot` class should return a dictionary with a key named `schema_version` and a value equal to `SCHEMA_VERSION`.
- The `schema_version` field in the report root dictionary should have the same value as `SCHEMA_VERSION`.
- The schema version is present in the report root, regardless of any potential issues with formatting or data.
- The test passes if the `ReportRoot` class implements a proper `__repr__()` method that returns a string containing the schema version.

COVERAGE

| | |
|---|---|
| src/pytest_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: 364-380, 382, 385, 387, 390, 393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestSourceCoverageEntry::test_to_dict` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test coverage entry serialization.

**Why Needed:** This test prevents a bug where the `CoverageEntry` object is not properly serialized to JSON.

**Key Assertions:**

- The 'file_path' key should match the expected value.
- The 'line_ranges' key should match the expected value.
- The 'line_count' key should match the expected value.
- The 'to_dict()' method of the `CoverageEntry` object should return a dictionary with all required keys.
- The values in the returned dictionary should be strings or integers.
- The string representation of the `CoverageEntry` object should not contain any non-ASCII characters.
- The JSON representation of the `CoverageEntry` object should match the expected output.
- The 'file_path' key should be present in the dictionary even if it's empty.

COVERAGE

| | |
|---|---|
| src/pytest_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: 71-78) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestSourceReport::test_to_dict_minimal` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that the `to_dict` method of `LlmAnnotation` returns a dictionary with required fields.

**Why Needed:** This test prevents a potential bug where the minimal annotation is missing some required fields in its serialized representation.

**Key Assertions:**

- The presence of 'scenario' in the dictionary is expected.
- The presence of 'why_needed' in the dictionary is expected.
- The presence of 'key_assertions' in the dictionary is expected.
- The absence of 'confidence' in the dictionary is expected (it's optional and can be None).

COVERAGE

| | |
|---|---|
| src/pytest_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: 277-279, 281, 283) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestSourceReport::test_to_dict_with_run_id` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test SourceReport to_dict_with_run_id function with a SourceReport object.

**Why Needed:** This test prevents the loss of run_id in the output dictionary when converting SourceReport to a dictionary.

**Key Assertions:**

- The 'run_id' key should be present in the output dictionary.
- The value of 'run_id' should match the provided run_id.
- If no run_id is provided, the 'run_id' key should still be present but its value should be an empty string or None.
- If a SourceReport object does not have a run_id attribute, the test should fail with an assertion error.
- The output dictionary should contain only one key-value pair for the SourceReport object.
- The 'run_id' key should be included in the output dictionary even if it is empty or null.
- If the source report has multiple attributes, the 'run_id' key should not be included in the output dictionary.

COVERAGE

| | |
|---|---|
| `src/pytest_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: 277-279, 281-283)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_models.py::TestSummary::test_to_dict` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that `CoverageEntry.to_dict()` correctly serializes the test summary.

**Why Needed:** This test prevents a potential bug where the serialized test summary is incorrect or missing critical information.

**Key Assertions:**

- The 'file_path' key in the dictionary should match the expected value.
- The 'line_ranges' key in the dictionary should contain the correct ranges and count.
- The 'line_count' key in the dictionary should have the same value as the original test entry.
- All other keys in the dictionary should be present and have the correct values.

COVERAGE

| | |
|---|---|
| `src/pytest_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: 449-457, 459, 461)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_models.py::TestTestCaseResult::test_minimal_result` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that a minimal result has the required fields.

**Why Needed:** This test prevents regression where a minimal result is missing some necessary information.

**Key Assertions:**

- The 'nodeid' field should match the expected value.
- The 'outcome' field should be set to 'passed'.
- The 'duration' field should be set to 0.0 (or any other default value).
- The 'phase' field should be set to 'call'.

COVERAGE

| | |
|---|---|
| `src/pytest_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` | 17 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestTestCaseResult::test_result_with_coverage` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test verifies that the `result` dictionary contains a single 'coverage' key.

**Why Needed:** This test prevents regression in coverage reporting when using the `result` object.

**Key Assertions:**

- The 'coverage' list should contain exactly one entry.
- The first element of the 'coverage' list should have a 'file_path' attribute equal to 'src/foo.py'.
- The 'coverage' list should not be empty.
- All elements in the 'coverage' list should have a 'line_ranges' attribute and a 'line_count' attribute.
- Each element in the 'coverage' list should have a 'file_path' attribute that matches the expected value.
- Each element in the 'coverage' list should have exactly one 'line_ranges' attribute with values ranging from 1 to 5 inclusive.
- The 'line_ranges' attribute should not be empty.
- All elements in the 'coverage' list should have a valid 'line_count' attribute.

COVERAGE

| | |
|---|---|
| `src/pytest_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` | 22 lines (ranges: 40-43, 161-165, 167, 169, 171, 173, 176-178, 180, 182, 184, 186, 188, 190) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestTestCaseResult::test_result_with_llm_opt_out` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test case "Result with LLM opt-out" verifies that the `TestCaseResult` object includes a flag indicating whether LLM optimization was opted out.

**Why Needed:** This test prevents regression where the `TestCaseResult` object does not include this flag when LLM optimization is explicitly set to `False`.

**Key Assertions:**

- The value of `llm_opt_out` in the `result.to_dict()` output should be `True`.
- The key `llm_opt_out` exists in the dictionary representation of the `result` object.
- The value of `llm_opt_out` is a boolean value (`True` or `False`).
- The `TestCaseResult` object includes this flag when LLM optimization is explicitly set to `False`.
- When LLM optimization is not opted out, the `llm_opt_out` flag should be present in the output.
- The presence of this flag prevents regression where the `TestCaseResult` object does not include it.
- This test ensures that the correct behavior is observed when LLM optimization is disabled.

COVERAGE

| | |
|---|---|
| src/pytest_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 | 18 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180-182, 184, 186, 188, 190) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestTestCaseResult::test_result_with_rerun` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'Result with reruns' verifies that the 'rerun_count' and 'final_outcome' fields are correctly populated in a TestCaseResult object.

**Why Needed:** This test prevents regression where the 'rerun_count' field is not updated when re-running the test.

**Key Assertions:**

- The 'rerun_count' field should be equal to the expected value of 2.
- The 'final_outcome' field should be equal to 'passed'.
- The 'rerun_count' field should match the number of reruns performed.
- The final outcome should always be 'passed'.
- Reruns should update the 'rerun_count' field correctly.
- Reruns should not affect the 'final_outcome' field.

COVERAGE

| | |
|---|---|
| `src/pytest_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: 161-165, 167, 169, 171, 173-176, 178, 180, 182, 184, 186, 188, 190) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_models.py::TestTestCaseResult::test_result_without_rerun_excludes_fields` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'result_without_rerun_excludes_fields' verifies that the 'TestCaseResult' object does not include 'rerun_count' and 'final_outcome' fields.

**Why Needed:** This test prevents a regression where the 'TestCaseResult' object includes these fields when result is rerun.

**Key Assertions:**

- The 'result_without_rerun_excludes_fields' method should return an instance of 'TestCaseResult' with no 'rerun_count' and 'final_outcome' fields.
- The 'result_without_rerun_excludes_fields' method should not include 'rerun_count' in the dictionary representation of the object.
- The 'result_without_rerun_excludes_fields' method should not include 'final_outcome' in the dictionary representation of the object.
- The 'result_without_rerun_excludes_fields' method should return an instance with no 'rerun_count' and 'final_outcome' fields when result is rerun.
- The 'result_without_rerun_excludes_fields' method should not include 'rerun_count' in the dictionary representation of the object even if result is rerun.
- The 'result_without_rerun_excludes_fields' method should not include 'final_outcome' in the dictionary representation of the object even if result is rerun.
- The 'result_without_rerun_excludes_fields' method should raise an AssertionError when result is rerun and includes 'rerun_count' or 'final_outcome' fields.
- The 'result_without_rerun_excludes_fields' method should not include 'rerun_count' in the dictionary representation of the object even if result is rerun.

COVERAGE

| | |
|---|---|
| src/pytest_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 | 17 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestConfig::test_default_values` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Tests default configuration values.

**Why Needed:** Prevents regression in default settings when testing LLMs without a provider.

**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'

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147, 224)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_options.py::TestConfig::test_get_default_config` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the default configuration is correctly initialized.

**Why Needed:** Prevents a potential bug where the default configuration is not properly set to 'none'.

**Key Assertions:**

- The `cfg` variable should be an instance of `Config`.
- The `cfg.provider` attribute should be set to `'none'`.
- The `cfg` object should have a `provider` attribute that matches the expected value.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147, 233) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_options.py::TestConfig::test_is_llm_enabled 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that the `is_llm_enabled` check returns False for a provider without an LLM.

**Why Needed:** Prevents regression in case the LLM is not enabled by default.

**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'`.
- The function `Config.is_llm_enabled()` should not return a value when the provider is set to an unknown or default value.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestConfig::test_validate_invalid_aggregate_policy` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Testing the `test_validate_invalid_aggregate_policy` test function.

**Why Needed:** This test prevents a potential bug where an invalid aggregation policy is used, causing the configuration to fail validation and potentially leading to unexpected behavior or errors.

**Key Assertions:**

- The configuration object `cfg` has a `validate()` method that returns a list of error messages.
- At least one error message in the list indicates that the `aggregate_policy` field is invalid ('random').
- The error message explicitly states that the aggregate policy 'random' is invalid.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 20 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-197, 201-202, 209, 211, 213, 215, 217, 220) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestConfig::test_validate_invalid_context_mode` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Testing the `validate()` method with an invalid context mode.

**Why Needed:** This test prevents a potential bug where the validation of an invalid context mode fails and returns incorrect error messages.

**Key Assertions:**

- The `validate()` method should return exactly one error message for an invalid context mode.
- The error message should contain 'Invalid llm_context_mode' as its prefix.
- The error message should not be empty or null.
- The error message should include the exact string 'Invalid llm_context_mode' in its text.
- The error message should not include any other strings that are not part of the 'Invalid llm_context_mode' phrase.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 20 lines (ranges: 107, 147, 175, 178-179, 185-189, 193-194, 201-202, 209, 211, 213, 215, 217, 220) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestConfig::test_validate_invalid_include_phase` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Tests the test_validate_invalid_include_phase function to ensure it correctly validates an invalid include phase.

**Why Needed:** This test prevents a potential bug where an invalid include phase is not properly validated, potentially leading to incorrect configuration or unexpected behavior in downstream code.

**Key Assertions:**

- The 'include_phase' parameter passed to the Config class should be one of 'background', 'main', or 'pre_game'.
- An error message indicating that the specified include phase is invalid should be included in any validation errors returned by the validate() method.
- When an invalid include phase is provided, a single error message should be reported with the specific invalid value ('lunch_break').

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 20 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-205, 209, 211, 213, 215, 217, 220) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestConfig::test_validate_invalid_provider` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test validation of an invalid provider.

**Why Needed:** Prevents a potential bug where the test fails with an error message indicating an invalid provider.

**Key Assertions:**

- The function `validate()` should return exactly one error message.
- The error message should contain the string 'Invalid provider ' followed by the actual invalid provider name.
- The error message should not be empty.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 19 lines (ranges: 107, 147, 175, 178-181, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestConfig::test_validate_numeric_ranges` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test validation of numeric constraints for TestConfig.

**Why Needed:** Prevents regression where the default values are not validated correctly.

**Key Assertions:**

- The configuration is valid if llm_context_bytes is greater than or equal to 1000.
- llm_max_tests should be set to a positive value (e.g., 1) for no limit.
- llm_requests_per_minute should be at least 1 for a reasonable number of requests per minute.
- llm_timeout_seconds should be at least 1 for a reasonable timeout in seconds.
- llm_max_retries should be set to 0 or positive for no retries.
- The configuration is invalid if llm_context_bytes is less than 1000.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 22 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209-218, 220) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestConfig::test_validate_valid_config` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Testing the `validate()` method with a valid configuration.

**Why Needed:** This test prevents a potential bug where an invalid configuration is passed to the validation process, potentially causing unexpected behavior or errors.

**Key Assertions:**

- A Config object is created and initialized.
- The `validate()` method is called on the Config object.
- An empty list (`[]`) is returned from the `validate()` method.
- No error messages are printed to the console.
- The configuration is successfully validated without any issues.
- The `errors` attribute of the Config object is set to an empty list.
- A valid configuration is used to test the validation functionality.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 17 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestLoadConfig::test_load_aggregation_options` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test loads aggregation options with correct aggregate directory, policy and run ID.

**Why Needed:** This test prevents regression where the aggregate options are not loaded correctly due to incorrect or missing values.

**Key Assertions:**

- The aggregate directory is set to 'aggr_dir'.
- The aggregate policy is set to 'merge'.
- The aggregate run ID is set to 'run-123'.
- The aggregate group ID is set to 'group-abc'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147, 248, 251, 253, 255, 257, 259, 261, 263, 270, 272, 274, 276, 278, 280, 282, 286-294, 298, 300) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestLoadConfig::test_load_config_invalid_int_ini` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'test_load_config_invalid_int_ini' verifies that the test handles invalid integer values in INI correctly.

**Why Needed:** This test prevents a potential regression where the test crashes due to an invalid integer value in the INI file.

**Key Assertions:**

- The function `load_config(mock_pytest_config)` should return the expected configuration with the default value of 10 for 'llm_max_retries'.
- The function `getini` should not crash when called with an invalid integer value in the INI file.
- The test should be able to handle different types of invalid integer values (e.g. negative, zero) without crashing or returning unexpected results.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147, 248, 251, 253, 255, 257, 259, 261, 263-267, 270, 272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestLoadConfig::test_load_coverage_source` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that the `llm_coverage_source` option is set to 'cov_dir' after loading the configuration.

**Why Needed:** This test prevents a bug where the coverage source is not correctly set when using the `--coverage` flag with PyTorch Lightning.

**Key Assertions:**

- The value of `cfg.llm_coverage_source` should be 'cov_dir'.
- The `llm_coverage_source` option in the configuration file should match the value set by the test.
- The coverage source path should be correctly resolved to 'cov_dir' when loading the configuration.
- The PyTorch Lightning configuration should update correctly with the new coverage source setting.
- The `--coverage` flag should not cause any issues when using PyTorch Lightning with this configuration.
- The test should fail if the coverage source is not set to 'cov_dir' after loading the configuration.
- The `llm_coverage_source` option in the configuration file should be correctly updated after setting it to 'cov_dir'.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/options.py` | 25 lines (ranges: 107, 147, 248, 251, 253, 255, 257, 259, 261, 263, 270, 272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294-295, 298, 300) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestLoadConfig::test_load_defaults` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the `load_defaults` test loads a default configuration without any options.

**Why Needed:** Prevents regression when no options are set, ensuring the test behaves as expected.

**Key Assertions:**

- cfg.provider should be set to 'none' in this case.
- cfg.report_html should be None.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/options.py` | `24 lines (ranges: 107, 147, 248, 251, 253, 255, 257, 259, 261, 263, 270, 272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_options.py::TestLoadConfig::test_load_from_cli_overrides_ini` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that CLI options override ini options.

**Why Needed:** This test prevents a bug where the CLI overrides ini settings, potentially causing unexpected behavior or incorrect results.

**Key Assertions:**

- ini_value is set to 'cli_report.html' instead of 'ini_value'
- llm_requests_per_minute is set to 100 instead of None
- llm_request_per_minute is not set in ini

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147, 248, 251, 253, 255, 257, 259-261, 263, 270-272, 274, 276, 278, 280-282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options.py::TestLoadConfig::test_load_from_cli_retries` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Testing the `load_from_cli` option to ensure it correctly sets the maximum retries.

**Why Needed:** This test prevents a potential bug where the `llm_max_retries` option is not set correctly, leading to incorrect configuration.

**Key Assertions:**

- The value of `llm_max_retries` in the loaded configuration should be equal to 9.
- The `load_from_cli` option should have been able to find a valid configuration with retries set to 9.
- The `llm_max_retries` option should not be set to a negative value (0) or an invalid value (e.g., less than 1).
- No error message should be raised when the `load_from_cli` option is called without setting the `llm_max_retries` option.
- The configuration loaded from the CLI with retries set to 9 should have the same structure as the default configuration.
- The `load_from_cli` option should not throw an exception if the specified number of retries is reached.
- The `load_from_cli` option should be able to handle cases where the specified number of retries is greater than the configured maximum.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/options.py` | `25 lines (ranges: 107, 147, 248, 251, 253, 255, 257, 259, 261, 263, 270, 272, 274, 276, 278, 280, 282-283, 286, 288, 290, 292, 294, 298, 300)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_options.py::TestLoadConfig::test_load_from_ini` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test loading values from ini options.

**Why Needed:** Prevents a potential bug where the `load_config` function does not properly initialize the `provider`, `model`, and other configuration variables with default values.

**Key Assertions:**

- The `provider` attribute of the loaded configuration should be set to 'ollama'.
- The `model` attribute of the loaded configuration should be set to 'llama3'.
- The `llm_context_mode` attribute of the loaded configuration should be set to 'balanced'.
- The `llm_requests_per_minute` attribute of the loaded configuration should be set to 10.
- The `llm_max_retries` attribute of the loaded configuration should be set to 5.
- The `report_html` attribute of the loaded configuration should be set to 'report.html'.
- The `report_json` attribute of the loaded configuration should be set to 'report.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` | 32 lines (ranges: 107, 147, 248, 251-265, 270, 272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_aggregation_settings` 1ms 🛡 3

AI ASSESSMENT

**Scenario:** Test Config with aggregation settings.

**Why Needed:** Prevents regression in aggregation settings configuration.

**Key Assertions:**

- The `aggregate_dir` attribute should be set to `/reports`.
- The `aggregate_policy` attribute should be set to 'merge'.
- The `aggregate_include_history` attribute should be True.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_all_output_paths` 1ms 🛡 3

AI ASSESSMENT

**Scenario:** Test Config with all output paths.

**Why Needed:** Prevents a potential bug where the report is not generated for all possible output files.

**Key Assertions:**

- The `report_html` attribute of the test configuration should match 'report.html'.
- The `report_json` attribute of the test configuration should match 'report.json'.
- The `report_pdf` attribute of the test configuration should match 'report.pdf'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_capture_settings` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the configuration of capturing failed output and setting a maximum number of characters for capturing output.

**Why Needed:** This test prevents a bug where the test fails due to an incorrect capture settings, causing it to fail unexpectedly.

**Key Assertions:**

- config.capture_failed_output is True
- config.capture_output_max_chars = 8000

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_compliance_settings` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the configuration of compliance settings.

**Why Needed:** This test prevents a potential bug where the configuration file is not correctly set to 'metadata.json'.

**Key Assertions:**

- The `metadata_file` attribute of the `Config` object is set to 'metadata.json'.
- The `hmac_key_file` attribute of the `Config` object is set to 'key.txt'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_coverage_settings` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the configuration of coverage settings.

**Why Needed:** Prevents a potential bug where coverage settings are not correctly applied.

**Key Assertions:**

- config.omit_tests_from_coverage is set to False (expected)
- config.include_phase is set to "all" (expected)
- The omit_tests_from_coverage parameter has the correct value (False) and includes phase "all"

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_custom_exclude_globs` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the ability to include custom exclude globs in the LLM context configuration.

**Why Needed:** This test prevents a potential bug where the default exclude globs are not properly propagated to the LLM context.

**Key Assertions:**

- The '*.pyc' glob should be included in the list of excluded files.
- The '*.log' glob should be included in the list of excluded files.
- The custom exclude globs provided through the Config object should be properly propagated to the LLM context configuration.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 147)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_include_globs` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the `llm_context_include_globs` configuration option includes only `.py` files.

**Why Needed:** Prevents a potential bug where the include globs are not correctly filtered to only include `.py` files.

**Key Assertions:**

- The `*.py` glob matches any file with a `.py` extension.
- The `*.pyi` glob matches any file with an `.pyi` extension.
- The `llm_context_include_globs` configuration option is set to include only these globs.
- The `include_globs` attribute of the `Config` object contains the expected list of globs.
- The `llm_context_include_globs` attribute does not contain any `.pyi` files.
- The `*.py` glob matches a file that is also a `.pyi` file (e.g., `.pyi.py`).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_invocation_settings` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the `include_pytest_invocation` attribute of `Config` object is set to `False` when `pytest_invocation` is not specified.

**Why Needed:** This test prevents a potential bug where the `include_pytest_invocation` attribute of `Config` object is incorrectly set to `True` when `pytest_invocation` is `None`.

**Key Assertions:**

- config.include_pytest_invocation is False
- config.include_pytest_invocation is not True if pytest_invocation is None
- pytest_invocation is None or config.include_pytest_invocation is True

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_llm_execution_settings` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the LLM execution settings are correctly configured.

**Why Needed:** This test prevents a bug where the maximum number of tests is not set to 50, potentially leading to performance issues or unexpected behavior.

**Key Assertions:**

- The value of llm_max_tests is equal to 50.
- The value of llm_max_concurrency is equal to 8.
- The value of llm_requests_per_minute is equal to 12.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_llm_param_settings` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the configuration of LLM parameter settings.

**Why Needed:** Prevent regression in LLM parameter setting configuration.

**Key Assertions:**

- config.llm_include_param_values is True
- assert config.llm_param_value_max_chars == 200

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_llm_settings` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Tests the configuration of LLM settings.

**Why Needed:** Prevents a potential bug where the model is not properly set to 'llama3.2'.

**Key Assertions:**

- assert config.provider == "ollama",
- assert config.model == "llama3.2",
- assert config.llm_context_bytes == 64000

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_repo_root_path` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the `repo_root` attribute is correctly set to `/project` for a given configuration.

**Why Needed:** This test prevents potential issues where the repository root path is not set correctly, potentially leading to incorrect behavior or errors in subsequent tests.

**Key Assertions:**

- The `repo_root` attribute of the `Config` object is set to `Path('/project')`.
- The `repo_root` attribute of the `Config` object is equal to `Path('/project')` (case-insensitive).
- The directory path `/project` exists and is a valid directory.
- The file name `.git` does not exist in the repository root `/project/.git`.
- The file name `README.md` exists in the repository root `/project/README.md`.
- The file name `LICENSE` exists in the repository root `/project/LICENSE`.
- The directory path `/project/.git` exists and is a valid Git repository.
- The file name `.env` does not exist in the repository root `/project/.env`.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_extended.py::TestConfigAnnotations::test_valid_phase_values` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verifies that all valid include_phase values pass validation without raising any errors.

**Why Needed:** Prevents a potential bug where invalid or missing include_phase values are passed to the Config class, potentially causing runtime errors or data corruption.

**Key Assertions:**

- The validate() method of the Config object should not return any error messages for valid include_phase values.
- Any error message containing 'include_phase' should be ignored and not propagated to the caller.
- All included phases (run, setup, teardown, all) should be successfully validated without raising any errors.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/options.py` | 17 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_maximal.py::TestConfigDefaultsMaximal::test_default_exclude_globs` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verifies that the default exclude globs are correctly set to include only Python c files, __pycache__, and secret files.

**Why Needed:** This test prevents a potential bug where the default exclude globs do not include all necessary files for the LLM context.

**Key Assertions:**

- The function `Config().llm_context_exclude_globs` returns a list of strings that includes `*.pyc`, `__pycache__/*`, and `*secret*`.
- The function `Config().llm_context_exclude_globs` returns a list of strings that includes `*_password_* files.
- The function `Config().llm_context_exclude_globs` returns a list of strings that includes all necessary files for the LLM context, including Python c files, __pycache__, and secret files.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_maximal.py::TestConfigDefaultsMaximal::test_default_redact_patterns` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the default redact patterns configuration.

**Why Needed:** Prevents a potential bug where sensitive information like passwords and tokens are not properly redacted.

**Key Assertions:**

- The `--password` pattern should match any occurrences of `--password` in the provided patterns.
- The `--token` pattern should match any occurrences of `--token` in the provided patterns.
- The `--api[_-]?key` pattern should match any occurrences of `--api[_-]?key` in the provided patterns.
- All sensitive information like passwords and tokens should be redacted according to these default patterns.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_options_maximal.py::TestConfigDefaultsMaximal::test_default_values 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the default values of the configuration options.

**Why Needed:** This test prevents a potential regression where the default values are not correctly set for the configuration.

**Key Assertions:**

- The `provider` attribute should be set to `'none'`.
- The `llm_context_mode` attribute should be set to `'minimal'`.
- The `llm_context_bytes` attribute should be set to `32000`.
- The `omit_tests_from_coverage` attribute should be set to `True`.
- The `include_phase` attribute should be set to `'run'`.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147, 233) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_maximal.py::TestConfigHelpersMaximal::test_is_llm_enabled` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verifies that the `is_llm_enabled` method returns False for providers without a specified provider name.

**Why Needed:** Prevents regression where the LLM is not enabled by default (e.g., when no provider is specified).

**Key Assertions:**

- Config(provider='none').is_llm_enabled() should return False.
- Config(provider='ollama').is_llm_enabled() should return True.
- Config(provider='litellm').is_llm_enabled() should return True.
- Config(provider='gemini').is_llm_enabled() should return True.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147, 224)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_options_maximal.py::TestConfigValidationMaximal::test_validate_invalid_aggregate_policy` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the validation of an invalid aggregate policy.

**Why Needed:** To prevent a potential bug where an invalid aggregate policy is passed to the Config class, causing it to throw an error without providing any meaningful information about the issue.

**Key Assertions:**

- The validate method returns exactly one error message.
- The error message contains the string 'Invalid aggregate_policy 'invalid''.
- The error message includes the specified invalid aggregate policy as a substring.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/options.py` | 20 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-197, 201-202, 209, 211, 213, 215, 217, 220) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_maximal.py::TestConfigValidationMaximal::test_validate_invalid_context_mode` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the `validate()` method of the `Config` class when an invalid context mode is provided.

**Why Needed:** This test prevents a potential bug where the `validate()` method returns incorrect error messages for invalid context modes.

**Key Assertions:**

- The `validate()` method should return exactly one error message with the specified error message.
- The error message should contain 'Invalid llm_context_mode ' followed by the actual value of the invalid mode.
- The test should fail when an invalid context mode is provided.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/options.py` | `20 lines (ranges: 107, 147, 175, 178-179, 185-189, 193-194, 201-202, 209, 211, 213, 215, 217, 220)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |

PASSED `tests/test_options_maximal.py::TestConfigValidationMaximal::test_validate_invalid_include_phase` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test validates an invalid include phase.

**Why Needed:** Prevents a potential bug where the test incorrectly accepts an invalid include phase.

**Key Assertions:**

- The config object should have exactly one error message.
- The error message should contain 'Invalid include_phase' as its key.
- The error message should be present in the first error item of the list.
- The error message should not be empty.
- The error message should not be a string.
- The config object should raise an exception with the provided error message when validate() is called.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/options.py` | 20 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-205, 209, 211, 213, 215, 217, 220) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_maximal.py::TestConfigValidationMaximal::test_validate_invalid_provider` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test validates an invalid provider.

**Why Needed:** Prevents a potential bug where the test fails due to an invalid provider being used.

**Key Assertions:**

- The function `validate()` should return exactly one error message.
- The error message should contain the string 'Invalid provider ' + provider_name'.
- The error message should not be empty.
- The error message should include the exact word 'invalid'.
- The error message should not contain any other strings besides 'Invalid provider '.
- The error message should not be a simple string but rather an actual error message from the provider.
- The error message should indicate that the provider is invalid.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/options.py` | 19 lines (ranges: 107, 147, 175, 178-181, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_maximal.py::TestConfigValidationMaximal::test_validate_numeric_bounds` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** tests/test_options_maximal.py::TestConfigValidationMaximal::test_validate_numeric_bounds

**Why Needed:** Prevents a potential bug where the config is not validated correctly due to invalid numeric values.

**Key Assertions:**

- The config should contain errors for invalid numeric values.
- The config should have errors with llm_context_bytes, llm_max_tests, llm_requests_per_minute, and llm_timeout_seconds keys.
- Each error message should indicate the specific key that is out of bounds.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 21 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209-217, 220) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_options_maximal.py::TestConfigValidationMaximal::test_validate_valid_config` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verifies that an empty configuration object is returned when a valid configuration is provided.

**Why Needed:** Prevents potential infinite recursion in the validation process by returning an empty list.

**Key Assertions:**

- The `validate()` method of the `Config` class should return an empty list for a valid configuration.
- An empty list should be returned when the input configuration is valid.
- The validation process should not lead to infinite recursion.
- The `validate()` method should only attempt to validate the configuration and return an error message if necessary.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 17 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_plugin_integration.py::TestPluginConfigLoading::test_config_defaults` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verify that the configuration has default settings.

**Why Needed:** Prevent a potential bug where the plugin defaults to an insecure configuration.

**Key Assertions:**

- The function `load_config(pytestconfig)` returns a `Config` object.
- The `isinstance(cfg, Config)` assertion checks if the returned value is indeed of type `Config`.
- If the `pytestconfig` has no registered options, this test will fail and raise an AssertionError.
- Without registering options in `pytestconfig`, the plugin defaults to a configuration that may not be secure by default.
- The `cfg` variable should hold a valid `Config` object with safe defaults.
- If the `cfg` is not of type `Config`, the test will fail and raise an AssertionError.
- The `cfg` object's attributes (e.g., `options`) should have their default values set correctly.
- Without setting default options in `pytestconfig`, the plugin may behave unexpectedly or insecurely.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 33 lines (ranges: 107, 147, 248, 251-259, 261, 263-265, 270, 272-276, 278, 280, 282, 286, 288, 290-292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_plugin_integration.py::TestPluginConfigLoading::test_markers_exist_in_config` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Verify that the `pytestconfig` object is accessible.

**Why Needed:** Prevent a bug where the plugin configuration is inaccessible due to incorrect import or setup.

**Key Assertions:**

- The `pytestconfig` object should be assigned a value from the `pytestconfig` fixture.
- The `pytestconfig` object should not be `None` when accessed.
- The `pytestconfig` object should have attributes that are accessible (e.g. `markers`, `plugins`, etc.).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_plugin_integration.py::TestPluginIntegration::test_llm_context_marker` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** test_llm_context_marker verifies that a context marker does not cause errors in the LLM integration.

**Why Needed:** This test prevents regression and ensures that the LLM integration works as expected without causing any errors due to context markers.

**Key Assertions:**

- The function `test_llm_context_marker` should assert True, indicating no error or exception occurred.
- The context marker should not be present in the output of the test.
- Any exceptions raised during the execution of the test should be captured and reported as errors.
- The LLM integration should work correctly without any issues caused by the presence of a context marker.
- The plugin configuration should be able to handle the presence of a context marker without causing any errors.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_plugin_integration.py::TestPluginIntegration::test_llm_opt_out_marker` 1ms 🛡️ 2

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED tests/test_plugin_integration.py::TestPluginIntegration::test_requirement_marker 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** The test verifies that the requirement marker does not cause any errors.

**Why Needed:** This test prevents a potential bug where the requirement marker could be misinterpreted or cause a runtime error.

**Key Assertions:**

- The function `self.requirement_marker()` should return None if no requirements are present.
- The function `self.requirement_marker()` should raise an exception with a meaningful message if a requirements string is provided.
- The function `self.requirement_marker()` should not throw any exceptions when called without arguments.
- The function `self.requirement_marker()` should not throw any exceptions when called with a single argument (requirements string).
- The function `self.requirement_marker()` should return None if the requirements string is empty.
- The function `self.requirement_marker()` should raise an exception with a meaningful message if the requirements string contains invalid characters.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_plugin_integration.py::TestReportGeneration::test_report_writer_integration` 33ms 🛡️ 6

AI ASSESSMENT

**Scenario:** The test verifies that the report writer correctly generates a full report with both JSON and HTML output.

**Why Needed:** This test prevents regression where the report writer fails to generate reports for tests with different error messages or durations.

**Key Assertions:**

- Verify that the 'report.json' file exists in the specified path.
- Assert that the total number of tests is correct (2 in this case).
- Check if the 'passed' count matches the expected value (1 in this case).
- Verify that the HTML output includes both test files ('test_a.py' and 'test_b.py').

COVERAGE

| | |
|---|---|
| src/pytest_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 | 79 lines (ranges: 161-165, 167-169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 364-380, 382-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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 | 131 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, 312, 314-315, 317-320, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 376, 378-379, 382, 385, 388, 391-395, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_disabled` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test that collectreport skips when disabled and pytest_collectreport is mocked correctly.

**Why Needed:** This test prevents a regression where collectreport fails to run due to an unconfigured stash.

**Key Assertions:**

- The `pytest_collectreport` function should be called with `_enabled_key` as the key and `False` as the value when `collectreport` is disabled.
- The `get` method of `session.config.stash` should return a mock object that returns `None` when called with `_enabled_key` as the argument and `False` as the value.
- The `assert_called_with` method of the mocked `get` method should be called with `_enabled_key` as the first argument and `False` as the second argument.
- The `session.config.stash.get` method should not have been called when `collectreport` is disabled.
- The `pytest_collectreport` function should not have been called when `collectreport` is disabled.
- The `mock_report.session.config.stash.get` method should return a mock object that returns `None` when called with `_enabled_key` as the argument and `False` as the value.
- The `pytest_collectreport` function should be mocked to return a mock object that behaves like an unconfigured stash.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 380-381, 384, 388-390, 401-402, 408-409) |

PASSED `tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_enabled` 2ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test that collectreport calls collector when enablement is enabled.

**Why Needed:** This test prevents a potential regression where the plugin does not call the collector when collectreport is enabled.

**Key Assertions:**

- The `pytest_collectreport` function should be called with `mock_collector` as its argument.
- The `handle_collection_report` method of `mock_collector` should have been called once with `mock_report` as its argument.
- The `stash_get` function of `mock_report.session.config.stash` should have returned `True` for the `_enabled_key` and `_collector_key` keys.
- The `handle_collection_report` method of `mock_collector` should not have been called if `mock_report` was not a valid report object.
- The `stash_get` function of `mock_report.session.config.stash` should return `None` for the `_enabled_key` key when it is not enabled.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 380-381, 384, 388-390, 401-402, 408, 412-414) |

PASSED tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_no_session 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Verify that `pytest_collectreport` does not raise an exception when no session is available.

**Why Needed:** Prevent regression in plugin behavior when a Pytest session is not present.

**Key Assertions:**

- The function `pytest_collectreport(mock_report)` should not be called with a `session` attribute that is `None` or `False`.
- The function `pytest_collectreport(mock_report)` should raise an exception with the message 'No session available' when `mock_report.session` is `None` or `False`.
- The function `pytest_collectreport(mock_report)` should not call any internal functions that require a valid session.
- The function `pytest_collectreport(mock_report)` should not modify its internal state in any way.
- The function `pytest_collectreport(mock_report)` should not raise an exception when called with a mock object that does not have a `session` attribute.
- The function `pytest_collectreport(mock_report)` should not call the `collectreport` method on the mock object.
- The function `pytest_collectreport(mock_report)` should not call any other functions that require a valid session.
- The function `pytest_collectreport(mock_report)` should not raise an exception when called with a mock object that has no `session` attribute.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 380-381, 384, 388-390, 401, 405) |

PASSED tests/test_plugin_maximal.py::TestPluginCollectReport::test_pytest_collectreport_session_none 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test the `pytest_collectreport` plugin with a null session.

**Why Needed:** To prevent a potential bug where the plugin skips collect reports when the session is set to None.

**Key Assertions:**

- The `pytest_collectreport` function should not be called with a `None` session.
- No exception should be raised when calling `pytest_collectreport(mock_report)`.
- The `mock_report.session` attribute should still be `None` after the call to `pytest_collectreport()`.
- The `pytest_collectreport` function should not modify or raise an error with a null session.
- The plugin's behavior should remain unchanged even when the session is set to None.
- No assertion errors should occur in the test.
- The `pytest_collectreport` function should still be able to collect reports without raising an exception.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 380-381, 384, 388-390, 401, 405) |

PASSED `tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_llm_enabled_warning` 3ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that LLM enabled warning is raised when using the `pytest_llm_report` plugin.

**Why Needed:** This test prevents a potential regression where the LLM report provider might be set to 'ollama' without proper configuration or warnings being displayed.

**Key Assertions:**

- The `llm_report_provider` option should be set to `'ollama'` when using the `pytest_llm_report` plugin.
- The `llm_report_html`, `llm_report_json`, and `llm_report_pdf` options should not be set to `None` when using the `pytest_llm_report` plugin.
- The `llm_evidence_bundle`, `llm_dependency_snapshot`, `llm_requests_per_minute`, `llm_aggregate_dir`, `llm_aggregate_policy`, `llm_aggregate_run_id`, and `llm_aggregate_group_id` options should not be set to `None` when using the `pytest_llm_report` plugin.
- The `llm_max_retries` option should have a valid value (either 1 or greater) when using the `pytest_llm_report` plugin.
- The `rootpath` and `stash` options should not be set to an empty string or `None` when using the `pytest_llm_report` plugin.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 44 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-257, 259, 261, 263, 270, 272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294-295, 298, 300) |
| src/pytest_llm_report/plugin.py | 29 lines (ranges: 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203-205, 207-208, 212-213, 380-381, 384, 388-390) |

PASSED tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_validation_errors 3ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that validation errors raise UsageError when invalid configuration is provided.

**Why Needed:** To prevent a Potential Bug where the plugin fails to configure due to invalid or missing required options.

**Key Assertions:**

- Mocking `pytest_configure` with an invalid config raises `pytest.UsageError`.
- The `getini` method of `mock_config` is called with an invalid key, which triggers the error.
- The `option.llm_report_html`, `option.llm_report_json`, `option.llm_report_pdf`, `option.llm_evidence_bundle`, `option.llm_dependency_snapshot`, `option.llm_requests_per_minute`, `option.llm_aggregate_dir`, `option.llm_aggregate_policy`, `option.llm_aggregate_run_id`, `option.llm_aggregate_group_id` options are all set to None.
- The `rootpath` option is set to `/project`, which may not be a valid path for the plugin's configuration.
- The `stash` option is an empty dictionary, which may not be suitable for storing stash data.
- The `llm_max_retries` option is set to None, which may not be a valid value for this option.
- The `option.llm_aggregate_run_id` and `option.llm_aggregate_group_id` options are not being used in the test.
- The `pytest_configure` function is called with an invalid config, indicating that the plugin configuration failed to validate correctly.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 43 lines (ranges: 107, 147, 175, 178-181, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 248, 251-253, 255, 257, 259, 261, 263, 270, 272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294-295, 298, 300) |
| src/pytest_llm_report/plugin.py | 25 lines (ranges: 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-199, 380-381, 384, 388-390) |

PASSED `tests/test_plugin_maximal.py::TestPluginConfigure::test_pytest_configure_worker_skip` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test that configure skips on xdist workers.

**Why Needed:** This test prevents a regression where the plugin might skip configuration due to an incorrect assumption about the number of workers.

**Key Assertions:**

- mock_config.addinivalue_line was not called before worker check
- addinivalue_line is still called for markers before worker check

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 169-171, 173-175, 177-179, 183-184, 380-381, 384, 388-390)` |

PASSED `tests/test_plugin_maximal.py::TestPluginConfigureFallback::test_pytest_configure_fallback_load` 3ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test that fallback to load_config occurs when Config.load is missing.

**Why Needed:** Prevents a potential bug where the plugin fails to configure due to missing Config.load method.

**Key Assertions:**

- mock_load.assert_called_once()
- mock_cfg.validate.return_value == []
- pytest_configure(mock_config) was called with mock_config
- mock_load.return_value is not None
- mock_cfg.validate.return_value is not None
- mock_cfg.getini.return_value is None
- mock_cfg.option.llm_report_html is None
- mock_cfg.option.llm_max_retries is None

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 29 lines (ranges: 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203-205, 207-208, 212-213, 380-381, 384, 388-390) |

PASSED `tests/test_plugin_maximal.py::TestPluginLoadConfig::test_load_config_all_ini_options` 2ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test loading all INI options for plugin configuration.

**Why Needed:** Prevents a bug where the plugin load configuration is not properly set when CLI options are not provided.

**Key Assertions:**

- The `llm_report_provider` option should be set to 'ollama'.
- The `llm_report_model` option should be set to 'llama3.2'.
- The `llm_report_context_mode` option should be set to 'complete'.
- The `llm_report_requests_per_minute` option should be set to 10.
- The `report_html` option should be set to 'ini.html'.
- The `report_json` option should be set to 'ini.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 | 31 lines (ranges: 107, 147, 248, 251-263, 270, 272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294-295, 298, 300) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_plugin_maximal.py::TestPluginLoadConfig::test_load_config_cli_overrides_ini` 2ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test CLI options override INI options.

**Why Needed:** This test prevents a bug where the CLI options override INI options, causing unexpected behavior in the plugin's configuration.

**Key Assertions:**

- The `llm_report_html` option is set to 'cli.html' instead of 'ini.html'.
- The `llm_report_json` option is set to 'cli.json' instead of 'ini.json'.
- The `llm_report_pdf` option is set to 'cli.pdf' instead of 'ini.pdf'.
- The `llm_evidence_bundle` option is set to 'bundle.zip' instead of 'ini.evidence_bundle'.
- The `llm_dependency_snapshot` option is set to 'deps.json' instead of 'ini.dependency_snapshot'.
- The `llm_requests_per_minute` option is set to 20 instead of the expected value from INI.
- The `aggregate_dir` option is set to '/agg' instead of the expected value from INI.
- The `aggregate_policy` option is set to 'merge' instead of the expected value from INI.
- The `aggregate_run_id` option is set to 'run-123' instead of the expected value from INI.
- The `aggregate_group_id` option is set to 'group-abc' instead of the expected value from INI.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 38 lines (ranges: 107, 147, 248, 251, 253, 255, 257, 259-263, 270-283, 286-295, 298, 300) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_plugin_maximal.py::TestPluginMaximal::test_terminal_summary_disabled` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test that terminal summary skips when plugin is disabled.

**Why Needed:** This test prevents a regression where the plugin's terminal summary might be incorrectly reported when it is disabled.

**Key Assertions:**

- The `pytest_terminal_summary` function should not report any results when the plugin is disabled.
- The `stash.get` method was called with the correct key and value (False) to check for enabled status.
- The `stash.get` method was not called at all when checking if the plugin is enabled.
- The `pytest_terminal_summary` function did not report any results even though it should have, indicating a bug in its logic.
- The test should fail with an assertion error when the plugin is disabled and terminal summary is expected to be reported.
- The test should pass without any assertions or errors when the plugin is enabled and terminal summary is not reported.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 238, 242-243, 380-381, 384, 388-390) |

PASSED `tests/test_plugin_maximal.py::TestPluginMaximal::test_terminal_summary_worker_skip` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test that terminal summary skips on xdist worker when given a valid configuration.

**Why Needed:** This test prevents the plugin from attempting to process the terminal summary for an xdist worker with a valid configuration.

**Key Assertions:**

- The `pytest_terminal_summary` function should return None without doing anything when given a valid configuration.
- The `workerinput` attribute of the mock config object should contain the expected value ('gw0')
- The `terminal_summary_worker_skip` method should not attempt to process the terminal summary for an xdist worker with a valid configuration
- No output or exceptions should be raised during execution of this test
- The test should pass if the plugin is properly configured and the xdist worker is skipped correctly

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 238-239, 380-381, 384, 388-390)` |

PASSED tests/test_plugin_maximal.py::TestPluginMaximal::testload_config 3ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test config loading from pytest objects (CLI + INI) to ensure correct configuration retrieval.

**Why Needed:** This test prevents regression in the plugin's functionality when using pytest as a CLI tool or when INI files are used for configuration.

**Key Assertions:**

- The `report_html` attribute of the loaded configuration object is set to 'out.html'.
- The `llm_report_json` attribute of the loaded configuration object is set to 'out.json'.
- The `rootpath` attribute of the loaded configuration object is set to '/root'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 36 lines (ranges: 107, 147, 248, 251, 253, 255, 257, 259, 261, 263, 270-283, 286-295, 298, 300) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_plugin_maximal.py::TestPluginRuntest::test_runtest_makereport_disabled` 2ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test makereport skips when disabled.

**Why Needed:** Prevents a regression where the plugin's report is not generated due to an unhandled generator.

**Key Assertions:**

- mock_item.config.stash.get() returns False for the mock item.
- mock_call() does not raise an exception when called with mock_outcome.
- gen.send(mock_outcome) raises StopIteration and passes it to next(gen),
- mock_outcome.get_result().get_result() is mocked but not tested.
- mock_item.config.stash.get() returns False instead of raising an assertion error.
- mock_call() does not raise an exception when called with mock_outcome.
- gen.send(mock_outcome) raises StopIteration and passes it to next(gen),
- mock_outcome.get_result().get_result() is mocked but not tested.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 380-381, 384-385, 388-390) |

PASSED tests/test_plugin_maximal.py::TestPluginRuntest::test_runtest_makereport_enabled 2ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test that makereport calls collector when enabled.

**Why Needed:** This test prevents a potential bug where the collector is not called even though makereport is enabled.

**Key Assertions:**

- The `pytest_runtest_makereport` function should be able to find and call the `mock_collector` instance when it is enabled.
- The `mock_collector.handle_runtest_logreport` method should be called with the correct arguments (the `mock_report` instance and the `mock_item` instance).
- The `mock_collector` instance should have a `handle_runtest_logreport` method that can be called without raising an exception.
- The `pytest_runtest_makereport` function should not raise a `StopIteration` exception when it is unable to find the collector.
- The `mock_item.config.stash.get` method should return `True` for the `_enabled_key` and `_collector_key` keys when the collector is enabled.
- The `pytest_runtest_makereport` function should not raise an exception if the collector is disabled or does not exist.
- The `mock_collector.handle_runtest_logreport` method should be called with the correct arguments (the `mock_report` instance and the `mock_item` instance) even when the collector is disabled.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |

PASSED `tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_collection_finish_disabled` 1ms 🛡️ 2

AI ASSESSMENT

**Scenario:** Test that collection_finish is skipped when disabled in Pytest.

**Why Needed:** To prevent a regression where Pytest's collection finish feature is not properly handled when the plugin is disabled.

**Key Assertions:**

- Mocking `pytest_collection_finish` with `mock_session.config.stash.get.return_value = False` to verify it does not call `_enabled_key`.
- Verifying that `mock_session.config.stash.get.return_value` returns `False` after calling `pytest_collection_finish`.
- Asserting that `pytest_collection_finish` is called without arguments when the plugin is disabled.
- Checking if the `_enabled_key` variable is set to `'pytest'` before calling it.
- Verifying that no exception is raised during the execution of `mock_session.config.stash.get.return_value = False`.
- Confirming that the mock session's stash value remains unchanged after calling `pytest_collection_finish`.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 380-381, 384, 388-390, 424-425) |

PASSED `tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_collection_finish_enabled` 2ms 🛡 2

AI ASSESSMENT

**Scenario:** Test that collection_finish is called when pytest CollectionFinish is enabled.

**Why Needed:** This test prevents a potential regression where pytest CollectionFinish is disabled and the collector is not properly cleaned up.

**Key Assertions:**

- The stash_get function returns True for _enabled_key and mock_collector.
- The stash_get function returns mock_collector for _collector_key.
- mock_collector.handle_collection_finish is called once with mock_session.items.
- The collection_finish method of the collector is not called when pytest CollectionFinish is disabled.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 380-381, 384, 388-390, 424, 428-430) |

PASSED `tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_sessionstart_disabled` 1ms 🛡 2

AI ASSESSMENT

**Scenario:** Test that sessionstart skips when disabled and checks enabled status.

**Why Needed:** Prevents a potential bug where the plugin fails to check the enabled status of the pytest session.

**Key Assertions:**

- mock_session.config.stash.get.assert_called_with(_enabled_key, False)
- pytest_sessionstart(mock_session) should not be called
- mock_session.config.stash.get.return_value should have been set to False

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 380-381, 384, 388-390, 441-442) |

PASSED `tests/test_plugin_maximal.py::TestPluginSessionHooks::test_pytest_sessionstart_enabled` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that sessionstart initializes collector when enabled and the collector is created correctly.

**Why Needed:** Prevents a potential bug where the collector is not initialized or does not exist when pytest_sessionstart is called with an enabled configuration.

**Key Assertions:**

- The '_collector_key' should be present in the mock stash.
- The '_start_time_key' should be present in the mock stash.
- The collector should have been created successfully.
- The collector's key should match _collector_key.
- The start time of the collection should be recorded correctly.
- The session start time should be available through pytest_sessionstart.
- The configuration should support get() and [] operations.
- The stash dictionary should contain _enabled_key and _config_key.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147)` |
| `src/pytest_llm_report/plugin.py` | `11 lines (ranges: 380-381, 384, 388-390, 441, 445, 448, 450-451)` |

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 prevents a potential bug where the plugin does not add all required arguments to the command line.

**Key Assertions:**

- parser.getgroup.assert_called_with('llm-report', 'LLM-enhanced test reports')
- group.addoption.call_args_list[0][0].startswith('--llm-report')
- group.addoption.call_args_list[1][0].startswith('--llm-coverage-source')

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 99 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 380-381, 384, 388-390) |

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_pytest_addoption_ini` 2ms 🛡️ 2

AI ASSESSMENT

**Scenario:** The test verifies that pytest_addoption adds INI options (lines 13-34) to the parser.

**Why Needed:** This test prevents a regression where pytest_addoption does not add INI options, potentially causing issues with plugin functionality.

**Key Assertions:**

- llm_report_html is added as an option
- llm_report_json is added as an option
- llm_report_max_retries is added as an option

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `99 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 380-381, 384, 388-390)` |

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_terminal_summary_coverage_calculation` 4ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test coverage percentage calculation logic for terminal summary.

**Why Needed:** This test prevents regression in coverage calculation when using the `pytest_terminal_summary` plugin with a mock configuration.

**Key Assertions:**

- The `report_html` option is set to 'out.html' and the `Coverage` class is created correctly.
- The `stash` dictionary is populated with correct values.
- The `mock_cov_cls.return_value` method is called once when creating a mock coverage object.
- The `mock_cov.report.return_value` method is called once when reporting coverage.
- The `patch` objects for `pathlib.Path.exists`, `coverage.Coverage`, and `pytest_llm_report.coverage_map.CoverageMapper` are created correctly.
- The `MockStash` class is used to mock the `stash` dictionary.
- The `MagicMock` object is used as a mock configuration instance.
- The `pytest_terminal_summary` function is called with correct arguments and returns a mock coverage report.

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: 107, 147, 224)` |
| `src/pytest_llm_report/plugin.py` | `53 lines (ranges: 238, 242, 246, 249, 268-269, 271, 273, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-305, 307, 309-312, 324-325, 330-331, 358-368, 380-381, 384, 388-390)` |

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_terminal_summary_llm_enabled` 3ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that terminal summary with LLM enabled runs annotations correctly when provider is 'ollama' and report HTML is set to 'out.html'

**Why Needed:** Prevents regression in terminal summary LLM functionality when the provider is 'ollama' and report HTML is set to 'out.html'.

**Key Assertions:**

- The function `pytest_terminal_summary` should be called with correct arguments.
- The `cfg` variable should have a value of `True` for the `_enabled_key` key.
- The `stash` dictionary should contain the expected values for the `_enabled_key` and `_config_key` keys.
- The `mock_config.stash` attribute should be set to the `MockStash` instance with the provided `stash` dictionary.
- The `mock_terminalreporter.stats` attribute should not be modified before calling `pytest_terminal_summary`.
- The `mock_annotate.call_args[0][1]` assertion should verify that the correct configuration is passed to `pytest_terminal_summary`.

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: 107, 147, 224) |
| `src/pytest_llm_report/plugin.py` | 59 lines (ranges: 238, 242, 246, 249, 268-269, 271, 273, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324-325, 330-333, 336, 338, 341-343, 350-355, 358-368, 380-381, 384, 388-390) |

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 regression where the plugin does not create a collector even when it is supposed to be missing.

**Key Assertions:**

- The stash object returned by pytest_terminal_summary() is correctly set to False.
- The coverage mapper returns an empty dictionary.
- The mock terminal reporter logs a message indicating that no collector was created.
- The stash object returned by pytest_terminal_summary() has the correct key-value pairs.
- The mock writer does not write any output when calling pytest_terminal_summary(0, mock_config).
- The coverage mapper returns an empty dictionary even after calling pytest_terminal_summary(0, mock_config).
- The mock terminal reporter logs a message indicating that no collector was created and the stash object returned by pytest_terminal_summary() is correctly set to False.
- The mock writer does not write any output when calling pytest_terminal_summary(0, mock_config).

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: 107, 147, 224) |
| src/pytest_llm_report/plugin.py | 45 lines (ranges: 238, 242, 246, 249, 268-269, 271, 273, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390) |

PASSED tests/test_plugin_maximal.py::TestPluginTerminalSummary::test_terminal_summary_with_aggregation 2ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test terminal summary with aggregation enabled.

**Why Needed:** This test prevents regression in the aggregation feature by ensuring it is properly configured and executed.

**Key Assertions:**

- The `aggregate_dir` parameter is set to `/agg` and the `report_html` and `report_json` parameters are set to `'out.html'` and `'out.json'`, respectively.
- The `stash` object has `_enabled_key` set to `True` and `_config_key` set to the configured `Config` instance.
- The `aggregate` method of the `Aggregator` class is called once with a mock report object.
- The `ReportWriter` class's `write_json` and `write_html` methods are called once each.
- The `pytest_terminal_summary` function is patched to call the `aggregate` method on the `Aggregator` instance.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 21 lines (ranges: 238, 242, 246, 249-250, 252-253, 256-257, 259, 261-265, 380-381, 384, 388-390) |

PASSED `tests/test_plugin_maximal.py::TestPluginTerminalSummaryErrors::test_terminal_summary_coverage_error` 4ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test coverage calculation error when loading a coverage map with an OSError.

**Why Needed:** This test prevents the 'Failed to compute coverage percentage' UserWarning from being raised during terminal summary generation.

**Key Assertions:**

- mock_cov.return_value.load.assert_called_once_with('coverage')
- mock_cov_cls.return_value.load.assert_called_once_with('coverage')
- mock_cov.return_value.load.side_effect == OSError("Disk full")
- pytest_terminal_summary(MagicMock(), 0, mock_config).report_html is not None
- pytest_terminal_summary(MagicMock(), 0, mock_config).report_html contains "Failed to compute coverage percentage"
- pytest_terminal_summary(MagicMock(), 0, mock_config).report_html does not contain "Disk full"
- mock_cov.return_value.load.call_count == 1
- mock_cov_cls.return_value.load.call_count == 1

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: 107, 147, 224) |
| `src/pytest_llm_report/plugin.py` | 52 lines (ranges: 238, 242, 246, 249, 268-269, 271, 273, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-305, 315-318, 324-325, 330-331, 358-368, 380-381, 384, 388-390) |

PASSED `tests/test_prompts.py::TestContextAssembler::test_assemble_balanced_context` 7ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test the ContextAssembler to assemble a balanced context with a test file and dependency.

**Why Needed:** This test prevents regression when assembling contexts with unbalanced dependencies, where only one module is imported.

**Key Assertions:**

- The 'utils.py' file should be present in the assembled context.
- The 'def util()' function should be found in the 'utils.py' file of the assembled context.
- Only 'test_a.py::test_1' should be included in the assembled context's source code.
- The 'utils.py' file should contain a single line with a count of 2 lines.
- The 'def util()' function should not have any other imports or uses.
- No other modules should be imported from outside the 'test_a.py::test_1' scope.
- The assembled context's source code should only include 'test_a.py::test_1'.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/prompts.py` | 51 lines (ranges: 33, 49, 52, 55, 58, 60-61, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 116, 132, 135-138, 140-141, 144-145, 148, 151-152, 154-155, 158-159, 163, 191-192, 194) |

PASSED `tests/test_prompts.py::TestContextAssembler::test_assemble_complete_context` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test that the ContextAssembler can assemble a complete context for a test file with no external dependencies.

**Why Needed:** To prevent regression and ensure that tests like `tests/test_prompts.py::TestContextAssembler::test_assemble_complete_context` pass correctly even when the test file has no external dependencies.

**Key Assertions:**

- The 'test_1' function is present in the source code of the assembled context.
- The 'test_a.py::test_1' path is present in the source code of the assembled context.
- The 'test_1' function is executed within the assembled context.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/prompts.py | 34 lines (ranges: 33, 49, 52, 55, 58, 60, 63, 65, 78-79, 82-83, 86-87, 92, 94, 98-101, 103-112, 116, 132-133, 180) |

PASSED `tests/test_prompts.py::TestContextAssembler::test_assemble_minimal_context` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Assembling a minimal context for testing `test_1` in `test_a.py`

**Why Needed:** Prevents regression by ensuring the minimal context is assembled correctly when `test_1` is tested.

**Key Assertions:**

- The 'test_1' function should be present in the source code of `test_a.py`.
- The assembly result should not include any additional context.
- The context object should have an empty dictionary as its value.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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 🛡️ 4

AI ASSESSMENT

**Scenario:** Test the ContextAssembler with balanced context limits to ensure it correctly truncates long content exceeding 20 bytes.

**Why Needed:** This test prevents a potential bug where the ContextAssembler does not truncate long content exceeding 20 bytes, potentially leading to incorrect results or errors.

**Key Assertions:**

- The 'f1.py' file in the context is truncated to 40 bytes or less.
- The 'f1.py' file contains the string 'truncated'.
- The length of the 'f1.py' file is within the allowed limit (20 + truncation message).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/prompts.py | 34 lines (ranges: 33, 49, 52, 55, 58, 60-61, 65, 78-79, 82-84, 132, 135-138, 140-141, 144-145, 148, 151-152, 154-156, 158-159, 163, 191-192, 194) |

PASSED `tests/test_prompts.py::TestContextAssembler::test_get_test_source_edge_cases` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Verify the correct handling of non-existent files and nested test names with parameters

**Why Needed:** This test prevents a potential bug where an invalid file path is used to retrieve the test source.

**Key Assertions:**

- The function `_get_test_source` returns an empty string for a non-existent file.
- The function `_get_test_source` correctly extracts the nested test name with parameters from the given file path.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/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: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `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 🛡️ 4

AI ASSESSMENT

**Scenario:** The test verifies that the `ContextAssembler` should exclude certain files based on their glob patterns.

**Why Needed:** This test prevents a potential bug where certain files are included in the output even though they should not be, potentially leading to unexpected behavior or errors.

**Key Assertions:**

- assert assembler._should_exclude('test.pyc') is True
- assert assembler._should_exclude('secret/key.txt') is True
- assert assembler._should_exclude('public/readme.md') is False

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/prompts.py | 5 lines (ranges: 33, 191-194) |

PASSED `tests/test_ranges.py::TestCompressRanges::test_consecutive_lines` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that consecutive lines are compressed to their correct ranges.

**Why Needed:** This test prevents a regression where consecutive numbers are not correctly compressed to their ranges.

**Key Assertions:**

- asserts that the compress_ranges function returns the expected range string for consecutive lines
- comparing the output of compress_ranges([1, 2, 3]) with '1-3'
- verifies that the output is correct and does not contain any incorrect ranges

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** The test verifies that the `compress_ranges` function correctly handles duplicate ranges.

**Why Needed:** This test prevents a potential bug where the function incorrectly identifies non-duplicate ranges as duplicates.

**Key Assertions:**

- assert compress_ranges([1, 2, 2, 3, 3, 3]) == '1-3'
- assert compress_ranges([1, 1, 2, 2, 3, 3]) == '1-3'
- assert compress_ranges([1, 2, 2, 2, 3, 3]) == '1-3'
- assert compress_ranges([1, 2, 3, 3, 3]) == '1-4'
- assert compress_ranges([]) == ''
- assert compress_ranges([1]) == '1-'
- assert compress_ranges([1, 1]) == '1-'

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `src/pytest_llm_report/util/ranges.py` | `12 lines (ranges: 29, 33, 35-37, 39-40, 42, 50, 52, 65, 67)` |

PASSED `tests/test_ranges.py::TestCompressRanges::test_empty_list` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Testing the `compress_ranges` function with an empty list.

**Why Needed:** The `compress_ranges` function is expected to return an empty string for an empty input list, as it may be used in various downstream operations such as data compression or analysis.

**Key Assertions:**

- assert compress_ranges([]) == "",
- assert str(compress_ranges([1, 2])) == ""
- assert str(compress_ranges([-1, -2])) == ""
- assert str(compress_ranges([1, 2, 3])) == ""
- assert str(compress_ranges([])) == ""
- assert compress_ranges([1]) == ""
- assert compress_ranges([-1]) == ""
- assert compress_ranges([1, -1]) == ""

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `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:** Test compresses mixed ranges and singles into a single string.

**Why Needed:** This test prevents regression when mixing ranges (e.g., 1-3) with singles (e.g., 5).

**Key Assertions:**

- The function correctly groups the ranges and singles together in the output.
- The range '1-3' is included in the output.
- The single value '5' is included in the output.
- The range '10-12' is included in the output.
- The single value '15' is included in the output.
- The function handles edge cases where a range is on one end of the list and there are no other ranges or singles before it.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `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:** Test the `compress_ranges` function with non-consecutive line numbers.

**Why Needed:** This test prevents regression in case of non-consecutive lines.

**Key Assertions:**

- The output should be a comma-separated string containing all numbers from the input list.
- The numbers should be consecutive within each group (e.g., 1, 2, 3).
- If there are any gaps between consecutive numbers, they should be represented as empty strings.
- Non-consecutive line numbers should not be included in the output.
- The function should handle lists with an odd number of elements correctly.
- The test should pass even if the input list contains duplicate numbers.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** The 'single_line' test verifies that a single line of code does not use the range notation.

**Why Needed:** This test prevents regression where the function compresses a list with only one element using range notation.

**Key Assertions:**

- assert compress_ranges([5]) == '5'
- assert len(compress_ranges(['1', '2', '3'])) == 1
- assert isinstance(compress_ranges(['1', '2', '3']), str)
- assert all(isinstance(x, int) for x in compress_ranges(['1', '2', '3'])),
- assert all(isinstance(y, str) for y in compress_ranges(['1', '2', '3'])),
- assert len(compress_ranges(['1'])) == 1
- assert isinstance(compress_ranges(['1']), str)

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/util/ranges.py` | 10 lines (ranges: 29, 33, 35-37, 39, 50, 52, 65-66) |

PASSED `tests/test_ranges.py::TestCompressRanges::test_two_consecutive` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The 'test_two_consecutive' test verifies that two consecutive lines in a list of integers are compressed to their ranges.

**Why Needed:** This test prevents regression where non-consecutive lines are incorrectly compressed to single numbers.

**Key Assertions:**

- assert compress_ranges([1, 2]) == '1-2'
- assert compress_ranges([3, 4]) == '3-4'
- assert compress_ranges([]) == ''
- assert compress_ranges([-5, -10]) == '-5-10'
- assert compress_ranges([1, 2, 3]) == '1-3'
- assert compress_ranges([1, 2, 3, 4]) == '1-4'
- assert compress_ranges([5, 6, 7]) == '5-7'

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** Test 'test_unsorted_input' verifies that the function handles unsorted input correctly.

**Why Needed:** This test prevents a potential bug where the function may return incorrect results for unsorted input.

**Key Assertions:**

- The input list is sorted in ascending order before being passed to the `compress_ranges` function.
- The output string contains the expected range notation (e.g., '1-3, 5').
- The function correctly handles duplicate values within the same range.
- No incorrect results are produced for unsorted ranges with multiple elements.
- The input list is sorted in descending order before being passed to the `compress_ranges` function.
- The output string contains the expected range notation (e.g., '5-1, 3'), even if the input list is unsorted.
- No incorrect results are produced for unsorted ranges with duplicate values within the same range.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `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:** Testing the `expand_ranges` function with an empty input.

**Why Needed:** This test prevents a potential bug where the function incorrectly handles empty strings as valid inputs.

**Key Assertions:**

- assert expand_ranges([]) == []
- assert expand_ranges('') == []
- assert expand_ranges(None) == []

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/util/ranges.py | 2 lines (ranges: 81-82) |

PASSED `tests/test_ranges.py::TestExpandRanges::test_mixed` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'test_mixed' verifies that the `expand_ranges` function correctly handles mixed ranges and singles.

**Why Needed:** This test prevents a potential bug where the function incorrectly expands single numbers into multiple ranges.

**Key Assertions:**

- The input string should be parsed into separate ranges and singles.
- Each range should contain exactly one integer value.
- Single numbers should not be expanded into multiple ranges.
- All integers in the input string should be present in the output list.
- The function should handle invalid input strings without raising an exception.
- The function should preserve the original order of single numbers within each range.
- The function should correctly handle cases where a range spans across multiple lines.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/util/ranges.py | 11 lines (ranges: 81, 84-91, 93, 95) |

PASSED `tests/test_ranges.py::TestExpandRanges::test_range` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The 'expand_ranges' function is expected to correctly expand a range of numbers.

**Why Needed:** This test prevents the function from expanding ranges that contain invalid or out-of-range values.

**Key Assertions:**

- The function should return a list of integers between 1 and 3 (inclusive).
- The function should handle negative numbers correctly.
- The function should not expand ranges containing non-numeric characters or strings.
- The function should raise an error for invalid input (e.g. 'abc').
- The function should return the correct range when given a single number.
- The function should handle edge cases where the input is empty.
- The function should correctly handle ranges with multiple elements.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `src/pytest_llm_report/util/ranges.py` | `10 lines (ranges: 81, 84-91, 95)` |

PASSED tests/test_ranges.py::TestExpandRanges::test_roundtrip 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that the `compress_ranges` and `expand_ranges` functions return the same input when called in reverse order.

**Why Needed:** This test prevents a potential bug where the inverse of these functions would cause incorrect results or unexpected behavior.

**Key Assertions:**

- The original list `[1, 2, 3, 5, 10, 11, 12, 15]` should be equal to its expanded version `original` after calling `expand_ranges(compressed)`.
- The compressed range `[1, 2, 4]` should be equivalent to the original range `[1, 2, 3, 5]` when called in reverse order.
- The original list `[10, 11, 12, 15]` should be equal to its expanded version `original` after calling `expand_ranges(compressed)`.
- The compressed range `[10, 11, 12]` should be equivalent to the original range `[1, 2, 3, 5]` when called in reverse order.
- The original list `[15, 10, 11, 12]` should be equal to its expanded version `original` after calling `expand_ranges(compressed)`.
- The compressed range `[15, 10, 11, 12]` should be equivalent to the original range `[1, 2, 3, 5]` when called in reverse order.
- The function `compress_ranges(original)` should return a list that can be used as input for `expand_ranges()` without any modifications.
- The function `expand_ranges(compressed)` should take the compressed list and expand it back to its original form without any changes.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** The test verifies that the `expand_ranges` function returns a list containing only one element when given a single number.

**Why Needed:** This test prevents a potential bug where the function incorrectly expands ranges for a single number, potentially leading to incorrect results or unexpected behavior.

**Key Assertions:**

- assert expand_ranges('5') == [5]
- assert len(expand_ranges('5')) == 1
- assert isinstance(expand_ranges('5'), list)
- assert all(isinstance(x, int) for x in expand_ranges('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/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/util/ranges.py | 7 lines (ranges: 81, 84-87, 93, 95) |

PASSED `tests/test_render.py::TestFormatDuration::test_milliseconds` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the format duration function for milliseconds when input is less than 1 second.

**Why Needed:** This test prevents a potential regression where the function does not correctly format durations in milliseconds for inputs less than 1 second.

**Key Assertions:**

- {'message': "Format should be 'ms' for input < 1s", 'description': "Expected output is '500ms' when input is 0.5"}
- {'message': "Format should be 'ms' for input < 1s", 'description': "Expected output is '1ms' when input is 0.001"}
- {'message': "Format should be 'ms' for input < 1s", 'description': "Expected output is '0ms' when input is 0.0"}

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/render.py` | 2 lines (ranges: 65, 67) |

PASSED tests/test_render.py::TestFormatDuration::test_seconds 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** tests/test_render.py::TestFormatDuration::test_seconds

**Why Needed:** Prevents regression in formatting of seconds.

**Key Assertions:**

- The function `format_duration(x)` correctly formats the input as a string with two decimal places.
- It handles inputs greater than or equal to 1 second correctly.
- It correctly handles inputs greater than or equal to 60 seconds correctly.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/render.py | 2 lines (ranges: 65-66) |

PASSED `tests/test_render.py::TestOutcomeToCssClass::test_all_outcomes` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** All outcomes should map to CSS classes.

**Why Needed:** This test prevents regression when new outcomes are introduced, ensuring consistency in the mapping of outcomes to CSS classes.

**Key Assertions:**

- outcome_to_css_class('passed') == 'outcome-passed'
- outcome_to_css_class('failed') == 'outcome-failed'
- outcome_to_css_class('skipped') == 'outcome-skipped'
- outcome_to_css_class('xfailed') == 'outcome-xfailed'
- outcome_to_css_class('xpassed') == 'outcome-xpassed'
- outcome_to_css_class('error') == 'outcome-error'

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/render.py | 8 lines (ranges: 79-85, 87) |

PASSED `tests/test_render.py::TestOutcomeToCssClass::test_unknown_outcome` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Tests for `outcome_to_css_class` function with unknown outcome.

**Why Needed:** Prevents regression in case of unknown outcomes, ensuring consistent styling.

**Key Assertions:**

- The function should return the default class 'outcome-unknown' when given an unknown outcome.
- The function should not throw any errors or exceptions when given an unknown outcome.
- The function should maintain its original behavior for known outcomes.
- The function should be able to handle cases where the outcome is not recognized by checking if it's in a list of known outcomes.
- The function should not throw any warnings or notices when given an unknown outcome.
- The function should return the correct class name based on the input value.
- The function should maintain its original case sensitivity for unknown outcomes (e.g., 'Unknown' and 'unknown' are considered different values).

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `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 malformed reports.

**Key Assertions:**

- The '' header should be present in the rendered HTML.
- The 'Test Report' section should be found in the rendered HTML.
- Each test result node should contain either 'PASSED' or 'FAILED' and an optional error message.
- Plugin and repo version information should be included in the rendered HTML.
- The report summary should display total, passed, and failed counts.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `src/pytest_llm_report/render.py` | `52 lines (ranges: 65-67, 79-85, 87, 121-124, 126-127, 131-132, 141-143, 145-153, 158-160, 196, 229-236, 239-245, 248-249)` |

PASSED `tests/test_render.py::TestRenderFallbackHtml::test_renders_coverage` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verifies that the test renders a fallback HTML with coverage information.

**Why Needed:** Prevents regression when coverage is not provided or is incomplete.

**Key Assertions:**

- The report root contains a 'tests' list with a single test case.
- The test case has an 'outcome' of 'passed' and a 'coverage' attribute containing a 'CoverageEntry' with 'file_path', 'line_ranges', and 'line_count' attributes.
- The HTML rendered by the function includes the file path 'src/foo.py' and contains 5 lines.
- The test case is included in the report root's 'tests' list.
- The coverage information is available in the report root.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/render.py | 52 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-129, 131-132, 141-142, 145-153, 158-160, 196, 229-236, 239-245, 248-249) |

PASSED `tests/test_render.py::TestRenderFallbackHtml::test_renders_llm_annotation` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test renders LLM annotation for login flow scenario.

**Why Needed:** This test prevents authentication bypass by ensuring the LLM annotation is present in the rendered HTML.

**Key Assertions:**

- The 'Tests login flow' string should be included in the rendered HTML.
- The 'Prevents auth bypass' string should be included in the rendered HTML.
- The LLM annotation for the 'Tests login flow' scenario should be present in the rendered HTML.
- The LLM annotation for the 'Prevents auth bypass' scenario should be present in the rendered HTML.
- The LLM annotation should not be empty or null.
- The LLM annotation should contain the correct text ('Tests login flow')
- The LLM annotation should contain the correct text ('Prevents auth bypass')

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/render.py` | 54 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-127, 131-134, 136-137, 141-142, 145-153, 158-160, 196, 229-236, 239-245, 248-249) |

PASSED `tests/test_render.py::TestRenderFallbackHtml::test_renders_source_coverage` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Tests the report root to include source coverage summary when rendering fallback HTML.

**Why Needed:** This test prevents regression by ensuring that source coverage summaries are included in fallback HTML, which is necessary for accurate reporting and debugging.

**Key Assertions:**

- The 'Source Coverage' section should be present in the rendered HTML.
- The report root should include a 'Source Coverage' section with the correct summary statistics.
- The 'Source Coverage' section should display the correct coverage percentage.
- The 'Source Coverage' section should include the source file path ('src/foo.py').
- The 'Source Coverage' section should display the correct number of statements (10) and missed statements (2).
- The 'Source Coverage' section should display the correct coverage percentage (80.0%) and covered ranges (1-4, 6-8).
- The 'Source Coverage' section should include the correct number of covered statements (8) and missed statements (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: 380-381, 384, 388-390) |
| `src/pytest_llm_report/render.py` | 63 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-127, 131-132, 141-142, 145-153, 158-164, 166-172, 177, 192, 196, 229-236, 239-245, 248-249) |

PASSED `tests/test_render.py::TestRenderFallbackHtml::test_renders_xpass_summary` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'Should include xfailed/xpassed summary entries' verifies that the report includes both XFailed and XPassed summaries.

**Why Needed:** This test prevents a regression where the report does not display XPassed summaries for tests with outcomes xfailed.

**Key Assertions:**

- The HTML contains the string 'XFailed', indicating that it is present in the summary.
- The HTML contains the string 'XPassed', indicating that it is present in the summary.
- The test passes if both 'XFailed' and 'XPassed' are found in the rendered HTML.
- The report includes XPassed summaries for tests with outcomes xfailed, preventing a regression.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/render.py` | 50 lines (ranges: 65, 67, 79-85, 87, 121-124, 126-127, 131-132, 141-142, 145-153, 158-160, 196, 229-236, 239-245, 248-249) |

PASSED `tests/test_report_writer.py::TestComputeSha256::test_different_content` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'different_content' verifies that the SHA-256 hash of different content produces different hashes.

**Why Needed:** This test prevents a potential bug where two different pieces of content produce the same SHA-256 hash, potentially leading to incorrect reporting or analysis.

**Key Assertions:**

- The function `compute_sha256(b'hello')` returns a unique hash for each input.
- The function `compute_sha256(b'world')` returns a unique hash for each input.
- The two generated hashes are different.
- If the inputs were swapped, e.g., `compute_sha256(b'world')` instead of `compute_sha256(b'hello')`, the test would fail.
- The function does not produce the same hash when given the same input but with different encoding (e.g., bytes vs. string).
- If the inputs were mixed, e.g., `compute_sha256(b'hello')` followed by `compute_sha256(b'world')`, the test would fail.
- The function handles non-string inputs correctly and produces a hash for those as well.
- The function is thread-safe and does not have any known side effects that could cause it to produce different hashes in certain scenarios.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 1 lines (ranges: 55) |

PASSED `tests/test_report_writer.py::TestComputeSha256::test_empty_bytes` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Testing the empty bytes case for consistency and correctness of the SHA-256 hash.

**Why Needed:** This test prevents a potential bug that could cause inconsistent or incorrect results when computing the SHA-256 hash with an empty byte string.

**Key Assertions:**

- The two computed hashes should be equal (i.e., they should produce the same output).
- Both hashes should have a length of 64 bytes (the expected hexadecimal representation of a SHA-256 hash).

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `src/pytest_llm_report/report_writer.py` | `1 lines (ranges: 55)` |

PASSED `tests/test_report_writer.py::TestReportWriter::test_build_run_meta` 5ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test ReportWriter::test_build_run_meta verifies that the build run meta includes version info.

**Why Needed:** This test prevents regression where the report writer does not include version information in the build run meta.

**Key Assertions:**

- assert meta.duration == 60.0
- assert meta.pytest_version
- assert meta.plugin_version == '0.1.0'
- assert meta.python_version

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 67 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) |

PASSED `tests/test_report_writer.py::TestReportWriter::test_build_summary_all_outcomes` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test that the `build_summary` method counts all outcome types and their corresponding test results.

**Why Needed:** This test prevents a regression where the total count of outcomes is incorrect when there are mixed success, failure, and skipped test cases.

**Key Assertions:**

- The `total` attribute should be equal to 6 (number of tests with all outcome types).
- The `passed` attribute should be equal to 1 (number of passed tests).
- The `failed` attribute should be equal to 1 (number of failed tests).
- The `skipped` attribute should be equal to 1 (number of skipped tests).
- The `xfailed` attribute should be equal to 1 (number of tests with 'x' outcome type and failure).
- The `xpassed` attribute should be equal to 1 (number of tests with 'x' outcome type and passed).
- The `error` attribute should be equal to 1 (number of tests with 'error' outcome type).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 19 lines (ranges: 156-158, 312, 314-315, 317-328, 330) |

PASSED `tests/test_report_writer.py::TestReportWriter::test_build_summary_counts` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test the ReportWriter's ability to build a summary with correct counts of outcomes.

**Why Needed:** This test prevents regression in the ReportWriter's functionality, ensuring that it correctly counts the total number of tests and their respective outcomes.

**Key Assertions:**

- The total count of all tests should be equal to the sum of passed and failed tests.
- The count of passed tests should match the given number (2 in this case).
- The count of failed tests should also match the given number (1 in this case).
- The count of skipped tests should be 0, as there are no skipped tests in this test set.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 13 lines (ranges: 156-158, 312, 314-315, 317-322, 330) |

PASSED `tests/test_report_writer.py::TestReportWriter::test_create_writer` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test that the `ReportWriter` initializes correctly with a given configuration.

**Why Needed:** This test prevents a potential bug where the `ReportWriter` does not initialize with the expected configuration.

**Key Assertions:**

- The `config` attribute of the `writer` object is set to the provided `Config` instance.
- The `warnings` list of the `writer` object is empty.
- The `artifacts` list of the `writer` object is empty.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 3 lines (ranges: 156-158) |

PASSED `tests/test_report_writer.py::TestReportWriter::test_write_report_assembles_tests` 5ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test 'ReportWriter::test_write_report_assembles_tests' verifies that the ReportWriter class writes a report with all tests.

**Why Needed:** This test prevents regression in case where no output paths are specified for the report configuration, causing the report to be empty.

**Key Assertions:**

- The length of `report.tests` should be equal to 2 (number of tests)
- The value of `report.summary.total` should be equal to 2 (total number of tests)
- All test nodes have a 'nodeid' attribute with the correct value
- Each test node has an 'outcome' attribute with one of 'passed' or 'failed'
- All test nodes are included in the report
- The report summary contains only two tests

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 93 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, 312, 314-315, 317-320, 330) |

PASSED `tests/test_report_writer.py::TestReportWriter::test_write_report_includes_coverage_percent` 6ms 🛡️ 4

AI ASSESSMENT

**Scenario:** The test verifies that the `ReportWriter` class writes a report with a total coverage percentage.

**Why Needed:** This test prevents regressions where the coverage percentage is not included in the report.

**Key Assertions:**

- The `summary.coverage_total_percent` attribute of the report should match the provided coverage percentage.
- The `report.write_report()` method should create a report with a total coverage percentage equal to the provided value.
- The test should fail if the coverage percentage is not included in the report.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 93 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, 312, 314, 330) |

PASSED tests/test_report_writer.py::TestReportWriter::test_write_report_includes_source_coverage 5ms 🛡️ 4

AI ASSESSMENT

**Scenario:** The test verifies that the `ReportWriter` class correctly writes a report with source coverage information.

**Why Needed:** This test prevents regression in reporting source code coverage, ensuring that reports accurately include this critical metric.

**Key Assertions:**

- source_coverage is an instance of SourceCoverageEntry with the specified file path and coverage statistics.
- report.source_coverage contains exactly one `SourceCoverageEntry` object.
- The first `SourceCoverageEntry` in `report.source_coverage` has a `file_path` attribute equal to 'src/foo.py'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 92 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, 312, 314, 330) |

PASSED `tests/test_report_writer.py::TestReportWriter::test_write_report_merges_coverage` 5ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test Report Writer should merge coverage into tests.

**Why Needed:** This test prevents a regression where the coverage is not merged correctly into the reports.

**Key Assertions:**

- The report should contain only one coverage entry for each test.
- The file path of the first coverage entry should match the file path of the test.
- All lines in the coverage entry should be present in the test's line ranges.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 94 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, 312, 314-315, 317-318, 330) |

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_atomic_write_fallback` 6ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test that the ReportWriter falls back to direct write if atomic write fails.

**Why Needed:** To prevent a regression where the atomic write operation fails and the direct write is used instead, causing warnings with code W203.

**Key Assertions:**

- The report file should exist at the specified path.
- All warnings in the report should have a code of W203.

COVERAGE

| | |
|---|---|
| `src/pytest_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` | 67 lines (ranges: 229-231, 233, 235, 364-380, 382-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510-512, 514, 516, 518, 520, 522) |
| `src/pytest_llm_report/options.py` | 2 lines (ranges: 107, 147) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/report_writer.py` | 125 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, 312, 314, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506-507, 509-512, 515-516) |

PASSED tests/test_report_writer.py::TestReportWriterWithFiles::test_creates_directory_if_missing 7ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the 'ReportWriter' class creates a directory if it does not exist.

**Why Needed:** This test prevents a regression where the report writer fails to create an output directory when the input JSON file does not exist.

**Key Assertions:**

- The 'report.json' file should be created in the specified directory.
- The 'ReportWriter' class should raise an exception if the input JSON file does not exist.
- The 'tmp_path / subdir / report.json' path should exist after calling 'writer.write_report(tests)'.

COVERAGE

| | |
|---|---|
| src/pytest_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 | 84 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 229-231, 233, 235, 364-380, 382-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510-512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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, 222-223, 226, 230, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 312, 314-315, 317-318, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-477, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_ensure_dir_failure` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test verifies that the test_ensure_dir_failure scenario is executed.

**Why Needed:** This test prevents a potential bug where the report writer fails to create a directory even if it's permission denied.

**Key Assertions:**

- The function `writer._ensure_dir(json_path)` should raise an exception when creating the directory.
- The function `writer.warnings` should contain at least one warning with code 'W201' (PermissionError).
- The function `writer.warnings` should not be empty after raising the exception.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 12 lines (ranges: 156-158, 470-473, 480-484) |

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_git_info_failure` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test 'test_git_info_failure' verifies that the `get_git_info` function handles git command failures by returning `None` for both SHA and dirty flags.

**Why Needed:** This test prevents a regression where the `get_git_info` function fails to return expected values when Git is not found, potentially causing downstream tests to fail or produce incorrect results.

**Key Assertions:**

- The `get_git_info` function should return `None` for both SHA and dirty flags when Git is not found.
- The `sha` variable should be set to `None` after calling `get_git_info()`
- The `dirty` variable should also be set to `None` after calling `get_git_info()`
- The function should raise an exception (e.g., `SystemExit`) when Git is not found, instead of returning a default value.
- The function should handle the case where Git is installed but not executable (i.e., `git --version` returns a non-zero exit code).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 9 lines (ranges: 67-73, 85-86) |

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_write_html_creates_file` 31ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test 'Should create HTML file' verifies that the report writer creates a new HTML file with expected content.

**Why Needed:** This test prevents regression where the report writer fails to create an HTML file even when there are tests that fail.

**Key Assertions:**

- The report.html file should exist in the specified path.
- The report.html file should contain the expected content (test1, test2, PASSED, FAILED, Skipped, XFailed, XPassed, Errors).
- All lines containing 'testX' should be present in the HTML string.
- Each line should start with either 'PASSED', 'FAILED', 'Skipped', or 'XFailed'.
- The report writer should not fail to create an HTML file even if there are tests that fail.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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 | 115 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, 312, 314-315, 317-320, 330, 376, 378-379, 382, 385, 388, 391-395, 470-471, 495, 497, 499-501, 503, 506) |

PASSED tests/test_report_writer.py::TestReportWriterWithFiles::test_write_html_includes_xfail_summary 32ms 🛡️ 5

AI ASSESSMENT

**Scenario:** The test verifies that the report includes xfail outcomes in the HTML summary.

**Why Needed:** This test prevents regression by ensuring that xfail outcomes are included in the report.

**Key Assertions:**

- The 'XFAILED' and 'XPASSED' tags should be present in the HTML summary.
- The 'xfailed' and 'xpassed' keywords should be found in the HTML content.
- The 'report.html' file should contain both 'XFAILED' and 'XPASSED' tags.
- The 'report.html' file should not contain any other xfail-related information (e.g., 'xfailed', 'xpassed')
- The report summary should include all relevant xfail outcomes (i.e., 'XFAILED' and 'XPASSED')
- The HTML content of the report should be able to be parsed by a tool like xmllint

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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, 312, 314-315, 317, 319, 321, 323-326, 330, 376, 378-379, 382, 385, 388, 391-395, 470-471, 495, 497, 499-501, 503, 506) |

PASSED tests/test_report_writer.py::TestReportWriterWithFiles::test_write_json_creates_file 6ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test verifies that a JSON file is created with the report.

**Why Needed:** This test prevents regression where the report writer does not create a JSON file.

**Key Assertions:**

- The `report.json` file should be created in the specified path.
- At least one artifact should be tracked for the report.
- The length of the artifacts list should be greater than or equal to 1.

COVERAGE

| | |
|---|---|
| src/pytest_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 | 78 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 364-380, 382-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 117 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, 312, 314-315, 317-318, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_write_pdf_creates_file` 34ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Test verifies that the `write_pdf` method creates a PDF file when Playwright is available.

**Why Needed:** This test prevents regression where the `write_pdf` method does not create a PDF file even if Playwright is available.

**Key Assertions:**

- The `report.pdf` path should be created and exist.
- All artifacts in the report should have paths matching the `report.pdf` path.
- Any generated PDF files should have a `.pdf` extension.
- The `report.pdf` path should not be empty after the test is run.
- The `writer.artifacts` list should contain at least one artifact with a path matching the `report.pdf` path.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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 | 125 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, 312, 314-315, 317-318, 330, 401, 410, 412, 414-423, 434-435, 437-443, 448, 453, 455, 458-462, 470-471) |

PASSED `tests/test_report_writer.py::TestReportWriterWithFiles::test_write_pdf_missing_playwright_warns` 6ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test should warn when Playwright is missing for PDF output.

**Why Needed:** To prevent a potential issue where the report writer does not generate a warning when Playwright is not found for PDF output.

**Key Assertions:**

- The file 'report.pdf' should exist.
- Any warnings generated by the report writer should have code WarningCode.W204_PDF_PLAYWRIGHT_MISSING.value.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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, 230-231, 233, 254, 256-259, 262-264, 266, 268-275, 277-278, 280-289, 291-294, 296-297, 299-300, 312, 314-315, 317-318, 330, 401-405, 408) |

PASSED `tests/test_report_writer_coverage_v2.py::test_report_writer_ensure_dir_creation` 1ms 🛡️ 4

AI ASSESSMENT

**Scenario:** Test ensures directory creation of report writer output.

**Why Needed:** Prevents a potential issue where the report writer creates an empty directory.

**Key Assertions:**

- The `tmp_dir / 'r.html'` path should exist before attempting to write to it.
- Any warnings from the report writer should be equal to 'W202'.
- After writing, the `tmp_dir / 'r.html'` path should no longer exist.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/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: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 11 lines (ranges: 156-158, 470-477) |

PASSED `tests/test_report_writer_coverage_v2.py::test_report_writer_metadata_skips` 10ms 🛡️ 5

AI ASSESSMENT

**Scenario:** Tests the scenario where report_writer_metadata_skips verifies that metadata skips when reports are disabled.

**Why Needed:** This test prevents regression because it ensures that metadata is always included in report writers, even when reports are disabled.

**Key Assertions:**

- The 'start_time' key should be present in the metadata.
- Metadata should contain a value for 'llm_model'.
- Metadata should not contain a value for 'report_html'.

COVERAGE

| | |
|---|---|
| src/pytest_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 | 36 lines (ranges: 364-380, 382-393, 395, 397, 399, 401, 403, 407, 419) |
| src/pytest_llm_report/options.py | 2 lines (ranges: 107, 147) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/report_writer.py | 67 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) |

PASSED tests/test_schemas.py::TestAnnotationSchema::test_from_dict_full 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that `AnnotationSchema.from_dict` can create a valid annotation from a dictionary with all required fields.

**Why Needed:** Prevents regression in case of missing or incorrect field definitions.

**Key Assertions:**

- assert schema.scenario == 'Verify login'
- assert schema.why_needed == 'Catch auth bugs'
- assert schema.key_assertions == ['assert 200', 'assert token']
- assert schema.confidence == 0.95

COVERAGE

| | |
|---|---|
| src/pytest_llm_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: 380-381, 384, 388-390) |

PASSED `tests/test_schemas.py::TestAnnotationSchema::test_to_dict_full` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** test_to_dict_full: Verify the conversion of an annotation to a dictionary with all required fields.

**Why Needed:** This test prevents regression in cases where authentication is not properly handled by the application, as it ensures that the 'why_needed' field is always present and contains relevant information.

**Key Assertions:**

- assert data['scenario'] == 'Verify login',
- assert data['why_needed'] == 'Catch auth bugs',
- assert data['key_assertions'] == ['assert 200', 'assert token'],
- assert data['confidence'] == 0.95

COVERAGE

| | |
|---|---|
| `src/pytest_llm_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: 380-381, 384, 388-390) |

PASSED tests/test_smoke_pytester.py::TestBasicReportGeneration::test_html_report_created 83ms 🛡️ 7

AI ASSESSMENT

**Scenario:** The HTML report is generated correctly and exists as expected.

**Why Needed:** This test prevents a bug where the report does not exist or contains incorrect information.

**Key Assertions:**

- The report path exists at the specified location.
- The content of the report includes the string ''.
- The string 'test_simple' is included in the report content.

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: 139-142) |
| src/pytest_llm_report/options.py | 46 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270-272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 166 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |

| | |
|---|---|
| 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 | 101 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, 312, 314-315, 317-318, 330, 376, 378-379, 382, 385, 388, 391-395, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_smoke_pytester.py::TestBasicReportGeneration::test_html_summary_counts_all_statuses` 121ms 🛡️ 7

AI ASSESSMENT

**Scenario:** test_html_summary_counts_all_statuses verifies that HTML summary counts include all statuses.

**Why Needed:** This test prevents regression where the report does not include all statuses, which could lead to incorrect analysis or reporting.

**Key Assertions:**

- The 'Total Tests' label should be present in the HTML summary.
- The 'Passed' label should have a count of 1 in the HTML summary.
- The 'Failed' label should have a count of 1 in the HTML summary.
- The 'Skipped' label should have a count of 1 in the HTML summary.
- The 'XFailed' label should have a count of 1 in the HTML summary.
- The 'XPassed' label should have a count of 1 in the HTML summary.
- The 'Errors' and 'Error' labels should be present in the HTML summary with correct counts.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 65 lines (ranges: 78-79, 90, 93-94, 96, 99-104, 106-107, 109-112, 114-119, 121-122, 124, 127, 132-133, 140, 155-159, 163, 167-169, 171, 181, 185-186, 198-199, 209-210, 212-214, 216, 227-228, 230-236, 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: 139-142) |
| src/pytest_llm_report/options.py | 46 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270-272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 166 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, |

| | |
|---|---|
| | 192, 195-196, 203, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| 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, 98-100, 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, 312, 314-315, 317-328, 330, 376, 378-379, 382, 385, 388, 391-395, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_smoke_pytester.py::TestBasicReportGeneration::test_json_report_created` 74ms 🛡️ 7

AI ASSESSMENT

**Scenario:** The JSON report is created successfully.

**Why Needed:** This test prevents a bug where the report generation fails due to missing configuration files or incorrect file paths.

**Key Assertions:**

- The `report_path` exists after running the test.
- The `data` dictionary in the report contains the expected schema version, summary statistics.
- The `summary` dictionary in the report has the correct total and passed/failed counts.
- The `passed` count is equal to 1 (i.e., one test passed) and the `failed` count is also 1 (i.e., one test failed).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 51 lines (ranges: 78-79, 90, 93-94, 96, 99-100, 104, 109-112, 114-118, 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: 139-142) |
| src/pytest_llm_report/models.py | 74 lines (ranges: 161-165, 167-169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 364-380, 382, 385, 387, 390-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 46 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270, 272-274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 166 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, |

| | |
|---|---|
| | 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| src/pytest_llm_report/report_writer.py | 107 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, 312, 314-315, 317-320, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED tests/test_smoke_pytester.py::TestBasicReportGeneration::test_llm_annotations_in_report 78ms 🛡️ 13

AI ASSESSMENT

**Scenario:** Verify that LLM annotations are included in the report generated by pytester.

**Why Needed:** This test prevents regressions caused when using a provider with LLM annotations in the report.

**Key Assertions:**

- asserts True
- asserts True
- asserts True

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: 139-142) |
| src/pytest_llm_report/llm/annotator.py | 69 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-68, 71-72, 74-78, 87-92, 97-98, 100, 102, 104, 115-122, 129-135, 137, 139, 165-168, 170-171, 173-174, 176, 178, 180, 185-190, 192-195, 198, 203) |
| src/pytest_llm_report/llm/base.py | 39 lines (ranges: 52-53, 72, 75, 80, 107, 110-111, 128, 136, 147, 165, 167, 175, 186-187, 190-191, 194-195, 198-200, 203, 205, 207, 212, 214-218, 245, 247, 249, 252, 257-258, 260) |
| src/pytest_llm_report/llm/litellm_provider.py | 23 lines (ranges: 37-38, 44, 46, 49, 51-52, 54-60, 62-63, 66-67, 69-70, 94-95, 97) |
| src/pytest_llm_report/llm/schemas.py | 7 lines (ranges: 38, 42-43, 50-53) |

| | |
|---|---|
| src/pytest_llm_report/models.py | 94 lines (ranges: 104-107, 109-111, 113, 115, 161-165, 167, 169, 171, 173, 176, 178-180, 182, 184, 186, 188, 190, 364-380, 382, 385, 387, 390-393, 395, 397, 399, 401, 403, 407-419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 47 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-259, 261, 263-265, 270, 272-274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 186 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203-205, 207-208, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-333, 336, 338, 341-345, 348, 350-355, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| 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 | 105 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, 291-296, 298-299, 312, 314-315, 317-318, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_smoke_pytester.py::TestBasicReportGeneration::test_llm_error_is_reported` 90.08s 🛡️ 12

AI ASSESSMENT

**Scenario:** Test that LLM errors are surfaced in HTML output.

**Why Needed:** To prevent regression where LLM errors are not reported correctly.

**Key Assertions:**

- The test verifies that the 'LLM error' is present in the report.
- The test verifies that the string 'boom' is also present in the report.
- The test checks for the correct HTML output format to ensure it matches expectations.
- The test ensures that LLM errors are surfaced correctly and reported in the expected location.
- The test verifies that the error message is displayed as intended, including the exact error string 'boom'.
- The test checks if the report contains any additional information related to the LLM error, such as a stack trace or more detailed error details.

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: 139-142) |
| src/pytest_llm_report/llm/annotator.py | 73 lines (ranges: 45, 48-49, 56-57, 59, 61, 64, 66-68, 71-72, 74-78, 87-92, 97-98, 100, 102, 104, 115-122, 129-135, 137-139, 165-168, 170-171, 173-174, 176, 178, 180, 185-190, 192-195, 198-201, 203) |
| src/pytest_llm_report/llm/base.py | 21 lines (ranges: 52-53, 72, 75, 80, 107, 110-111, 128, 136, 147, 165, 167, 175, 245, 247, 249, 252, 257-258, 260) |
| src/pytest_llm_report/llm/litellm_provider.py | 25 lines (ranges: 37-38, 44, 46, 49, 51-52, 54-60, 62-63, 78-79, 81-82, 84-85, 94-95, 97) |

| | |
|---|---|
| src/pytest_llm_report/options.py | 47 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-259, 261, 263-265, 270-272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 186 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203-205, 207-208, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-333, 336, 338, 341-346, 350-355, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| 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 | 101 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, 291-296, 298-299, 312, 314-315, 317-318, 330, 376, 378-379, 382, 385, 388, 391-395, 470-471, 495, 497, 499-501, 503, 506) |

PASSED tests/test_smoke_pytester.py::TestMarkers::test_llm_opt_out_marker 57ms 🛡️ 7

AI ASSESSMENT

**Scenario:** Test the LLM opt-out marker.

**Why Needed:** Prevents regression in LLM opt-out functionality.

**Key Assertions:**

- The test verifies that the LLM opt-out marker is recorded.
- The test asserts that the LLM opt-out marker is set to True for a single test.
- The test checks if the 'llm_opt_out' key exists in the report data and its value is True.

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: 139-142) |
| `src/pytest_llm_report/models.py` | 74 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180-182, 184, 186, 188, 190, 364-380, 382, 385, 387, 390-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| `src/pytest_llm_report/options.py` | 46 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270, 272-274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| `src/pytest_llm_report/plugin.py` | 166 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, |

| | |
|---|---|
| | 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| src/pytest_llm_report/report_writer.py | 105 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, 312, 314-315, 317-318, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED tests/test_smoke_pytester.py::TestMarkers::test_requirement_marker 56ms 🛡️ 7

AI ASSESSMENT

**Scenario:** Test the requirement marker functionality.

**Why Needed:** This test prevents a potential regression where the requirement marker is not recorded correctly, potentially leading to incorrect test results or missed tests.

**Key Assertions:**

- The `pytest.mark.requirement` decorator should be applied to a function with at least one requirement.
- The `requirements` key in the test data should contain both 'REQ-001' and 'REQ-002'.
- The `reqs` list in the test data should contain both 'REQ-001' and 'REQ-002'.

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: 139-142) |
| src/pytest_llm_report/models.py | 74 lines (ranges: 161-165, 167, 169, 171, 173, 176, 178, 180, 182, 184, 186, 188-190, 364-380, 382, 385, 387, 390-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 46 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270, 272-274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 166 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, |

| | |
|---|---|
| | 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| src/pytest_llm_report/report_writer.py | 105 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, 312, 314-315, 317-318, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_smoke_pytester.py::TestOutcomes::test_multiple_xfail_outcomes` 65ms 🛡️ 7

AI ASSESSMENT

**Scenario:** The test verifies that multiple xfailed tests are recorded in the report.

**Why Needed:** This test prevents regression where a single xfailed test is reported as successful.

**Key Assertions:**

- The number of xfailed tests should be equal to 2.
- All xfailed tests should have an outcome of 'xfailed'.
- Each xfailed test should have an outcome that matches the previous one.
- No other outcomes should be reported for any test.

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: 139-142) |
| src/pytest_llm_report/models.py | 74 lines (ranges: 161-165, 167-169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 364-380, 382, 385, 387, 390-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 46 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270, 272-274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 166 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, |

|  |  |
|---|---|
|  | 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| src/pytest_llm_report/report_writer.py | 108 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, 312, 314-315, 317, 319, 321, 323-324, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_smoke_pytester.py::TestOutcomes::test_skip_outcome` 57ms 🛡️ 7

AI ASSESSMENT

**Scenario:** Test that skipped tests are correctly recorded in the report.

**Why Needed:** This test prevents a regression where skipped tests are not properly reported in the report.

**Key Assertions:**

- The number of skipped tests is correctly counted and reported in the 'skipped' section of the report.
- The 'skipped' section contains the correct count of skipped tests.
- The 'summary' section of the report includes the correct information about skipped tests.
- The test data is loaded from the correct file path.
- The test data is correctly parsed as JSON and can be accessed using `json.loads()`.
- The assertion checks if the 'skipped' count matches the expected value (1 in this case).
- The test does not skip any tests, ensuring that all tests are run.

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: 139-142) |
| src/pytest_llm_report/models.py | 74 lines (ranges: 161-165, 167-169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 364-380, 382, 385, 387, 390-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 46 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270, 272-274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 166 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, |

| | |
|---|---|
| | 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| `src/pytest_llm_report/report_writer.py` | 107 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, 312, 314-315, 317, 319, 321-322, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED tests/test_smoke_pytester.py::TestOutcomes::test_xfail_outcome 59ms 🛡️ 7

AI ASSESSMENT

**Scenario:** Verifies that the test 'test_xfail' is marked as Xfailed and has a count of 1 in the report.

**Why Needed:** This test prevents regression by ensuring that tests marked as Xfailed are correctly recorded in the report.

**Key Assertions:**

- The 'summary' key under 'xfailed' should contain a single integer value, indicating that there is exactly one failed test.
- The 'xfailed' key should have a numeric value of 1, indicating that there is only one Xfailed test.
- The total count of all tests in the report should not exceed 10.
- The number of Xfailed tests should be greater than or equal to 0.
- The 'summary' key under 'xfailed' should contain a single integer value, indicating that there is exactly one failed test.
- The 'xfailed' key should have a numeric value of 1, indicating that there is only one Xfailed test.
- The total count of all tests in the report should not exceed 10.
- The number of Xfailed tests should be greater than or equal to 0.

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: 139-142) |
| src/pytest_llm_report/models.py | 74 lines (ranges: 161-165, 167-169, 171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 364-380, 382, 385, 387, 390-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 46 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, |

| | |
|---|---|
| | 270, 272-274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 166 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| src/pytest_llm_report/report_writer.py | 108 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, 312, 314-315, 317, 319, 321, 323-324, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED tests/test_smoke_pytester.py::TestParametrization::test_parametrized_tests 59ms 🛡️ 7

AI ASSESSMENT

**Scenario:** Test parameterized tests are recorded separately.

**Why Needed:** This test prevents regression by ensuring that the same set of parameters is used for each test run, avoiding potential inconsistencies or bugs that may arise from different inputs.

**Key Assertions:**

- The total number of tests should be equal to 3.
- All tests passed with a status code of 'passed'.
- Each test has been executed exactly once.

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: 139-142) |
| src/pytest_llm_report/models.py | 74 lines (ranges: 161-165, 167, 169-171, 173, 176, 178, 180, 182, 184, 186, 188, 190, 364-380, 382, 385, 387, 390-393, 395, 397, 399, 401, 403, 407, 419, 449-457, 459, 461, 500, 502-506, 508, 510, 512, 514, 516, 518, 520, 522) |
| src/pytest_llm_report/options.py | 46 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270, 272-274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 166 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, |

| | |
|---|---|
| | 192, 195-196, 203, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| src/pytest_llm_report/report_writer.py | 105 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, 312, 314-315, 317-318, 330, 340, 343-345, 348-349, 352-354, 357, 360-364, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_smoke_pytester.py::TestPluginRegistration::test_help_contains_examples` 51ms 🛡️ 3

AI ASSESSMENT

**Scenario:** The test verifies that the CLI help text contains usage examples.

**Why Needed:** This test prevents a potential bug where the help text is missing or incomplete, making it difficult for users to understand how to use the plugin.

**Key Assertions:**

- The output of `pytester.runpytest('--help')` should contain at least one line that matches 'Example:*--llm-report*'.
- The output should not contain any lines that do not match 'Example:*--llm-report*', but rather include the examples.
- The usage examples should be clear and descriptive, including '--llm-report' as an option.
- The test should pass if the CLI help text is properly formatted with the correct examples.
- If the output does not contain any matching lines, the test should fail.
- The test should only fail if the output contains more than one line that matches 'Example:*--llm-report*'.
- The test should only succeed if all lines match 'Example:*--llm-report*' and no other lines are present.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/options.py` | 45 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270, 272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| `src/pytest_llm_report/plugin.py` | 118 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, 380-381, 384, 388-390) |

PASSED tests/test_smoke_pytester.py::TestPluginRegistration::test_markers_registered 45ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that LLM markers are registered and correctly displayed in the pytest output.

**Why Needed:** This test prevents a potential bug where LLM markers are not properly registered or are incorrectly displayed in the pytest output.

**Key Assertions:**

- The 'llm_opt_out' marker is present in the pytest output.
- The 'llm_context' marker is present in the pytest output.
- The 'requirement' marker is present in the pytest output.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 45 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270, 272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 118 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, 380-381, 384, 388-390) |

PASSED `tests/test_smoke_pytester.py::TestPluginRegistration::test_plugin_registered` 52ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test that the plugin is registered correctly and can be accessed via pytest11.

**Why Needed:** This test prevents a potential issue where the plugin's registration might not be properly detected or reported by pytest11.

**Key Assertions:**

- The `pytester.runpytest()` method should return an instance of `fnmatch_lines` with the expected lines.
- The `stdout.fnmatch_lines` call should match the expected output.
- The plugin's registration should be correctly detected and reported via `--llm-report` flag.
- The `result.stdout` attribute should contain the expected output.
- The `pytester.runpytest()` method should return a non-empty result object.
- The `result.stdout.fnmatch_lines` call should not raise an exception.
- The `result.stdout` attribute should contain the expected lines after running pytest11.
- The `--llm-report` flag should be present in the output of pytest11.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/options.py | 45 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270, 272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 118 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, 380-381, 384, 388-390) |

PASSED `tests/test_smoke_pytester.py::TestSpecialCharacters::test_special_chars_in_nodeid` 84ms 🛡️ 7

AI ASSESSMENT

**Scenario:** Verify that special characters in nodeid are handled correctly by pytester.

**Why Needed:** This test prevents a potential crash and ensures the HTML report is valid.

**Key Assertions:**

- The 'report.html' file should exist after running pytester with --llm-report.
- The '' tag should be present in the contents of the 'report.html' file.

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: 139-142) |
| src/pytest_llm_report/options.py | 46 lines (ranges: 107, 147, 175, 178-179, 185-186, 193-194, 201-202, 209, 211, 213, 215, 217, 220, 224, 248, 251-253, 255-259, 261, 263-265, 270-272, 274, 276, 278, 280, 282, 286, 288, 290, 292, 294, 298, 300) |
| src/pytest_llm_report/plugin.py | 166 lines (ranges: 40, 43-47, 49-53, 55-59, 61-65, 67-71, 73-78, 80-85, 89-93, 95-99, 101-105, 107-111, 113-117, 121-124, 126-129, 131-134, 136-140, 142-145, 147-151, 153-156, 169-171, 173-175, 177-179, 183, 187-188, 190, 192, 195-196, 203, 212-213, 238, 242, 246, 249, 268-269, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-303, 324, 330-331, 358-368, 380-381, 384, 388-390, 401, 405, 424, 428-430, 441, 445, 448, 450-451) |
| 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 | 101 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, 312, 314-315, 317-318, 330, 376, 378-379, 382, 385, 388, 391-395, 470-471, 495, 497, 499-501, 503, 506) |

PASSED `tests/test_time.py::TestFormatDuration::test_boundary_one_minute` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Tests the format of a duration that is exactly one minute.

**Why Needed:** This test prevents regressions where durations are formatted as '1m 0.0s' when they should be '1m 00.0s'.

**Key Assertions:**

- The result of `format_duration(60.0)` should be exactly '1m 00.0s'.
- The '+' operator should not be used to concatenate strings.
- The format specifier `%M' should be used instead of '%m'.
- The format specifier `%S' should be used instead of '%s'.
- The result of `format_duration(60)` without any arguments should also be '1m 00.0s'.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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 the `format_duration` function with a duration of 500 microseconds.

**Why Needed:** Prevents regression where durations are incorrectly reported as milliseconds instead of microseconds.

**Key Assertions:**

- The result should contain 'µs' to indicate microsecond format.
- The duration value should be exactly 500 microseconds.
- The function name `format_duration` is used correctly.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/util/time.py | 2 lines (ranges: 39-40) |

PASSED `tests/test_time.py::TestFormatDuration::test_milliseconds_format` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Tests the `format_duration` function to ensure it correctly formats sub-second durations as milliseconds.

**Why Needed:** This test prevents a regression where the function incorrectly returns 'ms' instead of 'ms.' when given a duration in seconds.

**Key Assertions:**

- The function should return '500.0ms' for a duration of 0.5 seconds.
- The function should correctly handle durations greater than or equal to one second.
- The function should not incorrectly append an extra digit ('ms.') when the input is in seconds.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `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:** Test the 'minutes' format for durations over a minute.

**Why Needed:** Prevents regression where durations are incorrectly formatted as 'seconds'.

**Key Assertions:**

- The function should return a string with 'm' followed by '1' and then 'm' again, indicating one minute.
- The function should return the exact same string as '1m 30.5s' when given the duration 90.5 minutes.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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 the `format_duration` function with a scenario that involves formatting multiple minutes.

**Why Needed:** This test prevents regression in cases where the input duration is greater than or equal to one minute.

**Key Assertions:**

- The output of `format_duration(185.0)` should be '3m 5.0s'.
- The total number of seconds in the input duration should be calculated correctly.
- The minutes and seconds parts of the output should be separated correctly (e.g., '3m' for three minutes).
- Any trailing zeros in the seconds part should be preserved.
- The function should handle durations greater than or equal to one minute correctly.
- The function should not silently truncate any digits when they are too small to be represented as a digit.
- The function should preserve leading zeros in the output (e.g., '000m' for one minute).

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `src/pytest_llm_report/util/time.py` | `6 lines (ranges: 39, 41, 43, 46-48)` |

PASSED `tests/test_time.py::TestFormatDuration::test_one_second` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Verifies the correct formatting of a duration equal to one second.

**Why Needed:** Prevents incorrect formatting of durations greater than one second, potentially leading to inaccurate time representations in various applications.

**Key Assertions:**

- The function `format_duration(1.0)` returns '1.00s' for input 1.0.
- The function `format_duration(2.0)` returns '2.00s' for input 2.0.
- The function `format_duration(3.0)` returns '3.00s' for input 3.0.
- The function `format_duration(4.0)` returns '4.00s' for input 4.0.
- The function `format_duration(5.0)` returns '5.00s' for input 5.0.
- The function `format_duration(-1.0)` raises a ValueError, as it is not possible to format negative durations.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** Verifies the 'seconds' unit of duration.

**Why Needed:** Prevents incorrect formatting when seconds are less than a minute.

**Key Assertions:**

- The function should return 's' as the unit of duration for seconds.
- The result should be equal to '5.50s' after conversion.
- The assertion should fail if the input is not an integer or float between 0 and 59.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** Test that the function formats small millisecond durations correctly.

**Why Needed:** This test prevents a potential bug where the function incorrectly returns '1.0ms' for very short durations (e.g., 0.001 seconds).

**Key Assertions:**

- The function should return the correct duration value (e.g., '1.0ms').
- The duration value should be accurate to at least two decimal places.
- Any non-numeric characters in the output string should be ignored.
- The function should handle durations of exactly 1 millisecond correctly.
- The function should return an error message for invalid input (e.g., negative numbers).
- The function should not silently truncate the duration value.
- The function should raise a ValueError for extremely short durations (e.g., 0.00001 seconds).

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** Verifies that the `format_duration` function correctly formats very small durations as microseconds.

**Why Needed:** This test prevents a potential bug where the function incorrectly returns '0' for very small durations (e.g., 1 microsecond).

**Key Assertions:**

- The function should return the correct string representation of the duration in microseconds.
- The function should handle very small durations correctly and not return an incorrect value.
- The function should be able to handle negative durations without returning an error.
- The function should support all possible input values (0-9999999999 microseconds).
- The function should maintain its precision for very small durations.
- The function should not silently truncate or round the result for very small durations.
- The function should return a string in the correct format (e.g., '1µs', '1ms', etc.).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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 the ISO format of a datetime object with UTC timezone.

**Why Needed:** This test prevents a potential bug where the ISO format is incorrectly formatted when the datetime object has an offset from UTC.

**Key Assertions:**

- The result should be in the expected format 'YYYY-MM-DDTHH:MM:SS+HH:MM:SS' (UTC timezone).
- The offset of the datetime object should be correctly converted to UTC timezone.
- The resulting string should not have any leading or trailing whitespace.
- The resulting string should not contain any non-ASCII characters.
- The resulting string should start with 'YYYY-MM-DDTHH:MM:SS'.
- The resulting string should end with '+HH:MM:SS'.
- The resulting string should be in the UTC timezone.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** Test the naive datetime format without timezone.

**Why Needed:** Prevents a potential bug where naive datetime formats are not correctly handled.

**Key Assertions:**

- The output of `iso_format(dt)` should be '2024-06-20T14:00:00'.
- The `datetime` object passed to `iso_format()` is in the naive timezone.
- The `iso_format()` function does not throw an error for invalid datetime inputs.
- The `iso_format()` function correctly formats a naive datetime with no timezone.
- The `iso_format()` function preserves the original timezone information of the input datetime.
- The `iso_format()` function handles ambiguous datetime inputs (e.g., 2024-06-20T14:00) correctly.
- The `iso_format()` function does not throw an error for invalid datetime formats (e.g., 2024/6/20T14:00)
- The `iso_format()` function preserves the original timezone information of the input datetime even if it is ambiguous.

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| `src/pytest_llm_report/plugin.py` | 6 lines (ranges: 380-381, 384, 388-390) |
| `src/pytest_llm_report/util/time.py` | 1 lines (ranges: 27) |

PASSED `tests/test_time.py::TestIsoFormat::test_formats_with_microseconds` 1ms 🛡️ 3

AI ASSESSMENT

**Scenario:** Test the `iso_format` function with a datetime object containing microseconds.

**Why Needed:** This test prevents regressions where the `iso_format` function returns an incorrect value when the input datetime contains microseconds.

**Key Assertions:**

- The result of `iso_format(dt)` should contain the string '123456'.
- The length of the result should be greater than or equal to 6 characters.
- The first character of the result should be a digit (1-9).
- The second character of the result should be a letter (a-z) or a number (0-9).
- The third character of the result should be a digit (1-9).
- The fourth character of the result should be a letter (a-z) or a number (0-9).
- The fifth character of the result should be a digit (1-9).
- The sixth character of the result should be a letter (a-z) or a number (0-9).

COVERAGE

| | |
|---|---|
| `src/pytest_llm_report/collector.py` | `14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210)` |
| `src/pytest_llm_report/plugin.py` | `6 lines (ranges: 380-381, 384, 388-390)` |
| `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:** Verifies that the `utc_now()` function returns a datetime object with a valid UTC timezone.

**Why Needed:** This test prevents regression when switching to a different time zone or region.

**Key Assertions:**

- The returned datetime object has a non-null `tzinfo` attribute and is equal to `UTC`.
- The `tzinfo` attribute of the returned datetime object is set to `UTC`.
- The returned datetime object does not have any other timezone information (e.g., `zone`, `offset`, etc.)
- The returned datetime object has a valid timezone identifier (in this case, `UTC`).

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** Verifies that the function `utc_now()` returns a time within a certain tolerance of the current UTC time.

**Why Needed:** This test prevents a potential issue where the `utc_now()` function may return an incorrect or outdated time due to clock skew.

**Key Assertions:**

- The result of `utc_now()` should be within a certain range (e.g. ±1 second) of the current UTC time.
- The difference between the before and after times should not exceed the tolerance specified in the test.
- The function should return an error or raise an exception if it is unable to determine the current UTC time due to clock skew.
- The function should use a suitable algorithm (e.g. polling, synchronization) to determine the current UTC time.
- The function should be able to handle cases where the system clock is not synchronized with the UTC time.
- The function should provide a meaningful error message or indication of failure if the current UTC time cannot be determined.
- The function should maintain consistency in its behavior across different systems and environments.

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| 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:** The `utc_now()` function should return a datetime object.

**Why Needed:** This test prevents a potential issue where the returned datetime is not in UTC.

**Key Assertions:**

- result instanceof Date (not datetime)
- result instanceof Object (not datetime)
- result.getTime() !== result.getUTCHours() || result.getTime() !== result.getUTCMinutes() || result.getTime() !== result.getUTCSeconds() || result.getTime() !== result.getUTCMilliseconds()
- result instanceof Object (not datetime)
- result instanceof Date (not datetime)
- result instanceof Object (not datetime)
- result instanceof Object (not datetime)
- result instanceof Object (not datetime)
- result instanceof Object (not datetime)

COVERAGE

| | |
|---|---|
| src/pytest_llm_report/collector.py | 14 lines (ranges: 90, 93, 96, 99, 110-112, 114-115, 124, 127, 140, 209-210) |
| src/pytest_llm_report/plugin.py | 6 lines (ranges: 380-381, 384, 388-390) |
| src/pytest_llm_report/util/time.py | 1 lines (ranges: 15) |

## 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 | 116 | 5 | 111 | 95.69% | 13, 15-19, 21, 35, 38, 44, 46, 52-53, 55-57, 59, 61-64, 69, 73-74, 77-80, 84, 87-89, 93, 103, 109-111, 113-117, 119-120, 125, 127-128, 130-131, 134-135, 141-144, 146, 148, 162, 164, 168, 170, 172, 182, 184-188, 190-191, 194, 196, 205, 217, 219-233, 235, 237, 245-246, 248-249, 251, 253-255, 259, 262-263, 265-266, 269, 271-272, 274, 276-277, 281 | 66, 90-91, 192, 203 |
| src/pytest_llm_report/cache.py | 47 | 3 | 44 | 93.62% | 13, 15-19, 21, 27, 33, 39-41, 43, 53, 55-56, 58, 60-62, 68-69, 78, 86, 88, 90, 92, 94, 97, 103, 107, 118-119, 121, 123, 129, 132-136, 141, 144, 153 | 64-65, 130 |
| src/pytest_llm_report/collector.py | 111 | 2 | 109 | 98.2% | 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, 143, 155, 163-164, 167-169, 171, 173, 181-182, 185-189, 191, 198-200, 202, 209-210, 212-214, 216, 218, 227-228, 230-236, 238, 241, 250-252, 254, 261, 264-265, 268-269, | 141, 239 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | 271, 277, 279, 285 | |
| src/pytest_llm_report/coverage_map.py | 135 | 10 | 125 | 92.59% | 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, 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, 246-248, 250, 252-254, 259-260, 263-264, 271, 273, 276-279, 281-283, 285, 299-300, 302, 308 | 62, 123, 125, 128, 157, 221, 249, 251, 257, 274 |
| src/pytest_llm_report/errors.py | 35 | 0 | 35 | 100.0% | 8-9, 12, 25-28, 31-36, 39-42, 45-46, 49-51, 54-55, 64-66, 68, 70, 74-76, 80, 129, 139 | - |
| src/pytest_llm_report/llm/__init__.py | 3 | 0 | 3 | 100.0% | 4-5, 7 | - |
| src/pytest_llm_report/llm/annotator.py | 110 | 0 | 110 | 100.0% | 4, 6-10, 12-15, 21-22, 25-28, 31, 45-46, 48-50, 54, 56-57, 59, 61-62, 64, 66-68, 71-72, 74-82, 87, 97-98, 100, 102, 104-105, 115, 127, 129-132, 137-139, 142, 165-168, 170-171, 176, 178, 180-183, 185-190, 192-193, 198-201, 203, 206, 229-232, | - |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | 234, 236-237, 239-240, 245-246, 248-253, 255-256, 261-264, 266 | |
| src/pytest_llm_report/llm/base.py | 78 | 0 | 78 | 100.0% | 13, 15-18, 26, 40, 46, 52-53, 55, 72, 75-76, 78, 80, 101, 107-108, 110-111, 122, 128, 130, 136, 138, 147, 149, 165, 167-173, 175, 177, 186-187, 190-192, 194-195, 198-200, 203-208, 212, 214, 220-221, 224-225, 228-230, 233, 245, 247, 249-250, 252-253, 255, 257-258, 260, 262-263, 265, 267 | - |
| src/pytest_llm_report/llm/gemini.py | 275 | 18 | 257 | 93.45% | 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, 87-88, 91-97, 99-103, 105, 107-114, 121-122, 125, 128, 134, 136-139, 141-142, 144, 160-161, 167-169, 171-172, 174, 176-184, 186-188, 190-191, 193, 196, 200-208, 210-211, 213-215, 217-223, 225-227, 233-234, 238-239, 242-243, 245-248, 252-253, 260, 266-267, 269, 273-277, 279-283, 286-287, 292-293, 300-301, 303, 315, 317-318, 322, 327, 330-332, 335-343, 345-346, 348, 352-355, 357, 360-366, 368-374, 380-382, 384-387, 389, 391-392, | 89, 104, 106, 115-117, 199, 230-231, 235-237, 244, 250, 256, 367, 441, 444 |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  |  |  |  |  | 396-402, 405, 408-410, 412-414, 416-421, 427-428, 430-434, 437-440, 442-443, 445-447 |  |
| src/pytest_llm_report/llm/litellm_provider.py | 32 | 1 | 31 | 96.88% | 7, 9, 11-12, 18, 21, 37-38, 44, 46, 49, 51-52, 54-56, 66-67, 69-70, 73, 76, 78-79, 81-82, 84, 88, 94-95, 97 | 74 |
| src/pytest_llm_report/llm/noop.py | 13 | 0 | 13 | 100.0% | 8, 10, 12-13, 20, 26, 32, 34, 50, 52, 58, 60, 66 | - |
| src/pytest_llm_report/llm/ollama.py | 43 | 1 | 42 | 97.67% | 7, 9, 11-12, 18, 24, 40-41, 47, 50, 52, 54-55, 57-60, 62-63, 66-67, 71-72, 74-75, 77, 81, 87-88, 90-92, 96, 102, 104, 114, 116-117, 127, 132, 134-135 | 69 |
| 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/models.py | 240 | 10 | 230 | 95.83% | 17-18, 21, 24-25, 34-36, 38, 40, 47-48, 61-67, 69, 71, 82-83, 95-100, 102, 104, 109-115, 118-119, 141-157, 159, 161, 167-171, 173-182, 184, 186, 188-190, 193-194, 202-203, 205, 207, 213-214, 223-225, 227, 229, 233-235, 238-239, 248-250, 252, 254, 261-262, 271-273, 275, 277, 281-283, 286-287, 324-353, | 172, 183, 185, 187, 460, 513, 515, 517, 519, 521 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | 355-360, 362, 364, 382-405, 407-419, 422-423, 437-445, 447, 449, 459, 461, 464-465, 482-492, 494, 500, 502, 508-512, 514, 516, 518, 520, 522 | |
| src/pytest_llm_report/options.py | 117 | 45 | 72 | 61.54% | 106, 146, 175, 178-180, 185-187, 193-195, 201-203, 209-218, 220, 224, 233, 248, 251-267, 270-283, 286-295, 298, 300 | 13-15, 21-22, 90-94, 97-99, 102-105, 122-123, 126-132, 135-137, 140-142, 145, 156-160, 163-164, 167, 169, 222, 227, 236 |
| src/pytest_llm_report/plugin.py | 151 | 24 | 127 | 84.11% | 40, 43, 49, 55, 61, 67, 73, 80, 89, 95, 101, 107, 113, 121, 126, 131, 136, 142, 147, 153, 169, 173, 177, 183-184, 187-188, 190, 192, 195-197, 203-204, 212-213, 238-239, 242-243, 246, 249-250, 252-253, 256-257, 259, 261-265, 268-269, 271, 273, 276-277, 280-281, 283-284, 287-291, 293, 296-297, 299, 302-305, 307, 309-312, 315-316, 324-325, 330-333, 336, 338, 341-346, 348, 350, 358-359, 380-381, 384-385, 388-390, 401-402, 405, 408-409, 412-414, 424-425, 428-430, 441-442, 445, 448, 450-451 | 13, 15-17, 19-20, 22, 28-31, 34, 160, 216, 320-321, 326-327, 372-373, 393, 417, 433-434 |

| | | | | | | |
|---|---|---|---|---|---|---|
| src/pytest_llm_report/prompts.py | 75 | 5 | 70 | 93.33% | 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, 116, 118, 132-133, 135-138, 140-141, 144-145, 148, 151-152, 154-156, 158-159, 163, 165, 180, 182, 191-194 | 80, 114, 142, 146, 149 |
| src/pytest_llm_report/render.py | 50 | 0 | 50 | 100.0% | 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, 141-143, 145, 158-163, 177, 196 | - |
| src/pytest_llm_report/report_writer.py | 167 | 10 | 157 | 94.01% | 13, 15-25, 27-29, 46, 55, 58, 67-68, 76, 83-84, 89, 98-100, 102, 105-108, 110, 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, 303, 312, 314-315, 317-328, 330, 332, 340, 343-345, 348-349, 352-354, 357, 360, 368, 376, 378-379, 382, 385, 388, 391, 399, 401-402, 408, 410, 412, 414-423, 434-435, 437-439, 447-448, 453, 455, 458, 461-462, 464, 470-474, 480-481, 488, 495, 497, | 113, 135-137, 424-425, 432, 449-451 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | 499-501, 503, 506-507, 509, 515-516 | |
| src/pytest_llm_report/util/fs.py | 34 | 3 | 31 | 91.18% | 11, 13-14, 17, 30, 33, 36, 39, 42, 45, 55-56, 58-60, 63-64, 70, 79, 82, 100, 103, 111-113, 116-117, 119-121, 123 | 40, 65, 67 |
| src/pytest_llm_report/util/hashing.py | 36 | 0 | 36 | 100.0% | 12, 14-17, 23, 32, 35, 44-48, 51, 61, 64, 73-74, 76-78, 80-81, 86, 96, 103-104, 107, 113-114, 116-121 | - |
| src/pytest_llm_report/util/ranges.py | 33 | 0 | 33 | 100.0% | 12, 15, 29-30, 33, 35-37, 39-40, 42, 45-47, 50, 52, 55, 65-67, 70, 81-82, 84-91, 93, 95 | - |
| src/pytest_llm_report/util/time.py | 16 | 0 | 16 | 100.0% | 4, 6, 9, 15, 18, 27, 30, 39-44, 46-48 | - |