SAMPLE reliability report on a DEMO system

System: offline refund-policy lookup agent with a mocked LLM.
Status: verified local test run, not a production assessment.
Generated: 2026-10-18T13:39:29.731814+00:00

Result

The unchanged three regression assertions failed against the intentionally buggy implementation and passed against the fixed implementation. The complete fixed suite passed. No model API, network connection or client system was used.

Run	Passed	Failed	Errors	Skipped	Exit code
Buggy baseline, three regressions	0	3	0	0	1
Fixed implementation, full suite	27	0	0	0	0

Reproduced failures and controls

Regression assertion	Buggy baseline	Fixed implementation
`test_malformed_tool_call_is_blocked`	Reproduced	Passed
`test_wrong_tool_is_blocked_before_dispatch`	Reproduced	Passed
`test_unsupported_claim_is_blocked`	Reproduced	Passed

1. **Malformed tool-call JSON:** the buggy agent raises a JSON decoding exception. The fixed agent returns a structured blocked result before any tool runs.
2. **Wrong tool:** a schema-valid `close_ticket` call is dispatched by the buggy agent during a read-only lookup. The fixed agent checks workflow permission after schema validation and before dispatch, so no ticket is closed.
3. **Unsupported claim:** the buggy agent repeats a fabricated 90-day refund policy. The fixed agent checks each claim against its retrieved source and releases no answer when the evidence does not match.

The full suite also checks missing citations, successful cited answers, invalid argument types, extra fields, duplicate JSON keys, unknown tools, mixed supported and unsupported claims, and requests outside this demo's scope.

Demo method and run instructions

The same regression test file runs against both versions. Only the implementation selection changes. The default `python -m pytest` run selects the fixed version. The baseline is intentionally expected to exit with code 1. Fixes block unsafe outputs; they do not repair a model response or retry automatically.

This public sample summarises the recorded demo run. Raw execution logs, evidence files and source hashes are not included in this public copy.

In the companion demo project, Python 3.10 or newer and pytest are required. From its project root, `python demo/run_demo.py` runs the three expected baseline failures and the complete fixed suite, then generates the report. The default `python -m pytest` selects the fixed version. The demo code and dependency files are not distributed in this public sample.

The results above describe the recorded 18 October 2026 run. They were not rerun as part of this website merge. The baseline and full fixed suite have different sizes.

Limits and remaining risks

This is a SAMPLE on a DEMO system, not client work, a security audit or a guarantee of production reliability. Mock outputs are deterministic. The tool sources are trusted local fixtures. Only one exact refund-policy request is supported, and the mock search returns a fixed source. Citation checking accepts exact source text only, so it blocks paraphrases as well as unsupported claims. This is not a general semantic entailment checker. It does not establish source truthfulness.

There is no live model variability, prompt-injection assessment, load test, authentication, deployment, production write tool or ongoing monitoring. A real sprint would agree representative failure cases, source trust, tool permissions and acceptance criteria for the client's workflow before implementation.

Recommended next step

Use these controls as a small starting example. For a real workflow, add cases from observed failures and verify each permitted tool and evidence boundary in the target environment. Further work needs its own agreed scope.

Prepared for demonstration only. Read the execution status and limitations before using this report.