PULL-PANDA BlackBox Test Report
Covering Versions 1.2 & RAG 1.3

Report Overview

This document serves as a for the Black-Box testing of the PULL-PANDA system. It aggregates findings from two major testing phases:

1. **RAG Version 1.3 :** Validating the RAG integration and Static Analysis engine.
2. **Version 1.2 :** Validating system basic functioning and AI reasoning capabilities.

Part I: RAG Version 1.3 (RAG + Static Analysis)

Version: RAG 1.3
Test Repo: blackbox_test_v1.3_RAG

1 Summary (v1.3)

This phase tested the integration of retrieval-augmented generation and automated code linting. The test involved a batch of 18 automated Pull Requests.
Key Findings:

- **RAG Integration: FUNCTIONAL**. The system successfully ingested `coding_standards.md` and repository files, correctly citing rule violations in reviews.
- **Static Analysis: NON-FUNCTIONAL**. The module failed across all 18 test cases due to a configuration error ("File Not Found").
- **Resolution:** The Static Analysis defect was patched and verified *after* this test cycle concluded.

2 Findings (RAGv1.3)

2.1 Finding #1: RAG System Accuracy

The RAG pipeline demonstrated 100% effectiveness in identifying specific compliance violations that required external knowledge (i.e., rules not inherent to the code logic itself).

Evidence from Generated Reviews:

Rule Violated	Affected PRs	AI Response Evidence
”All functions must have type hints.”	PR #4, PR #14	”Critical Bugs: 2. Missing type hints: Multiple functions... are missing type hints” (from `review_pr4_Meta.md`).
”All public functions must have a docstring.”	PR #5	”Critical Bugs: 2. Incomplete function implementations: ... ’get_initials’ lacks a docstring.”
”Use `black` for formatting.”	PR #1	”Code Quality: 1. Run ’black’ for formatting...”
”Do not hardcode user roles.”	PR #5, PR #18	”Code Quality: 2. Avoid hard-coded values: The ’is_admin’ function hardcodes the ’admin’ string.”

2.2 Finding #2: Context Boundary Verification

The test confirmed that the AI did not hallucinate rules.

- **Scenario:** PRs deliberately missing API timeouts (e.g., PR #3).
- **Observation:** The AI correctly identified this as a general best practice improvement but did **not** cite it as a violation of ”Coding Standards,” because the timeout rule was excluded from the knowledge base for this specific test.

2.3 Finding #3: Static Analysis Defect

Status: CRITICAL FAIL
The `static_analysis.py` module consistently failed to execute linter commands.

- **Error Signature:**

```
| Pylint: F0001: No module named [filename]
| Flake8: E902 FileNotFoundError: [Errno 2] No such file or directory
```

- **Diagnosis:** The script was attempting to run tools on file paths relative to its own execution environment rather than the git checkout directory.
- **Impact:** All 18 PRs received ”File Not Found” errors in the review body.

Part II: Version 1.2

Version: 1.2
Test Repo: blackbox_test

3 Summary (v1.2)

This phase established the system's baseline stability and logical reasoning capabilities. The system was subjected to a manual load test of 25 Pull Requests.

Key Findings:

- **System Stability:** **PASS**. The system processed all 25 PRs in a single run without crashing.
- **Functional Accuracy:** **PASS**. The reviewer selected appropriate prompt and consistently identified code-level bugs without needing RAG context.

4 Findings (v1.2)

The 25 PRs were designed around 5 specific bug classes. The AI successfully identified the core issue in every instance.

4.1 Bug Class 1: Off-by-One Errors

Affected PRs: 1, 6, 11, 16, 21
Scenario: A loop condition `while i < b` excluded the upper bound.
AI Analysis (Sample from PR #1/PR #6):

> "Off-by-one error: The while loop condition in the `sum_range` function is incorrect. It should be `i <= b` instead of `i < b`. This will cause the function to exclude the upper bound `b` from the sum."

4.2 Bug Class 2: Unvalidated Dictionary Access

Affected PRs: 2, 7, 12, 17, 22
Scenario: Accessing `data['name']` without checking if keys exist.
AI Analysis (Sample from PR #2/PR #7):

> "KeyError: The code does not handle cases where the input dictionary `data` is missing the keys 'name' or 'age'. This will raise a `KeyError` exception."

4.3 Bug Class 3: Cache Logic Flaws

Affected PRs: 3, 8, 13, 18, 23
Scenario: A simple dictionary cache with no size limit or thread safety.
AI Analysis (Sample from PR #3/PR #8):

> "No cache set functionality... No cache invalidation: The cache does not have a mechanism to remove entries. This could lead to memory leaks... The cache is implemented as a simple dictionary, which is not thread-safe."

4.4 Bug Class 4: Missing Null Checks

Affected PRs: 4, 9, 14, 19, 24
Scenario: A function comparing `token == ""` crashing when token is `None`.
AI Analysis (Sample from PR #4/PR #9):

> ”Missing None Check: The function `is_authenticated` will throw an error when `token` is `None` because it tries to compare `None` with an empty string.”

4.5 Bug Class 5: Broad Exception Handling

Affected PRs: 5, 10, 15, 20, 25
Scenario: Using `except Exception as e:` to catch division errors.
AI Analysis (Sample from PR #5):

> ”Inadequate Exception Handling: The `except` block catches all exceptions using `Exception`, which is too broad. This can mask other unexpected errors... It's better to catch specific exceptions like `ZeroDivisionError`.”

5 Overall Conclusion

The testing campaign demonstrates the evolution of PULL-PANDA from a stable logic checker to a context-aware compliance tool.

1. **Version 1.2** proved the core value proposition: the LLM is highly effective at finding logical bugs (off-by-one, crash risks) and provides decent review.
2. **RAG Version 1.3** successfully layered on organizational awareness via RAG, enforcing style guides and custom rules. While the Static Analysis module faced initial configuration issues, the core AI reasoning remained robust and accurate.