Module 6 Capstone Milestone: Software Project Testing Plan

Alexander Ricciardi

Colorado State University Global

CSC480: Capstone Computer Science

Dr. Shaher Daoud

July 20, 2025

Module 6 Capstone Milestone: Software Project Testing Plan

In software development, software testing is a crucial activity of a project development lifecycle that aims to evaluate and improve software systems (IEEE Computer Society, n.d.). It ensures that these systems function correctly, are secure, meet stakeholders' requirements, and provide value to end users. Many types of software testing methodologies are available to developers, each with different advantages and disadvantages. The Mining Regulatory Compliance Assistant (MRCA) is an AI-powered system that uses a novel RAG technique, Advanced Parallel HybridRAG (APH), that combines vector similarity search, graph-based traversal search, and intelligent semantic fusion to provide grounded and accurate answers about U.S. Mine Safety & Health Administration (MSHA) regulations (30 CFR). This combination of AI and novel RAG system adds testing challenges to traditional software testing, such as non-deterministic outputs, AI data dependency factors, and the risk of AI hallucinations. This paper describes a testing strategy for the MRCA, including unit, integration, end-to-end for UI/API, reliability/fault-injection, and overall architectural evaluation testing, focusing on Architecturally Significant Requirements (ASRs) of the project, and provides five use test cases based on these testing strategies.

Architecturally Significant Requirements

To effectively design and execute test cases for a software project, it is important to understand its Architecturally Significant Requirements (Keeling, 2017). For MRCA, this translates to ensuring that the system is reliable in retrieving and generating MSHA regulations by implementing evaluation metrics such as retrieval confidence scores for vector and graph results, search results fusion quality scores, and final confidence scores for the aggregated outputs. These evaluation scores can be based on or similar to approaches such as Hybrid

Parameter-Adaptive RAG (HyPA-RAG) that score each retrieved passage for domain relevance in the legal field called a signals that can be defined as identifiers, terms, and numeric thresholds (Kalra et al., 2025); a complementary reliability-aware weighting scores for fused retrieved result similar to Reliability-Aware RAG (RA-RAG) scoring proposed by Hwang et al. (2024); and a retrieval evaluators that assign a final confidence score similar to the Auepora (A Unified Evaluation Process of RAG) RAG evaluation metric proposed by Ya et al. (2024).

Testing Strategies Overview

One of the most important testing principles is to test early, test often, mix white-box and black-box, and escalate from quick checks to targeted and deep evaluations (CSU Global, 2025). A white-box is a test that has explicit knowledge of the internal structure and implementation details of the software; on the other hand, a black-box is a test that does not have explicit knowledge of the internal structure and implementation details of the software. MRCA uses five test levels, which are a combination of white-box and black-box tests that focus on its ASRs. Unit Tests for CFR parsing, query normalization, embedding calls, graph traversals, and fusion. Component integration tests for context validity for VectorRAG, GraphRAG, and fusion, to evaluate the accuracy and confidence of the generated results. End-to-end testing for UI and API flows, measuring performance, reliability, and user value. Reliability and fault-injection testing to simulate service outages, latency, and auth failures to test the circuit breaker system and user messaging. Finally, perform an overall architecture evaluation to score ASRs with the goal of collecting feedback and identifying areas needing improvement.

Use Test Cases

Test Case 1: Regulatory Citation Retrieval - Component Integration Test

Confirm that MRCA returns the correct CFR citation results with reasonable confidence scores and latency. Example of input: “What does 30 CFR 56.12016 say about grounding of electrical equipment?” should output a correct CFR result with source; confidence ≥ .85 when vector+graph agree; p95 latency ≤ 5s. A failure will be a malformed citation and section missing

Test Case 2: Multi-Domains Query - Component Integration Test

Test the fusion validity across multiple regulatory domains, such as a combination of dust control, ventilation, and respirators domains. An input Example: “What are the regulations for underground drilling generating silica dust near diesel equipment?” The expected output of the parallel vector and graph searches and Advanced Hybrid fusion should cite multiple CFR sections related to the domains found in the prompt with a fusion_quality score ≥ .70. A failure will be one missing section or domain, conflicting passages, low complementarity, that is a low fusion score.

Test Case 3: Reliability Under Degraded External Services - End-to-End Testing

Tests the circuit breaker functionality, on/off states, retry/backoff, and user messaging when the Neo4j database, LLM providers, or network links or API fail. Example of condition (API input): simulate Neo4j DB downtime, slow LLM completion, no internet connection, expired API key. The system should detect these faults or issues; the circuit breaker should change from closed to open, and it should return a message to the user; most importantly, the session should be preserved; and the system should automatically retry the process experiencing the issue.

Test Case 4: Reliability, Fault-Injection Test, and Unit Tests

Tests the performance of the system under concurrent load and by injecting faults. Test the performance and injected fault against the typical metrics, such as a response time that should

be 10–35s (bound by LLM response time), and typical correct CFR citation, responses. The metric used should be a combination of regulatory section citations, topics, scenarios, comparative regulations, and large queries in varied concurrency tests.

Test Case 5: Confidence Score and Hallucination – Overall Architecture Evaluation (AHP)

Tests if the system detects responses to unsupported prompts, and checks confidence outputs. Example of inputs, off-domain (regulations) prompts such as “What sound a dog makes?”. The expected output should be a premade general response informing the user that their question is off-domain, that the system does not answer off-domain prompts; on the other hand, the system should answer in-domain prompts with a high final confident score. A failure will be a response low final confidence score and/or a reply off-domain.

Conclusion

MRCA is a complex system that combines AI and a novel RAG system, adding testing challenges to traditional software testing. This document provided a testing plan for MRCA, addressing its novel APH technology by implementing traditional and RAG/generative AI testing techniques that include unit testing, integration testing, end-to-end testing for UI/API, reliability testing for fault tolerance, and architectural evaluations based on ASRs. The provided use test cases map to these test types, they showcase specific inputs and expected results, and test regulatory responses, system performance under heavy load, and resilience to failure (circuit breaker). Through these testing techniques, MRCA can be improved and become a more reliable, accurate, robust, and valuable tool for querying MSHA regulations.

References

CSU Global (2025). Module 6: Project testing [Interactive Lecture]. Canvas. https://csuglobal.instructure.com/courses/110425/pages/module-6-overview?module_item_id=5733358

Hwang, J., Park, J., Park, H., Park, S., & Ok, J. (2024, June 2). *Retrieval-augmented generation with estimation of source reliability* (arXiv:2410.22954) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2410.22954

IEEE Computer Society. (n.d.). *The importance of software testing*. IEEE Computer Society. https://www.computer.org/resources/importance-of-software-testing

Kalra, R., Wu, Z., Gulley, A., Hilliard, A., Guan, X., Koshiyama, A., & Treleaven, P. (2025, February 25). *HyPA-RAG: A hybrid parameter adaptive retrieval-augmented generation system for AI legal and policy applications* (arXiv No. 2409.09046v2) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2409.09046

Keeling, M. (2017). Chapter 12: Give the architecture a report Card. *Design it*! *From programmer to software architect*. Pragmatic Bookshelf. ISBN-13: 978-1-680-50209-1

Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., & Liu, Z. (2024, July 3). *Evaluation of retrieval-augmented generation: A survey* (arXiv No. 2405.07437v2) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2405.07437