**MILESTONE 3**

**Milestone 2 Completion Status**

**Milestone 2 Completion**

All requirements for Milestone 2 have been successfully completed. This includes the integration of PyTest for testing, enhancement of GitHub workflows for seamless CI/CD, implementation of RAG (Retrieval-Augmented Generation) methodology, reorganization of the repository for better accessibility, comprehensive definition of the project's use case, and the development of an initial user testing plan. You can review the progress and implementation details in the repository.

[GitHub repository](https://github.com/Vishaalsai29/operator)

**Testing Results and Analysis**

We conducted systematic testing using DeepEval to comprehensively evaluate the system's performance and robustness. This approach allowed us to assess key metrics, identify vulnerabilities, and implement refinements based on real-world scenarios.

**Test Suite Setup**

DeepEval's advanced metrics suite was used to analyse three core areas:

* Correctness (GEval): Measured the factual accuracy of outputs compared to expected results.
* Faithfulness: Evaluated whether responses adhered to the provided context rather than relying on external knowledge.
* Contextual Relevancy: Assessed the alignment of retrieved content with the query's intent and its contribution to meaningful responses.

**Security/Vulnerability Findings**

The evaluation highlighted the importance of robust security and reliability mechanisms during API interactions and database queries. Challenges encountered included:

* Rate Limit Handling: OpenAI's rate limits posed interruptions during evaluation. To address this, retry mechanisms with exponential backoff were implemented, enabling smoother API interactions and uninterrupted operations.
* Data Sanitization: All user queries and retrieved contexts were sanitized to protect against vulnerabilities like SQL injection, data leaks, and API abuse. Malformed inputs were effectively handled to maintain system integrity.
* Authentication and Authorization: Secure API keys and environment variables were used for connections, ensuring no unauthorized access to sensitive systems.

**Performance Metrics**

Correctness had a pass rate of 33.33%, with most errors resulting from minor formatting issues like punctuation or rephrasing rather than factual inaccuracies. For example, a query about the top complaint categories for Wells Fargo failed due to an extra comma in the response, highlighting the need for formatting standardization through post-processing.

Faithfulness showed better performance, with a pass rate of 66.67%. Most responses adhered to the provided context, but occasional reliance on prior knowledge instead of retrieved information led to partially accurate or unsupported answers. For instance, a query about complaint resolution rates generalized data beyond the scope of the retrieved context.

Contextual relevancy presented significant challenges, as the system often retrieved irrelevant or insufficient information, leading to incomplete responses. For example, a query about complaints filed in a specific timeframe lacked the relevant data, highlighting the need for improved retrieval logic and vector embeddings.

**Reliability Measures**

To ensure the system's reliability during testing and deployment:

* Error Handling: Graceful fallback mechanisms were introduced. In cases where context retrieval or API calls failed, the system defaulted to general knowledge responses with appropriate disclaimers.
* Retry Mechanisms: Automated retry loops were added for failed API calls and database queries. This significantly improved reliability and reduced operational failures, particularly under heavy workloads or API rate-limit scenarios.

**Areas of Improvement**

Context retrieval requires significant refinement, as gaps in vector embeddings and retrieval logic often led to irrelevant or insufficient results. Improving database indexing and incorporating advanced embedding models will enhance the system's ability to retrieve context that aligns with user queries.

**Impact on Development**

The testing phase provided actionable insights to guide the final development and optimization of the RAG system.

**Final Project Planning**

**To ensure the successful completion of the project, the following steps have been planned**

**Front End Development:**

The front-end interface will be developed to ensure the application is user-friendly and aligned with the use case requirements. This includes integrating the UI with the backend services and ensuring a seamless query and response experience.

**Presentation Planning:**

**Slide Deck Preparation:** A comprehensive slide deck will be prepared, showcasing the problem statement, project objectives, implementation process, and achieved results.

**Flow-Based Slide Division:** The slides will be structured to follow the logical flow of the application development process, providing a clear and coherent narrative of the project journey.