MARMARA UNIVERSTY FACULTY OF ENGINEERING

Fall 2025-CSE3063

Group Number: 12

Object Oriented Software Design

Requirement Analysis Document

(RAD-Pyhton&LLM)

1. Vision

The primary vision of Iteration 2 is to migrate the existing Java-based RAG pipeline to a **Python** environment while enhancing the system's retrieval capabilities with "Semantic Search" features.

The scope includes replacing the basic Keyword Search with a pluggable architecture that supports **Vector Search** and **Hybrid Reranking**. Additionally, the system will introduce mechanisms for performance optimization (**Query Caching**) and automated bulk testing (**Batch Evaluation**) to ensure the accuracy and reliability of answers derived from university policy documents.

2. Goals

- **Language Migration:** Port the RAG pipeline to Python, maintaining strict adherence to SOLID principles and the Strategy Pattern.
- **Semantic Understanding:** Implement a `VectorIndex` and `EmbeddingStub` to retrieve documents based on meaning rather than just keyword matching.
- **Relevance Optimization:** Implement a `HybridReranker` to combine BM25 and Vector scores for better ranking.
- **Efficiency:** Implement `QueryCache` to store and retrieve answers for frequently asked questions.
- **Measurability:** Implement a `--batch` CLI mode and `EvalHarness` to calculate accuracy and latency metrics.

3. Functional Requirements

3.1. User Interface & Input (CLI)

- **FR-01 (Single Query):** The user shall be able to ask a single question via the command line using `python main.py --q "question"`.
- **FR-02 (Batch Mode):** The user shall be able to run bulk evaluations using `python main.py --batch questions.json`. The system must process all questions sequentially and output a summary report.
- **FR-03 (Configurability):** The system behavior (e.g., switching between `SimpleReranker` and `HybridReranker`) must be controlled via a `config.yaml` file without code modifications.

3.2. Processing Pipeline

- **FR-04 (Vector Retrieval):** The `Retriever` shall support a `VectorIndex` strategy that uses simulated embeddings (Stubs) to find relevant chunks based on cosine similarity.
- **FR-05 (Hybrid Reranking):** The `Reranker` shall support a `Hybrid` strategy that normalizes and combines scores from keyword retrieval and vector retrieval.
- **FR-06 (Caching):** The `RagOrchestrator` shall check the `QueryCache` before processing a request. If a hit occurs, the cached answer is returned immediately.
- **FR-07 (Answer Generation):** The `AnswerAgent` shall generate a final response using the top-ranked chunks and specific prompts, utilizing either a deterministic Stub or an LLM API.

3.3. Output & Logging

- **FR-08 (Citations):** Every generated answer must include a citation in the format `[docID:sectionID]` to verify the source of information.
- **FR-09 (Tracing):** The system must record detailed execution steps (inputs, outputs, latency) for every stage of the pipeline into a `.jsonl` file using the `TraceBus`.

4. Non-Functional Requirements

- **NFR-01 (Determinism):** Given the same input configuration and questions, the system must produce the exact same output and logs every time. "Stubs" must be used for random components like Embeddings/LLMs.
- **NFR-02 (Modularity):** The system must utilize the **Strategy Pattern** for `Retriever` and `Reranker` components, ensuring the Open/Closed Principle (OCP) is respected.

- **NFR-03 (Constraint):** No external database (SQL/NoSQL) or GUI shall be used. All data must be persisted in local JSON or YAML files.
- **NFR-04 (Traceability):** Each Python class and method must be traceable back to a specific requirement or Use Case step.

5. Risk Analysis(Python-LLM)

- **LLM Hallucinations:** The model may generate plausible but incorrect answers not found in the documents.
- **Mitigation:** Enforce strict citation validation ; any answer without a valid `[docID:section]` reference must be flagged or discarded.
- **Context Window Overflow:** Passing too many retrieved chunks to the Python LLM interface may exceed token limits (e.g., >4096 tokens).
- **Mitigation:** Implement a "Context Pruning" mechanism in `AnswerAgent` to truncate input or prioritize only the top-ranked chunks before prompting.

6. Glossary

- **RAG (Retrieval-Augmented Generation):** A framework that retrieves data from outside a foundation model and augments the prompts by adding the relevant retrieved data in context.
- **VectorIndex:** A data structure that stores text as mathematical vectors (embeddings) to enable semantic search.
- **Hybrid Reranking:** A technique that combines keyword-based scoring (like TF-IDF/BM25) with semantic scoring to improve search relevance.
- **Stub:** A piece of code used to stand in for some other programming functionality (e.g., an API) during testing to ensure determinism.
- **JSONL:** A text-based format where each line is a valid JSON object, used here for logging trace events.