CLOUD AMBASSADORS

ML Engineer Role

Take-Home Assessment

Instructions

- You are expected to complete within 5 working days from the date of receiving the assessments
- **Open-Source Requirement:** Use only open-source tools, libraries, and frameworks for your project (e.g., Python, Flowise, n8n etc.,).
- **Deployment and Execution:** You can execute the project on your local machine or on any public server if needed. Ensure that all tools and datasets used are freely available and open source.
- **Source Code:** Host all your source code, configuration files, and dataset links (if applicable) on a public GitHub repository. Make sure the repository is well-organized, documented, and includes instructions for running the project locally.
- **Documentation:** Provide clear documentation in the GitHub repository with details on how to run your code, any assumptions made, and descriptions of the tools you used.

Agentic RAG Implementation with Open-Source and Gemini 1.5 Flash/Pro LLM

Description:

In this assignment, you will build a retrieval-augmented generation (RAG) pipeline that leverages an open-source Large Language Model (LLM) combined with the Gemini 1.5 Flash/Pro environment.

The pipeline should ingest a set of documents, store them efficiently for retrieval, and serve as a knowledge base that can be queried by an LLM-driven “agent.” The agent should use these documents as references to produce contextually relevant answers.

Requirements:

1. **Document Ingestion & Storage:**

- Choose a collection of text documents (e.g., technical documentation, knowledge base articles, or research papers) to form your corpus.
- Preprocess and embed these documents using an open-source embedding model.
- Store the resulting vectors in a vector database (e.g., pgVecor, Qadrant, Chroma or another open-source vector store).

2. **Model Integration:**

- Utilize an open-source LLM (e.g., a Llama-based model) **or** Gemini 1.5 Flash/Pro. Ensure that the environment is set up correctly and the model is integrated seamlessly.
- Implement a retrieval-augmented generation agent that:
  - Takes a natural language query as input.
  - Fetches the most relevant documents from the vector database.
  - Uses the retrieved context to produce a more accurate, contextually informed response.

3. **Agentic Behavior:**

- The agent should be able to reason about when to retrieve documents versus when it can rely on existing context.
- It should be capable of chain-of-thought reasoning (e.g., using LangChain) to break down complex queries into steps and fetch documents at the right time.
- Clearly log the reasoning steps and retrieval process for transparency and debugging.

4. **Evaluation & Testing:**

- Provide a set of sample queries and demonstrate how the agent responds before and after integrating the retrieved documents.

- Include simple evaluations (e.g., manual checks or a small question-answer test set) to show improvements in response quality when using retrieval.

5. **Code Quality & Documentation:**

- Write clear, maintainable code with proper structure and comments.
- Include a README with:
    - Instructions for environment setup (including Gemini 1.5 Flash/Pro configuration details).
    - Steps for running the ingestion, retrieval, and Q&A pipeline.
    - Explanation of the agent's architecture and reasoning mechanism.

6. **Open Source & Version Control:**

- Use only open-source components for embeddings, LLM (except Gemini), and vector storage.
- Push all code, documentation, and instructions to a public GitHub repository.
- Include a brief design document (as Markdown) in the repo explaining the approach and any design trade-offs.

Deliverables:

- A publicly accessible GitHub repository containing:
- Source code for the ingestion, embedding, and RAG pipeline.
- A README detailing setup and usage instructions.
- Example queries and responses.
- A brief design explanation and architectural overview document.