LLM Specialist Assignment

Overview

Create a **Retrieval-Augmented Generation (RAG) pipeline** that allows users to upload documents and ask questions based on their content. The system should leverage **vector databases** for efficient retrieval and an **LLM API** (e.g., OpenAI, Gemini, or another REST-based model) for generating responses. The entire application should be containerized using **Docker** and deployable on **cloud or local environments**.

Requirements:

1. Document Ingestion & Processing:

- Support uploading up to 20 documents, each with a maximum of 1000 pages.
- Chunk documents into manageable sizes for efficient retrieval.
- Use text embeddings to store document chunks in a vector database (e.g., FAISS, Pinecone, Weaviate, or ChromaDB).

2. Retrieval-Augmented Generation (RAG) Pipeline:

- Accept user queries and retrieve relevant document chunks.
- Pass the retrieved chunks to the **LLM API** for contextual response generation.
- Ensure responses are **accurate, concise, and relevant** to the uploaded documents.

3. API & APPLICATION ARCHITECTURE:

a. Implement a **REST API** using **FastAPI, Flask, or Express.js**.

b. Expose endpoints for:

   i. Uploading documents

   ii. Querying the system

   iii. Viewing processed document metadata

c. Store document metadata in a **relational or NoSQL database**.

4. DEPLOYMENT & CONTAINERIZATION:

a. Provide a **Docker Compose** setup with all necessary services.

b. Ensure seamless deployment on **local machines and cloud environments** (e.g., AWS, GCP, Azure).

5. TESTING & DOCUMENTATION:
    a. Write **unit and integration tests** for document retrieval and query handling.
    b. Provide a **clear README.md** with:
        i. Setup and installation instructions.
        ii. API usage and testing guidelines.
        iii. Configuration details for using different LLM providers.

DELIVERABLES

- **GitHub repository** with the complete source code.
- **Docker setup** for local and cloud deployment.
- **Well-documented README.md** with setup and API usage instructions.
- **Automated tests** for validation.
- **Postman collection** (optional) for testing API endpoints.

EVALUATION CRITERIA:

- **Efficiency** of document retrieval and response generation.
- **Scalability** and **performance** of the RAG pipeline.
- **Code quality, modularity, and adherence to best practices**.
- **Ease of setup and deployment** using Docker.
- **Thoroughness of documentation and test coverage**.

SUBMISSION:

Submit a **GitHub repository link** within **2 days** including all deliverables.

You are encouraged to submit a working demo via a publicly accessible URL. While preferred, this is not mandatory.