Python Test

Project Description:
Build a RAG-based Question-Answering System

Objective:
Create a Retrieval-Augmented Generation (RAG) API using an LLM to answer questions over a set of documents. The system should be scalable, API-driven, and containerised.

Key requirements:

- **LLM Integration:**
  Use an LLM API (e.g., OpenAI GPT, Gemini) or an open-source model.
  Connect it with a retrieval pipeline (e.g., FAISS, Weaviate, Chorma).
- **Backend API:**
  Build APIs using FastAPI or Flask.
  Expose endpoints for:
    Document Upload & Indexing
    Query Answering
    Fetching Logs
- **Logging:**
  Store queries and responses in a NoSQL database (e.g., MongoDB).
  Log query text, context, response, timestamp, and duration.
  Provide an endpoint to fetch logs.
- **Deployment:**
  Dockerize the project for easy deployment.
  Include a Dockerfile and clear setup instructions.
- **Optional (Bonus Points):**
  Implement a multi-agent workflow (e.g., retrieval agent + refinement agent).
  Add basic monitoring and health-check APIs.

Deliverables:

- **Code Repository:**
  - Source code with APIs, logging, and documentation.
  - Dockerfile for deployment.
  - README with setup instructions.
- **NoSQL Logs:**
  - Sample log entries stored in the database.
  - API endpoint to retrieve logs.
- **Optional:**
  - Multi-agent workflow example.
  - Monitoring or health-check endpoints.

Evaluation Criteria:

API Functionality
Query Accuracy
NoSQL Logging Implementation
Code Quality & Documentation
Docker Deployment
Optional Features (Bonus)