**Test Task: Build an Agentic RAG System with LangGraph, Pinecone, and OpenAI**

**Objective**

Develop a minimal Retrieval-Augmented Generation (RAG) system using **LangGraph**, with support for **PDF documents only**. The system should use **Pinecone** as the vector database, **OpenAI embeddings** for text representation, and an **OpenAI LLM** for query answering. The system must expose functionality through a **FastAPI service**.

**Requirements**

**1. Document Support**

* Only **PDF files** should be supported.
* Use a text splitter.

**2. Core Components**

* **Vector DB**: Pinecone/Chroma DB
* **Embeddings**: OpenAI embeddings/Huggingface
* **LLM**: OpenAI GPT model/ Any other LLM
* **Graph Orchestration**: LangGraph / Openai Agents SDK

**3. API Endpoints (FastAPI)**

Implement these endpoints:

1. **POST /add\_file** Upload a PDF, extract text, create embeddings, and store them in Pinecone.  
    **Output**: JSON with file\_id and success message.
2. **POST /chat** Accept a user query, retrieve relevant chunks from Pinecone, and use OpenAI LLM to generate a response.  
    **Input**: { "query": "What does the document say about X?" }  
    **Output**: JSON with answer.
3. **DELETE /delete\_file/{file\_id}** Delete all vectors related to a given file ID from Pinecone.  
    **Output**: Success or error message.
4. **PUT /update\_file/{file\_id}** Replace the existing file’s vectors with embeddings from the new PDF.  
    **Output**: Success message.

**Directory Structure**

Follow this structure for clean, maintainable code:

rag\_app/

│

├── main.py                 # FastAPI entrypoint

├── api/                    # API routes

│   ├── \_\_init\_\_.py

│   ├── routes\_chat.py      # /chat endpoint

│   ├── routes\_files.py     # add, delete, update file endpoints

├── core/                   # Core configurations

│   ├── \_\_init\_\_.py

│   ├── config.py           # Load env vars (OpenAI key, Pinecone API key, etc.)

│

├── services/               # Business logic

│   ├── \_\_init\_\_.py

│   ├── data\_injestion\_service.py      # PDF text extraction and chunking

│   ├── embeddings\_service.py      # openai/huggingface embeddings

│   ├── vectordb\_service.py # Pinecone/chroma insert, delete, update

│

├── utils (optional)/                  # Utility functions/helpers

│   ├── \_\_init\_\_.py

│   ├── logger.py           # Centralized logging

│

├── requirements.txt        # Dependencies

├── README.md               # Setup instructions

├── .env                    # To store API keys

**Deliverables**

1. **Codebase** following the structure above.
2. **README.md** with setup and run instructions, including:
   * How to start a FastAPI app.
   * How to configure environment variables (.env with Pinecone API key, OpenAI key).
   * Example requests for all endpoints (curl or Postman).

**Evaluation Criteria**

* Correctness of API implementations.
* Proper integration of LangGraph, Pinecone, and OpenAI.
* PEP8-compliant coding and directory structure.
* Modular and clean separation of concerns.
* Error handling for invalid inputs or missing files.