AI Developer Assignment

Objective

Build an AI-powered system that can **read and understand PDF documents**, and **answer user queries** based on the content of those PDFs.

Task Description

Create a web-based application (or CLI, optionally) that performs the following:

1. PDF Upload & Ingestion

- Allow the user to upload one or more PDF files.
- Extract the text from the uploaded PDFs.
- Chunk the content intelligently (e.g., by paragraph or sentence).
- Store the embeddings of those chunks in a vector database.

2. Model Training (Embedding Generation)

- Generate embeddings for the extracted text using any pre-trained embedding model (e.g., OpenAI, HuggingFace Sentence Transformers, etc.).
- Save these embeddings into a vector database like:
  - **Pinecone**
  - **FAISS**
  - **ChromaDB**
  - **Weaviate**
  - **CouchDB**

3. Query Answering

- Provide a UI/CLI where the user can enter natural language questions.
- When a query is entered:
  - Convert it to an embedding.
  - Search the vector database for the most relevant chunks.

    - Use a language model (e.g., OpenAI GPT, Llama, Mistral, or any open-source model) to generate an answer using **retrieval-augmented generation (RAG)**.
- Display the answer, and optionally also:
    - Show the source PDF and the context used to answer.

Functional Requirements

- Support uploading of **multiple PDF files**.
- Store uploaded data and embeddings persistently.
- Queries should return contextually accurate answers using uploaded documents only.
- Handle large PDFs efficiently.
- Include proper error handling and validations.

Technical Requirements

You can use any stack, but here's a suggested tech outline:

Backend

- Python (preferred)
- Flask / FastAPI / Streamlit (if web UI is needed)

AI/NLP Libraries

- `langchain` or `llama-index` (optional)
- `sentence-transformers` or OpenAI embeddings
- `PyMuPDF`, `pdfplumber`, or `pdfminer.six` for PDF parsing
- `faiss`, `chroma`, or any vector DB for storing embeddings

Vector Database

- FAISS (for local setup)
- Pinecone / ChromaDB (for hosted solutions)

Frontend (Optional)

- React / Streamlit / simple HTML form

Bonus Points

- Show **source page number** or **highlighted text** from the PDF.
- Add support for querying using voice (speech-to-text).
- Allow re-training or updating PDF data.
- Containerize using Docker.
- Write unit/integration tests.
- Provide a UI that mimics ChatGPT-style conversation.

Deliverables

- GitHub repository containing:
  - Source code (with requirements.txt or environment.yaml)
  - Instructions to run locally
  - Sample PDF file(s)
  - README with:
    - Tech stack used
    - How to run and test
    - Any known issues or limitations

Timeline

Please submit within **2 days** from receiving this assignment.