Assignment 1: AI-driven Document Insight Service

Description

Build a Python-based REST API (FastAPI or Flask) that:

- Ingests PDF or image documents (e.g., scanned contracts, invoices) and extracts text (via EasyOCR, PyMuPDF, or both, or other – choose to demonstrate).
- Answers user questions about the uploaded documents using a QA model (e.g., DistilBERT QA or any other transformer, local or any LLM via API).
- Find or create random/dummy test docs (store in Github repo).

Core Requirements

- **POST /upload**: accept one or more documents, store them for session-based or user-based retrieval.
- **POST /ask**: take a question, run a QA pipeline over the stored documents, and return the answer.
- Dockerize the entire application.

Optional Enhancements *(pick at least one)*

- Integrate Named Entity Recognition (choose model) to highlight entities in responses.
- Implement Retrieval-Augmented Generation (RAG) with embeddings (FAISS or similar) for improved context retrieval.
- Add a caching layer (Redis or other) to store and quickly fetch document embeddings.

General Optional Enhancements *(pick any)*

- **UI**: Prototype with Gradio or Streamlit to demonstrate the workflow.
- **MLOps**: Modular packaging, CI/CD integration, structured logging, monitoring.
- **Performance**: Profile model inference latency or API throughput and optimize.
- **Security**: JWT authentication, input sanitization, rate limiting.

ChatGPT/LLM Usage

You may use ChatGPT or any LLM to accelerate development. Just be ready to explain how you validated AI-generated code, and why you chose the final approach.

Delivery

Host your solution in a GitHub repository. Your **README.md** must include:

- Setup instructions
- Manual installation and Docker option instructions
- Example API requests and responses
- Brief description of your approach (tools/models chosen and why)
- *Optional*: screenshots, demo video, or other creative additions

Any extra creative feature beyond the above is a big plus. Show us your ingenuity!