Technical Test: Retrieval-Augmented Generation (RAG) with Dropbox Integration

Objective

Create a minimal FastAPI application in Python that performs Retrieval-Augmented Generation (RAG) from documents stored on Dropbox. The documents will be of varying formats (PDF, PowerPoint, Word) and may contain images requiring OCR.

Requirements

1. **Core Functionality**
   - Connect to a Dropbox folder and retrieve documents (PDF, PPT, DOCX) automatically.
   - Extract text from the documents. For documents or slides containing images, perform OCR using any free OCR library.
   - Implement a simple RAG pipeline. When given a user query, the system should use the content of the retrieved documents to provide a response.
2. **API Endpoint**
   - Expose at least one endpoint with FastAPI to accept a query string.
   - Return a JSON response with the relevant snippets from the documents, as well as the generated answer.
3. **Project Setup**
   - Use Python and FastAPI.
   - Provide a clear and concise README explaining:
     - How to install and run the application.
     - How to set up any necessary environment variables (e.g., Dropbox API keys).
     - A brief explanation of your approach.
4. **Testing & Demonstration**
   - We will send you a set of documents during the live call.
   - You should demonstrate how your application processes these documents and answers questions.

- Be prepared to walk through your code, showing how you structured your solution.

5. **Evaluation Criteria**
   - **Code Quality & Readability**: Is the code organized, well-documented, and easy to understand?
   - **Functionality**: Does the solution successfully perform OCR, retrieve relevant content, and generate meaningful answers?
   - **RAG Accuracy**: Is the retrieval-augmented generation pipeline returning correct and relevant snippets?
   - **OCR Integration**: Use any free OCR library successfully (the actual OCR accuracy is not the focus; demonstration of implementation is).
   - **Maintainability**: Could this project be extended or maintained easily?
6. **Time Constraints**
   - You have one full day to work on this exercise. Use your time wisely to ensure a working prototype.
7. **What to Submit**
   - Your code in a Git repository or zipped folder.
   - A brief README with instructions and any setup steps.
8. **Technical Review Call**
   - We will schedule a call to review the code.
   - You will walk us through your project's architecture.
   - We will send you a few test documents and ask you to demo the solution live.