PDF Q&A App – Technical Summary

Architecture & Tech Stack Decisions

Frontend: React (JavaScript)

- **Why?** React provides a **dynamic UI** and **component-based structure** that makes the app interactive.
- **Decisions Made**:
  - Used **Create React App (CRA)** for simplicity.
  - Designed a **chat-like interface** for an intuitive user experience.
  - Implemented **useState and useRef** to manage UI state and file input efficiently.

Backend: FastAPI (Python)

- FastAPI is **lightweight, asynchronous**, and well-suited for **handling API requests efficiently**.(Rest Architecture)
- **Decisions Made**:
  - Used **FastAPI for handling requests** (file uploads, chat messages, and Pydantic validation).
  - Integrated **Hugging Face Inference API** for **summarization** and **question answering**.
  - Implemented **session-based handling** for managing user interactions using SQL based database(storing session id)

Session-Based State Management Using SQLite

In our PDF Q&A application, we use **SQLite** to store session data temporarily until the chat session is ended. Instead of relying on an in-memory dictionary , we utilize an SQLite database to maintain session persistence throughout the interaction.

How It Works

1. **Session Creation:**
    - When a user uploads a PDF, a unique `session_id` is generated.
    - The extracted text and summary of the PDF are stored in an SQLite database under this `session_id`.

2.Session Retrieval:

- When the user sends a message, the application retrieves the corresponding PDF data from the database using the `session_id`.
- This allows the system to generate context-aware responses based on the uploaded document.

2. **Session Termination:**
    - When the user decides to end the chat, the session data is deleted from the database.
    - This prevents unnecessary storage consumption and ensures that each interaction starts fresh.

Why Use SQLite?

- **Persistence:** Unlike in-memory storage, SQLite ensures that session data is not lost due to server restarts.
- **Lightweight & Efficient:** SQLite is an embedded database, making it an ideal choice for temporary session storage.
- **Concurrency Handling:** Multiple users can interact with different PDFs simultaneously without conflicts.

AI Models Used in the PDF Q&A Application

To enable document summarization and question-answering functionality, we integrate two state-of-the-art NLP models from Hugging Face. These models process user-uploaded PDFs, generate summaries, and facilitate interactive Q&A based on the document content.

Summarization Model

Model: `facebook/bart-large-cnn`

- This is a transformer-based model pre-trained by Facebook using the **BART** architecture.
- It is designed specifically for text summarization tasks, making it ideal for condensing long PDF documents into concise summaries.
- It captures key points of the document while maintaining coherence and readability.

Usage in the Application:

- When a user uploads a PDF, its extracted text is passed to this model.
- The model generates a summary, which is displayed before the interactive Q&A session begins.

Question-Answering Model

Model: `deepset/roberta-base-squad2`

- Based on the **RoBERTa** architecture, fine-tuned on **SQuAD 2.0**, this model excels at extracting relevant answers from a given text.
- It takes a user query and finds the most appropriate answer from the PDF's extracted content.
- The model can handle unanswerable questions gracefully by determining if the given context contains relevant information.

Usage in the Application:

- Once the PDF is uploaded and summarized, the user can start asking questions.

- The user’s query, along with the extracted document text, is passed to this model.
- It retrieves the most relevant answer from the document and returns it to the user in real-time.

Deployment: Docker

- **Why?** Docker ensures **consistent deployment across environments**.
- **Decisions Made**:
  - Containerized both the **FastAPI backend** and **React frontend**.
  - Used **Docker Compose** to manage multi-container setup.

Failures and Future Improvements

Challenges Faced

1. **Hugging Face API Constraints:**
   - The free-tier API has rate limits and sometimes returns **delayed or incorrect responses**, affecting the user experience.
   - Some queries return incomplete answers due to token length restrictions in free models.
2. **Quality of AI Responses:**
   - The **summarization model sometimes omits important details**, leading to incomplete document understanding.
   - The **QA model struggles with complex queries** and may return generic or irrelevant responses.
3. **Performance Issues:**
   - Due to API latency, users sometimes **experience a delay** in responses.
   - Sending long PDF texts to the AI model results in **high processing time** and possible timeouts.

Future Improvements (If I Had More Time)

1. **Fine-Tuning Custom AI Models**
   - Train a **custom summarization and QA model** using domain-specific data for better accuracy.
   - Deploy a **self-hosted AI model** instead of relying on Hugging Face's free-tier API.
2. **Enhancing API Performance**
   - Implement **caching mechanisms** to store previous answers and reduce repeated API calls.

    - Optimize **token length** and **context window** to improve the accuracy of responses.
3. **Better UI/UX & Error Handling**

    - Improve **real-time feedback** when models take too long to respond.
    - Display **loading indicators** and **suggest possible queries** to enhance usability.
4. **Multi-PDF Handling & Advanced Queries**

    - Allow users to upload **multiple PDFs** and cross-reference information.
    - Introduce **semantic search** to retrieve answers more efficiently instead of relying purely on text matching.

Final Thoughts

Despite API limitations, **the core system—file uploads, backend session management, and UI interactions—works well**. With additional time and resources, optimizing AI response quality and performance would be the top priority.