Document Oracle - A Technical Summary

A project by Rikhil Gupta

The project is deployed on this link: https://document-qa-frontend.onrender.com/

This document discusses the technical aspects of the development of the web app.

Backend

The backend is built with FastAPI and Python, providing a REST architecture. This was chosen due to its speed and efficiency in development time, and of course, my comfort. For reading the pdfs, the python library PyPDF2 was used. This stored the consolidated text in the PDFs into a string which was used for context to the AI model.

Frontend

The frontend is built with React and TypeScript, and care has been taken to design the frontend so that it is intuitive and easy to use in nature. React's useState hooks have been used for component level state management. It also maintains and displays the conversation history.
For connecting to the backend, it leverages Axios for HTTP requests.

AI/Agentic Component

Among the various candidates, GPT 4o-mini API was chosen to be the backbone of the web app. This was because of its ability to have a large context length - of 128,000 tokens. The large context length enables the web app to perform well on huge pdfs with a lot of information. Its performance on benchmarks rivals other models, while being significantly more cost effective ($0.15 per 1M input tokens, and $0.60 per 1M output tokens).

Additional Points

The web app is able to handles these cases gracefully:

- Invalid File uploads - Uploads of only text-pdfs are accepted. If other types of files are uploaded, or if the pdf has no text but only images, the system points it out and asks to re-upload the files.
- Relevancy - If the requested information is not present in the provided documents, the system points it out rather than hallucinating and giving wrong information.

- Large Files - The large context length of GPT 4o mini supports densely worded pdfs of more than 40-50 pages.
- Multiple documents - The web app supports upload of multiple documents, for better output results and relevancy.
- Automatic Document Summarization - Once uploaded, the documents are read, understood and summarized by the AI assistant. This brief ~300 word summary is automatically shown as soon as it’s available.

Challenges Faced

I was initially using GPT 3.5 Turbo, and when I uploaded large pdfs for testing, it used to fail. I browsed and looked for cost effective and large context lengthed AI models. I compared about 4-5 models, from deepseek, openAI, Anthropic and finally settled on 4o-mini due to reasons already described.
Being new to this, establishing a connection between the backend and frontend was a little challenging due to the data formats, but after some googling around I figured it all out.

Going Forward

I’m going to continue this as a hobby project, and make it more powerful with time. Some improvements that could be done:

1. Implement an image-to-text extraction from the documents - this would not only enable jpg and png uploads, but also pdfs which are essentially just images instead of text. Moreover, in text-and-image pdfs, more information could be extracted by creating a text summary of the image.
2. Using a hybrid of AI models - For text, we can continue with 4o-mini but when images and videos are encountered, we can use Dalle to make it more robust.
3. Implementing Search - Information retrieval from the internet can provide more relevant information and end up in higher output quality.