**Technical Report**

**Task to Do:**I have develop a chatbot-like interface that allows users to upload a PDF document and interactively ask questions.

* Extract text from the PDF.
* Process questions interactively and provide structured answers.
* Match answers word-to-word if the question exists directly in the text.
* Handle low-confidence answers gracefully by returning "Data Not Available."
* Output the results as a structured JSON blob.

**Technical Specifications**

* openai: For querying the GPT-4o-mini model.
* PyPDF2: For extracting text from PDF documents.
* json: For formatting and displaying results as a JSON blob.

**AI Model:** GPT-4o-mini (configured using OpenAI API).

**Output Format:**

* Line-by-line interactive answers displayed in CLI.
* Final results saved as a structured JSON blob.

**Usage**

* Run the Python script in a terminal.
* Provide the path to the PDF document when prompted.
* Interact with the chatbot by typing your questions.
* Type exit to end the session.
* The final output is displayed in JSON format with paired questions and answers.

**Requirements**

Python 3.8 or higher installed.

!pip install PyPDF2

!pip install openai==0.28

**Code Explanation:**

* The extract\_pdf\_text function uses PyPDF2 to read and extract text from all pages of the PDF document.
* The word\_to\_word\_match function scans the PDF text for an exact match of the user's question. If found, it directly returns the line.
* The get\_answers\_from\_model function queries GPT-4o-mini with the extracted PDF text and the user's question, returning an AI-generated answer.
* Responses that are low-confidence or ambiguous return "Data Not Available."
* The chatbot\_interface function manages user interaction. It prompts the user to upload a PDF, accepts questions, and displays answers line-by-line.
* At the end of the session, the script consolidates all questions and answers into a structured JSON blob and displays it.
* Te system handles missing or inaccessible PDF files, malformed input, and API failures gracefully.

**Other Possible Ways to Do This Task**

* Create embeddings for the PDF content using OpenAI embeddings and store them in a vector database (e.g., Pinecone, FAISS).
* Preprocess text (e.g., tokenize, remove special characters) for better parsing and matching.
* Implement keyword or regular expression-based matching for faster retrieval in cases of direct matches.
* Combine GPT answers with exact text matches and vector search for increased reliability and accuracy.

**Conclusion**

I have tried to give you a solution based on your requirement , as this code is available with Colab file system if required I will go with creating API's for the same.