Assignment_GenAI: Detailed Report

Project Overview

This project is an AI-powered **PDF-based Question Answering system**. It allows users to:

1. Upload a PDF.
2. Ask questions related to that PDF.
3. Get answers generated using **Google's Large Language Model (LLM)**, with relevant context retrieved via **SentenceTransformer** and **ChromaDB** (vector store).

🧠 Components Involved

1. SentenceTransformer

- Used to convert chunks of the PDF content into embeddings.
- Enables semantic search for relevant content to the question.

2. Google LLM (`get_answer`)

- Uses the Google Generative Language API to generate answers based on context returned by the embedding search.
- Context-aware QA.

🌐 Endpoints in `main.py`

`POST /upload_pdf/`

- **Purpose**: Uploads and processes a PDF.
- **Inputs**: PDF file and `pdf_name`.
- **Process**:

    - Saves the file.
    - Calls `process_pdf()` to extract content and store it in ChromaDB (vector DB).
- **Output**: JSON success message.

`POST /ask_question/`

- **Purpose**: Answers a question related to a specific PDF.
- **Inputs**:
    - `question`: the user's query.
    - `pdf_name`: the name of the already-uploaded PDF.
- **Process**:
    - Calls `search_chroma()` to find relevant chunks.
    - Calls `get_answer()` to generate an answer using Google's LLM.
- **Output**: The answer string.

Next, I'll analyze the logic in `pdf_processor.py` and `qa_engine.py` to include descriptions of all major functions and files in the final report.

Here's the breakdown of the key files and functions based on your `utils` folder:

📄 `pdf_processor.py`

Purpose: Extract text from PDFs, chunk the content, convert it into embeddings, and store it in ChromaDB.

🔧 **Key Components:**

- **`PdfReader`**: Reads the uploaded PDF file.
- **`RecursiveCharacterTextSplitter`**: Splits the text into manageable chunks.

- `SentenceTransformerEmbeddings`:
    - Model used: `all-MiniLM-L6-v2`
    - Converts chunks into dense vector embeddings.

⚙️ Key Function:

`process_pdf(pdf_path, pdf_name)`

- Reads and extracts text from the PDF.
- Chunks the text.
- Creates vector embeddings using SentenceTransformer.
- Stores them in ChromaDB under the collection named after the PDF.

🧠 `qa_engine.py`

Purpose: Performs semantic search and answers user questions using Google Gemini.

🛠️ Key Components:

- **Google Generative AI (`gemini-1.5-pro-002`)** via LangChain.
- **Chroma**: Retrieves similar content using embeddings.
- **SentenceTransformer**: Reused from `pdf_processor`.

🔍 Functions:

`search_chroma(question, pdf_name)`

- Uses the vector store to find the top relevant chunks from the specified PDF based on the question.
- Returns those chunks as context.

`get_answer(context, question)`

- Sends both the `context` and `question` to Google Gemini model.
- Returns a high-quality natural language answer.