**PROJECT REPORT**

**Visual and Text Question Answering**

**Atishay Jain (E23CSEU1738)**

**Submitted to: Dr. Nitin Arvind shelke**

A logo for a university

AI-generated content may be incorrect.

**1. Introduction**

In today's data-driven world, PDFs are one of the most common formats for storing important documents, be it academic research papers, business reports, or user manuals. However, extracting specific information, whether textual or visual, from large PDFs can be tedious.  
Our project addresses this problem by creating a chatbot capable of answering user questions based on the content of any uploaded PDF. The chatbot is not limited to text; it can also understand and respond to queries related to images embedded in the document. Additionally, it offers advanced functionalities like summarization, web-based fact checking, chat history management, and image extraction.

**2. Objective**

* Build a chatbot that can intelligently answer questions based on a given PDF's **text** and **images**.
* Provide users with additional functionalities such as **summarization**, **chat history saving**, **web-based fact search**, and **image extraction**.
* Deliver a user-friendly experience through a **web application** interface.

**3. Features**

**3.1 PDF-Based QA**

* Upload any PDF and ask questions.
* The chatbot retrieves answers based on the textual and visual content inside the PDF.

**3.2 Image-Based QA**

* Supports answering queries related to images extracted from the PDF using specialized models.

**3.3 Chat History**

* Every question and its corresponding answer are saved into a chat history for easy reference.

**3.4 Summarization**

* Summarize the entire content of the uploaded PDF with just a single click.

**3.5 Web-Based Fact Checking**

* Enable the "Web Search" option to validate answers using Google search.
* Real-time retrieval of up-to-date information through Google's Gemini API.

**3.6 Image Extraction**

* Extracts and displays all images embedded inside the PDF separately for easier access.

**4. Technology Stack**

* **Python**: Core programming language.
* **Streamlit**: Used for building the web-based user interface.
* **LangChain**: Applied for Retrieval-Augmented Generation (RAG) and chaining various components.
* **Sentence-Transformers**: Employed to generate sentence embeddings for semantic search.
* **PyMuPDF**: Utilized for reading and extracting content (both text and images) from PDF files.
* **Ollama Models**:
  + **Llama2**: Used for text-based question answering.
  + **Llava**: Specialized in answering questions based on visual data (images).
* **Google Gemini API**: Used for fetching real-time factual data from the web.

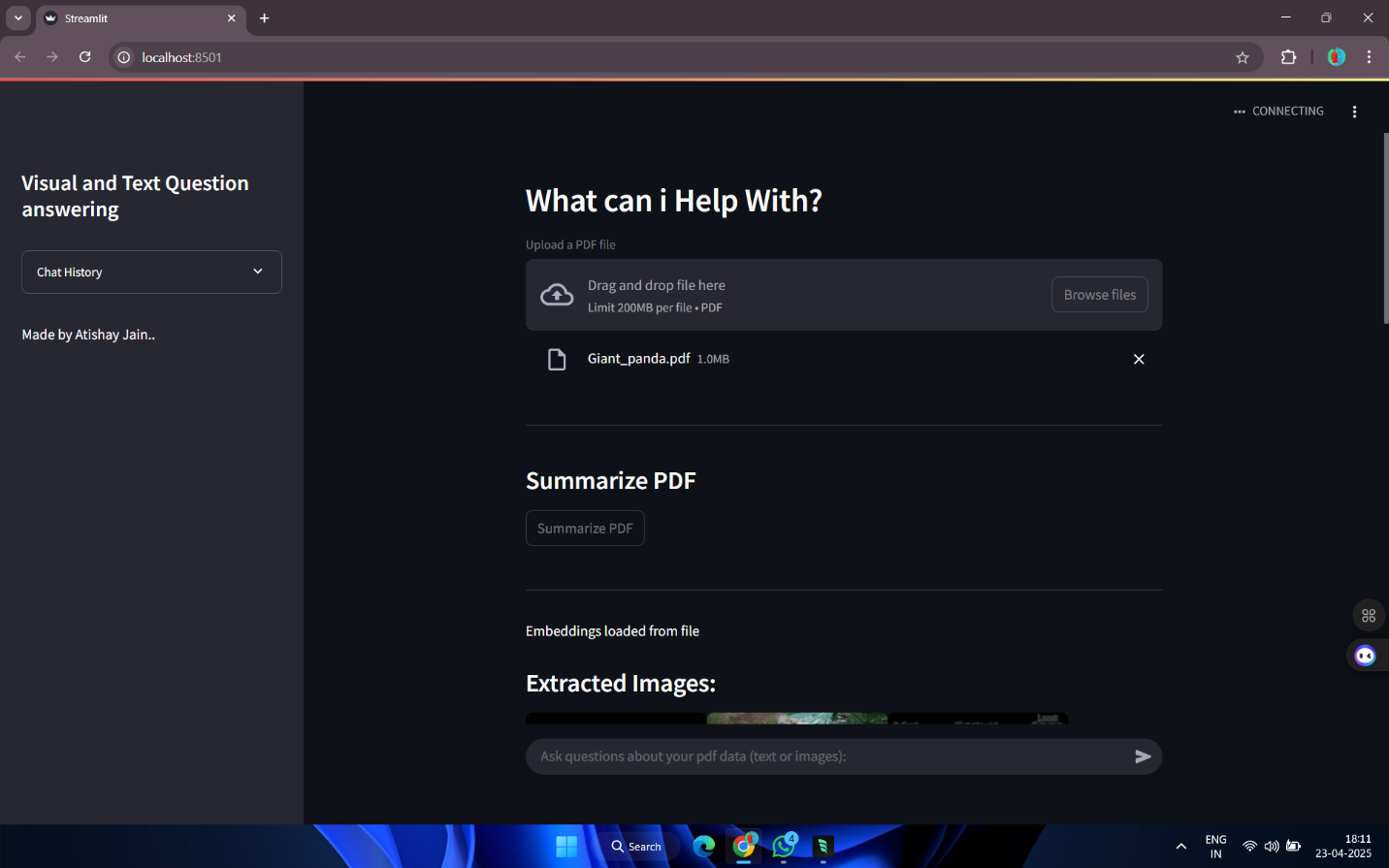
**5. Working Mechanism**

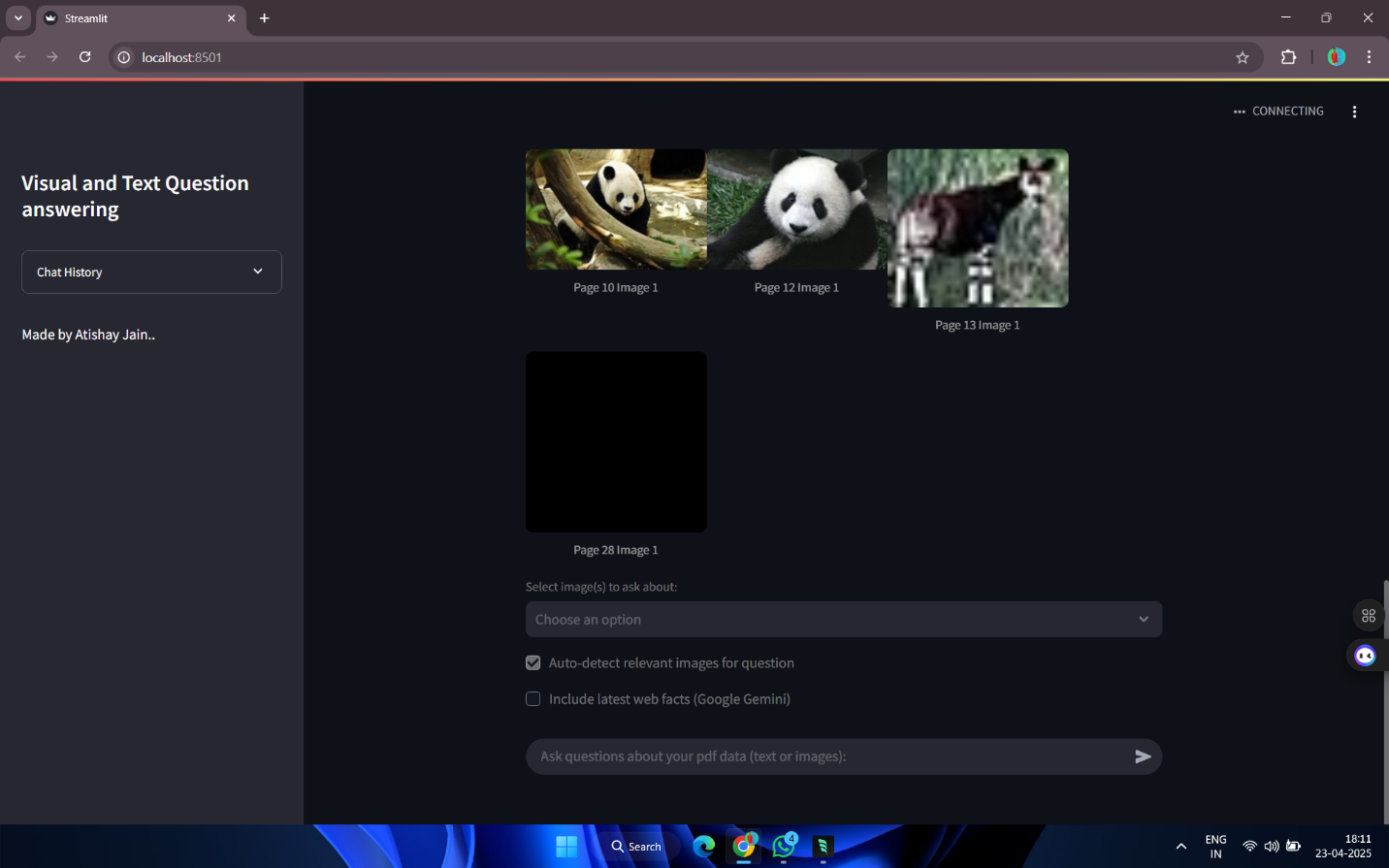
1. **PDF Upload**: The user uploads a PDF file through the web app.
2. **Content Extraction**:
   * PyMuPDF is used to extract text and images separately.
   * Text embeddings are generated using Sentence-Transformers.
3. **Query Handling**:
   * If the question relates to text, LangChain + Llama2 are used to fetch and generate an answer.
   * If the question involves images, Llava processes the extracted images to find an appropriate answer.
4. **Summarization**:
   * Summarization is triggered via a button, using Llama2 to generate a concise version of the document.
5. **Web Fact Search**:
   * If web search is enabled, the chatbot uses the Google Gemini API to verify and augment the answer.
6. **Chat History Saving**:
   * All interactions are stored and can be reviewed during the session.

**6. Conclusion**

The Visual and Text Question Answering Chatbot makes document exploration seamless by intelligently answering both text-based and image-based queries from uploaded PDFs. By integrating advanced AI models and web search capabilities, it provides a reliable and enriched user experience.  
This project demonstrates the powerful application of large language models (LLMs), multimodal AI, and real-time web search integration in building intelligent, interactive systems for everyday tasks

1. **Snippets**

****

****