Title of the Project in Capital Letters.....AI - Chatbot

Capstone-I project report submitted

by the student of

Hybrid UG program in Computer Science & Data Analytics

Student Name Prince Patel

Roll No. ……. 24A12RES465

Group No. …….. 108

INDIAN INSTITUTE OF TECHNOLOGY PATNA

BIHTA – 801106, INDIA

Date.....30-05-2025

Declaration

I hereby declare that this submission is my own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date:

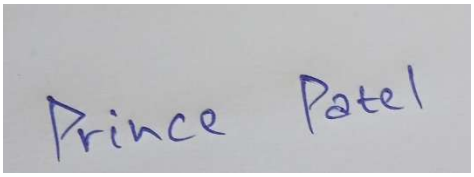
30-05-2025

Student Name, Roll No, Group No., and Signature

Name: Prince Patel

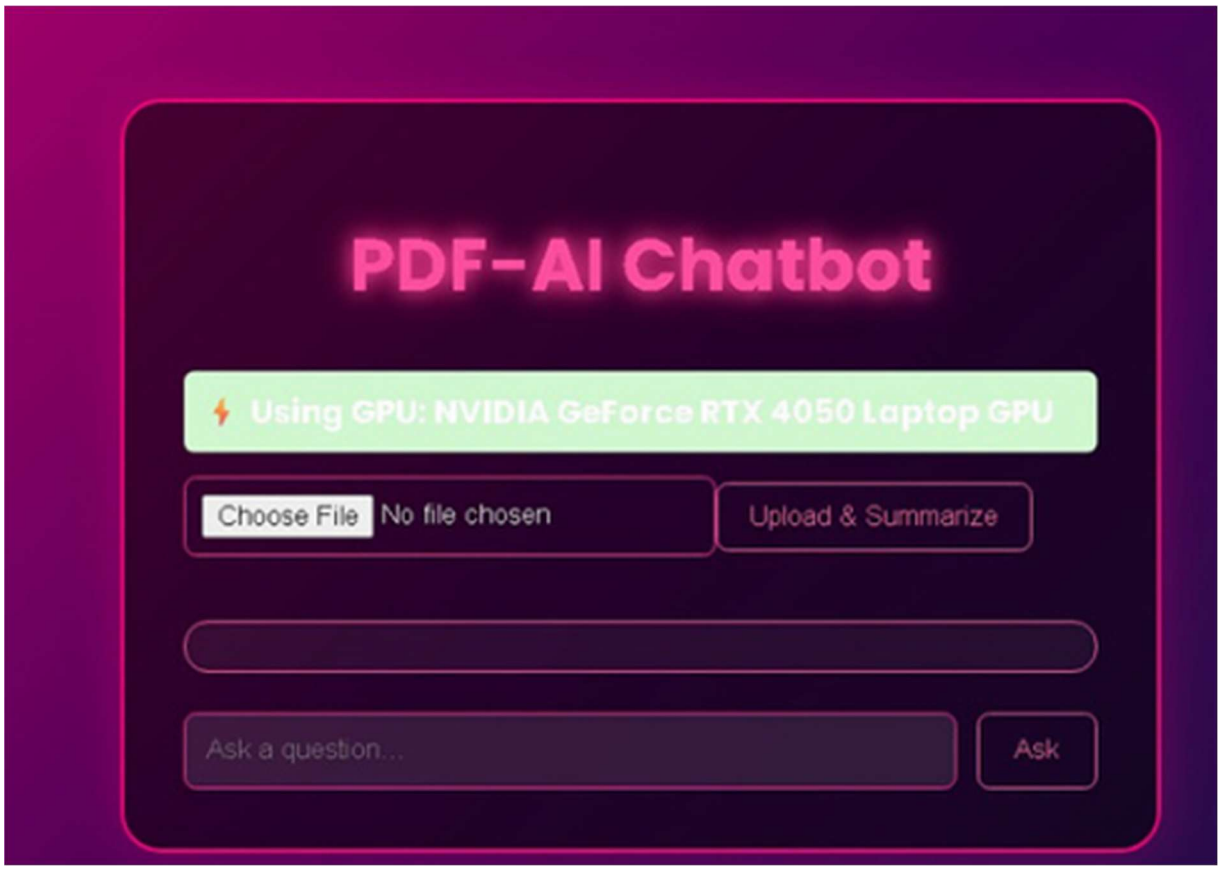
Roll No. - 24A12RES465

Group No. - 108



Summary of the Project

We propose an AI-powered chatbot system that combines intelligent document processing with conversational AI. Our solution is built on a modular architecture that integrates PDF handling, summarization, and question answering using advanced LLMs Large Language Models).



Problem Statement

Problem Statement

In today's fast-paced academic and professional environments, individuals often need to extract relevant information from large PDF documents such as research papers, textbooks, legal files, or reports. Manually scanning through these documents to find specific answers is time-consuming, inefficient, and mentally exhausting. Traditional keyword-based search tools fail to understand context, cannot summarize content effectively, and offer no conversational interface.

Therefore, there is a need for an intelligent system that can understand the content of a PDF, generate concise summaries, and provide accurate, context aware answers to user queries in a conversational manner.

Solution

To address the inefficiencies of manually extracting information from large PDF documents, we propose an AI-powered chatbot system that combines intelligent document processing with conversational AI. Our solution is built on a modular architecture that integrates PDF handling, summarization, and question answering using advanced LLMs Large Language Models).

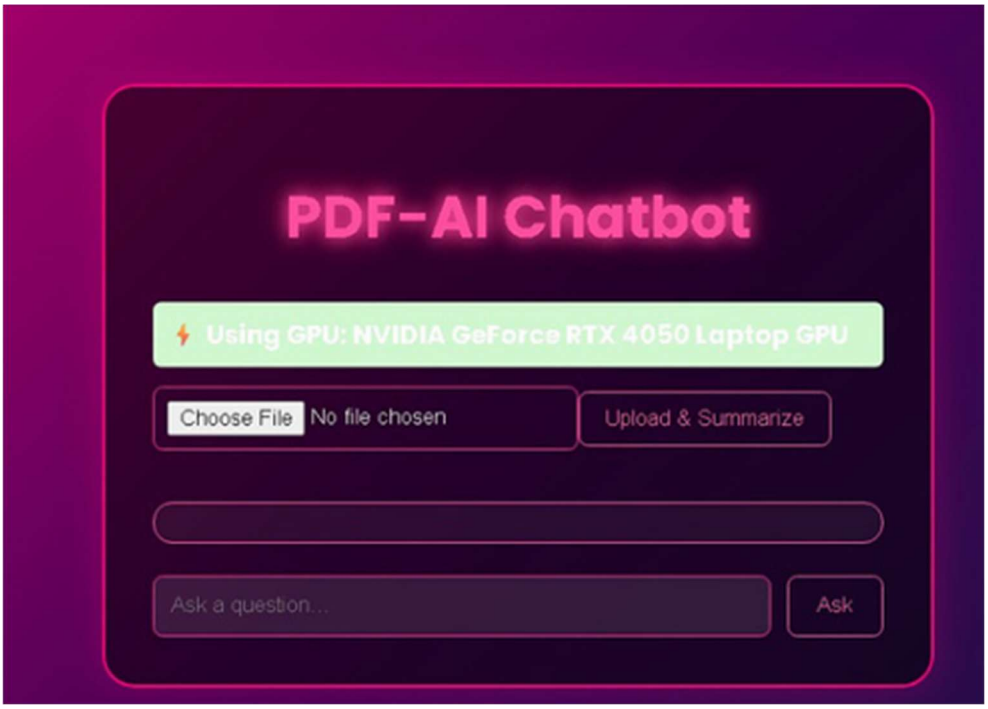
🔧 Key Components:

PDF Ingestion & Preprocessing

Users can upload one or more PDF files. The system parses and chunkifies the content, preserving logical structure while preparing it for language model processing.

My Learnings:

FRONTEND - {REACT}



This frontend is built in React using functional components. It features a file upload input, chat interface, and real-time interaction with a FastAPI backend. Axios is used for all network communication. The app displays real-time status messages, tracks chat history, and adapts based on whether the backend is using CPU or GPU. The UI leverages a clean aesthetic with a responsive and intuitive layout.

APP.js - Core Frontend

🔧 1. State Management (Using useState)

```
const [pdfFile, setPdfFile] = useState(null);
const [status, setStatus] = useState("");
const [question, setQuestion] = useState("");
const [chatHistory, setChatHistory] = useState([]);
const [deviceStatus, setDeviceStatus] = useState(null);
```

State Variable	Pur pose
pdfFile	Stores the selected PDF file
status	Tracks the upload/summarization status
question	Stores user input for chat
chatHistory	List of Q&A pairs
deviceStatus	Indicates if GPU/CPU is active (fetched from backend)

🔧 2. Device Info Fetcher (UseEffect)

```
React.useEffect(() => {
axios.get("http://localhost:8000/status")
.then((res) => setDeviceStatus(res.data))
.catch((err) => console.error("Failed to fetch device status:", err));
}, []);
```

- Runs **once on load**
- Hits /status endpoint
- Sets deviceStatus to display GPU or CPU badge

🔧 3. PDF Upload Handler

```
const handleUpload = async () => {
if (!pdfFile) return;
const formData = new FormData();
formData.append("file", pdfFile);
```

- Appends selected file into a FormData object
- Posts it to: POST /upload
- Calls: POST /summarize right after that
- Updates status to keep user informed

* This separation ensures modular backend: upload and summarize are two endpoints, not overloaded.

🔧 4. Chat Q&A Handler

```
const handleAsk = async () => {
if (!question.trim()) return;
setChatHistory([...chatHistory, { question, answer: "..." }]);
```

- Checks if user typed something.
- Appends question + placeholder ... answer to chat.
- Sends `POST /chat` with (question)
- Updates with `chatHistory` with actual answer on response.

- This dynamic chatHistory updating makes Q&A feel live and conversational.

App.css – Styling Overview

The project uses a custom app.css file for a visually appealing and modern UI. Key styling features include:

- **Gradient Background**: A diagonal gradient blend of pink, purple, and dark blue creates a vibrant look.
- **Glassmorphism Effect**: Container styled with semi-transparent dark background and neon-pink border for a soft glass-like appearance.
- **Typography**: Clean and modern 'Poppins' font for better readability.
- **Buttons and Inputs**: Transparent elements with smooth hover effects and pink-themed borders.
- **Chat UI**: Scrollable chat box and styled messages with light-glow background and left-accent border.
- **Animations**: A fade-in animation adds smooth entry transitions.

This design ensures a user-friendly and aesthetically pleasing interface.

Team Member's Contribution:

Teammates and My Contribution

This project was a collaborative team effort, with each member taking charge of specific responsibilities crucial to the development and completion of the PDF AI Chatbot system.

➢ My Contribution

- Designed and developed the **frontend interface** using **React**.
- Implemented file upload UI, chat window integration, and overall user experience.

➢ Satyam Pathak

- Took lead in **backend development** using FastAPI, building APIs for file upload, summarization, and chat interactions.

➢ Viplove

- Conducting in-depth **research** and gathering **key insights** relevant to the project topic. Their analysis helped shape the direction and effectiveness of our overall work."

➢ Spandan Pradhan

- He help in **pushing the code** and **managing updates** on **GitHub**. They also played a key role in **debugging** and ensuring the smooth functioning of the project."

Conclusion & Future Work

Final Flow of the PDF-AI Chatbot (Current Version)

- UserUploads PDF

- FastAPI Backend Receives File]

- PDFis Chunked → Each Chunk Sent to Summarizer LLM (falconsai/text-summar)

- Summarized Chunks Stored In-Memory]

- UserAsks a Question via Frontend React)]

- Relevant Summaries Sent to QA LLM (mistralai/Mistral-7B Instruct-v0.2

- Response Generated → Sent to Frontend]

- User Sees Chat Response Related to PDF Content]

Stack Involved: React (frontend) FastAPI Pydantic (backend) OpenRouter LLM APIs + asyncio/threadpool (concurrency).

Features to Be Added (Future Scope)

🔮 Future Work

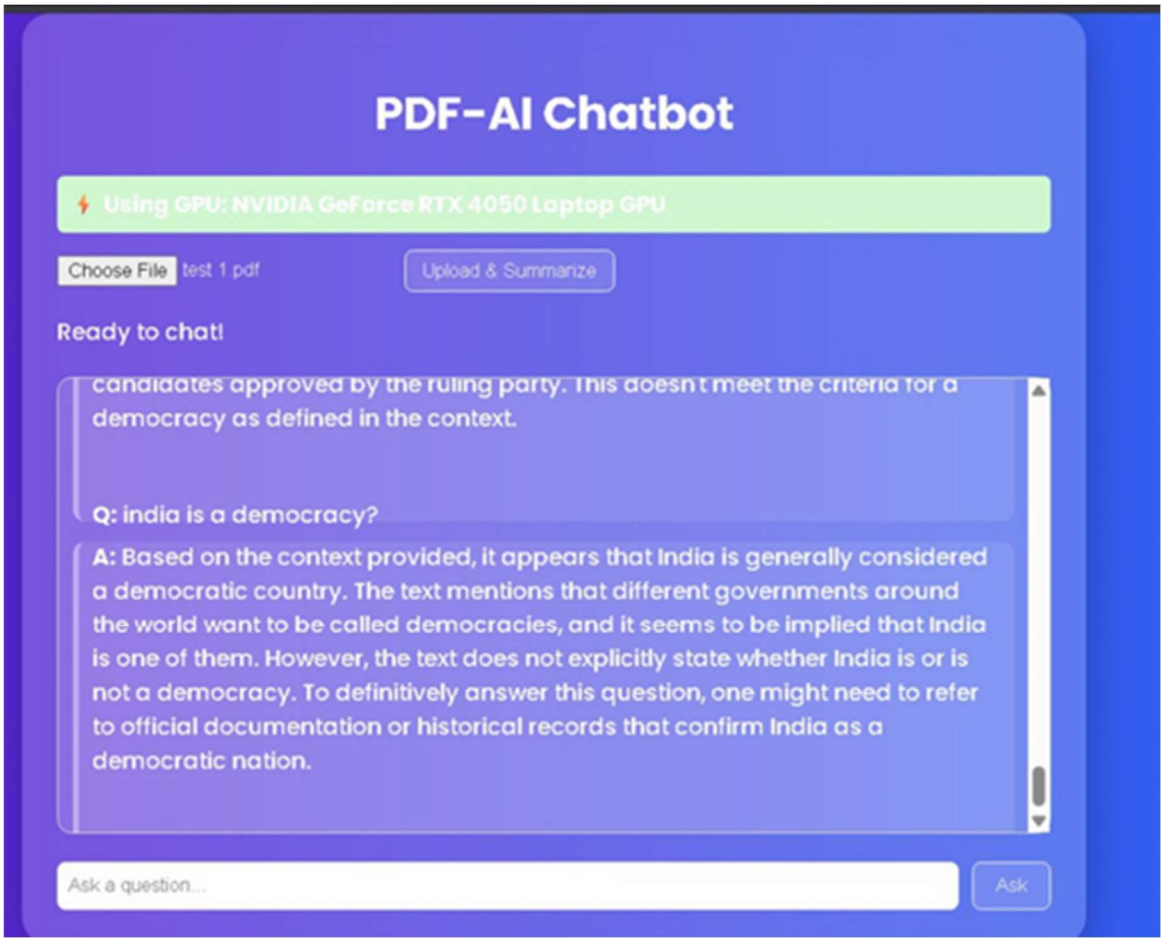
- **Switch to Azure AI APIs** for better inference instead of OpenRouter.
- **Use PostgreSQL or MongoDB** to store PDFs, summaries, and chats.
- **Add ThreadPoolExecutor** for faster PDF processing.
- **Deploy backend on Azure** and frontend on Vercel.
- **Integrate Vector DB** (like FAISS) for long-term memory and context search.
- **Add authentication** and secure multi-file handling.
- **Improve UI** with multi-file querying, progress loader, and dark/light mode.

Final Conclusion

This project showcased the powerful integration of modern backend architecture, large language models, and an intuitive frontend to solve the problem of navigating large PDFs intelligently. With the ability to summarize and answer user queries based on uploaded documents, the PDF AI Chatbot proves how AI can enhance productivity.

By leveraging OpenRouter APIs, FastAPI, Pydantic, and React, we have laid the groundwork for a scalable, production-ready AI system. The next phase— powered by Azure AI services, database integration, and vector memory optimization—will push this project into a robust, real-world deployment that not only performs better but also handles real user workloads efficiently.

Github repo and project snapshots:



** This is from earlier frontends versions , we later changed it.

❖ **Backend repo →**

https://github.com/satyampathak2607/Capstone_project--pdf-ai_chatbot

❖ **Frontend repo ⟶**

https://github.com/satyampathak2607/pdf-ai-frontend/tree/master/src