🤖Smart PDF Chatbot

A next-gen AI assistant for your PDFs — ask questions, get instant answers, and manage conversations like a pro!

👎About

The **Smart PDF Chatbot** is a **Generative AI-powered assistant** that transforms static PDF documents into an interactive knowledge base. Ask natural language questions, get **context-aware answers**, and interact with multiple PDFs without manual searching. Think of it as **ChatGPT meets your PDF library**.

👉🏾Why This Project

- Tired of scrolling PDFs for key information?
- Want a **dynamic Q&A** experience with your documents?
- Need **persistent chat sessions** to continue your research?

This project solves all of that with:
- **RAG (Retrieval-Augmented Generation)**
- **FAISS vector storage**
- **HuggingFace embeddings**
- **Streamlit-based interactive UI**

😖Key Features

Feature	How It Helps You
📄**Upload PDFs**	Instantly process one or multiple PDFs
RAG-powered answers	Vector search + LLM reasoning
📰**Chat sessions**	Save, rename, and continue past conversations
🕓**Modern UI**	Dark, glassy interface with smooth animations
🗒️**Smart responses**	If info isn't in the PDF, bot tells you politely
🕐**Dynamic updates**	Add/update/remove PDFs seamlessly

💊How It Works

1. **Upload PDFs:** Documents loaded and parsed with `PyPDFLoader`.

2. **Text Chunking:** Large PDFs split into manageable text chunks.
3. **Vector Embeddings:** HuggingFace model generates embeddings.
4. **FAISS Storage:** Efficient vector search for retrieving relevant chunks.
5. **LLM Response:** GPT, Gemini, or Groq generates contextual answers.
6. **Session Management:** Each chat is stored, retrievable, and renameable.

🎉Tech Stack

- **Frontend:** Streamlit
- **LLM Integration:** OpenAI / Gemini / Groq
- **Document Processing:** LangChain (`PyPDFLoader`, `RecursiveCharacterTextSplitter`)
- **Embeddings:** HuggingFace Sentence Transformers
- **Vector Storage:** FAISS
- **State Management:** Streamlit `session_state`
- **UI Design:** Custom CSS, dark/glassy theme

👂 Project Layout

- `app.py` → Streamlit UI & main app
- `rag_pipeline.py` → RAG logic (retrieval + prompt construction)
- `vectorstore_manager.py` → FAISS & embeddings
- `chat_OpenAI.py` / `chat_Gemini.py` → LLM wrapper
- `session_manager.py` → Session handling
- `history_manager.py` → Saves conversation history
- `data/` → User-uploaded PDFs
- `requirements.txt` → Python dependencies

😰Quick Start

🤩Clone & Navigate

```
git clone https://github.com/aruk3002/RAG_PDF_CHATBOT.git
cd smart-pdf-chatbot
```

🤪Setup Virtual Environment

```
python -m venv venv
source venv/bin/activate  # Mac/Linux
venv\Scripts\activate     # Windows
```

🤫Install Dependencies

```
pip install -r requirements.txt
```

😈Add API Keys

Create a `.env` file in project root:

```
OPENAI_API_KEY=your_api_key
GEMINI_API_KEY=your_api_key
GROQ_API_KEY=your_api_key
```

🤭Run the App

```
streamlit run app.py
```

Open your browser: `http://localhost:8501/`

😅Example Conversation

User:

> What does manufacturing involve?

Bot:

> Manufacturing is not explicitly defined in the document, but it discusses processes like production efficiency, machinery, and quality control. Essentially, it converts raw materials into finished products.

😏Updating Knowledge Base

- **Add PDFs:** Upload → embeddings & FAISS updated automatically.
- **Replace PDFs:** Re-upload → re-embedding for only updated files.
- **Remove PDFs:** Delete → reflected in next session.

No retraining required!

😟Future Enhancements

- 📌Keyword-based suggested questions
- 💾Persistent FAISS index between sessions
- Multi-user session isolation
- 🗑️Chat export & delete options

Contributing

1. Fork the repo
2. Create a feature branch
3. Submit a pull request

🎉Author

Chandan Aruk
📊Data Analyst | 💡AI & Data Science Enthusiast
LinkedIn