GENAI PDF Chatbot

Project Objective:

The goal of this project is to create an intelligent, conversational chatbot using Streamlit, LangChain, and Google's Generative AI capabilities. The chatbot interacts with users by processing uploaded PDF files, indexing the contents, and providing accurate and context-aware responses to user queries. The solution integrates state-of-the-art language models and vector search techniques to ensure robust, precise, and detailed answers, transforming PDFs into a searchable knowledge base.

Project Components:

- Core Technologies Used:
  - Streamlit: A Python framework used for creating web apps with minimal code.
  - PyPDF2: Library for reading and extracting text from PDF files.
  - LangChain: A library that helps with building applications powered by language models.
  - FAISS (Facebook AI Similarity Search): Used for efficient vector search and document retrieval.
  - Google Generative AI (Gemini-Pro): Generative AI model used to provide accurate, human-like conversational responses.
  - dotenv: For loading and managing environment variables.

Functional Components:

1. PDF File Handling:

- PDFs are uploaded using the Streamlit interface.
- The `get_pdf_text` function reads all the text from each page in the uploaded PDFs using PyPDF2.

2. Text Splitting:

- Large chunks of text are split into smaller segments using LangChain's `RecursiveCharacterTextSplitter` to ensure effective processing and indexing.
- The split text chunks facilitate efficient similarity searches when queried.

3. Vectorization:

- Text chunks are vectorized using embeddings generated by Google's Generative AI embeddings (`GoogleGenerativeAIEmbeddings`).
- These embeddings represent the semantic meaning of text chunks, making it possible to perform similarity searches.
- FAISS, a vector search engine, is used to store and search through embeddings for relevant chunks quickly.

4. Conversational Chain Setup:

- A LangChain-based QA model, `ChatGoogleGenerativeAI`, is used for conversational responses.
- Customizable prompt templates are employed to ensure accurate, contextually relevant answers.
- If no relevant answer is found in the context, the chatbot gracefully informs the user.

5. User Interface:

- The main page layout is styled using Streamlit's `st.markdown` with custom HTML/CSS to provide an engaging and user-friendly experience.
- Users can upload multiple PDF files, submit queries, and receive contextually accurate responses.
- The sidebar contains a file uploader and triggers text processing and indexing functions.

Environment Setup:

```
from dotenv import load_dotenv

import os

load_dotenv()  # Load environment variables from .env file

genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))  # Configure API key for Google Generative AI
```

Functions Overview:

1. Extracting Text from PDFs:

    - `get_pdf_text(pdf_docs)`: Iterates over the uploaded PDF files and extracts text from each page using PyPDF2.

2. Splitting Text into Chunks:

    - `get_text_chunks(text)`: Uses LangChain's `RecursiveCharacterTextSplitter` to split text into chunks of customizable sizes, reducing overlap while ensuring context continuity.

3. Vectorization and Embedding Creation:

    - `get_vector_store(text_chunks)`: Uses Google Generative AI to embed text chunks and creates a searchable FAISS index.

4. Conversational Interaction:

    - `get_conversational_chain()`: Loads a LangChain QA model and creates a conversational prompt chain.
    - `user_input(user_question)`: Takes user input and searches for relevant documents using FAISS. Returns a response using the conversational model.

5. UI Layout and Styling:

- Streamlit components (`st.set_page_config`, `st.markdown`, etc.) are used to structure and style the chatbot's interface.

Project Workflow:

1. PDF Upload:

- The user uploads one or more PDFs using the Streamlit sidebar.
- Uploaded files are processed, and text content is extracted using `get_pdf_text`.

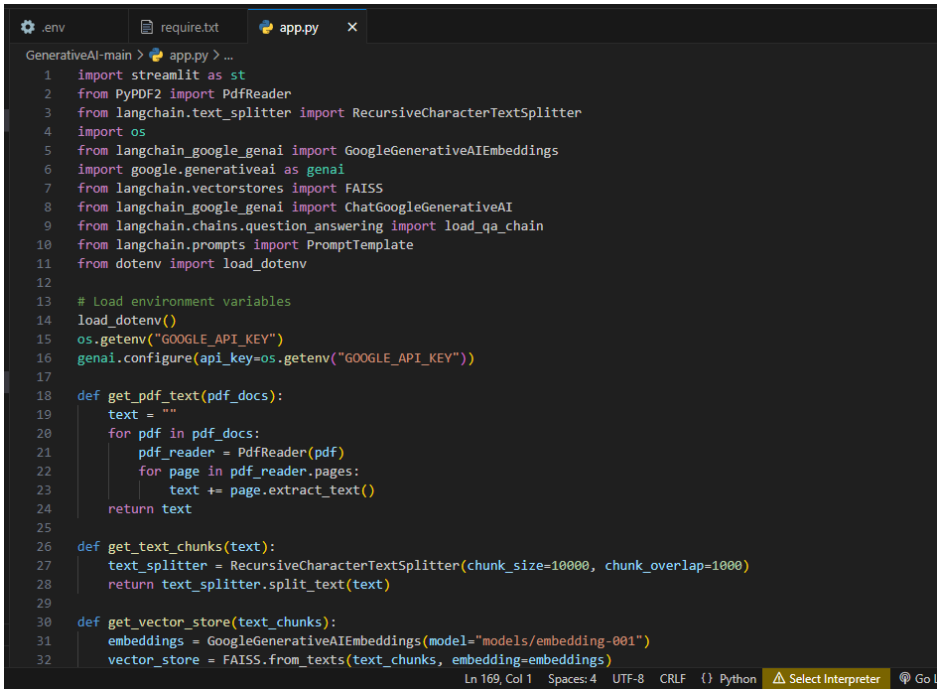
2. Text Processing:

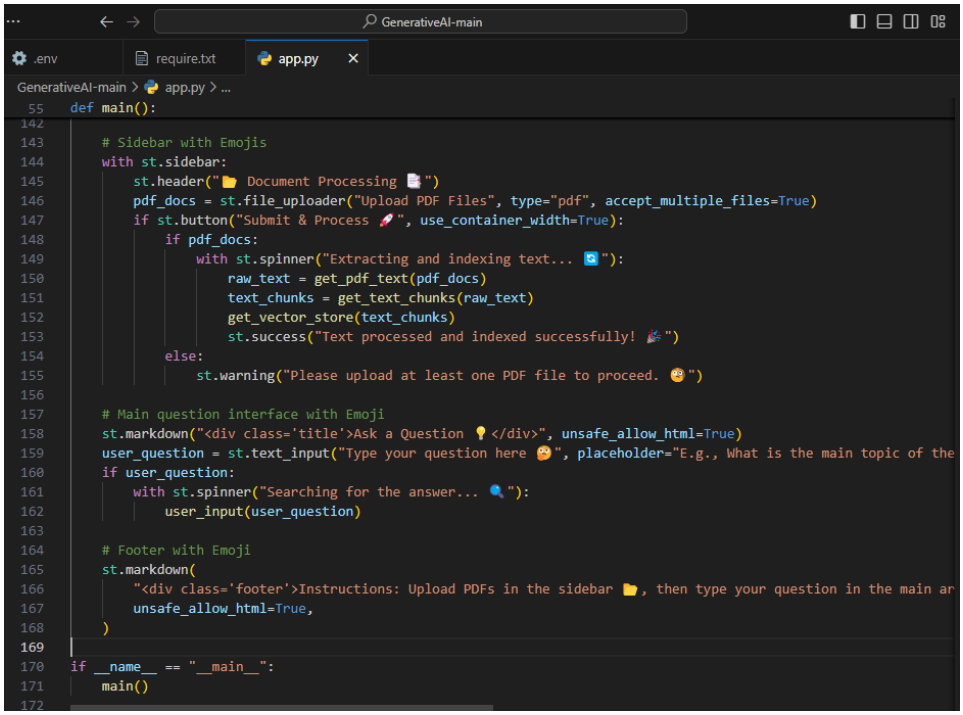
- Text is split into manageable chunks using LangChain's `RecursiveCharacterTextSplitter`.
- The chunks are embedded and stored using FAISS for efficient vector search and retrieval.

3. Query Handling:

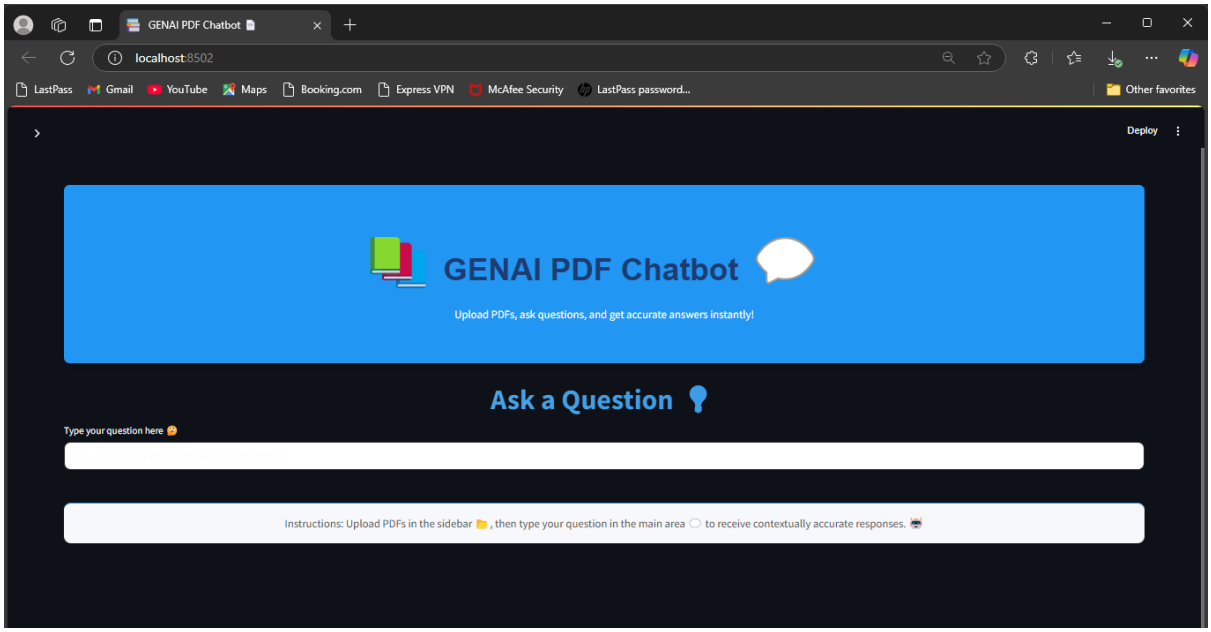
- The user enters a question in the text input box.
- A similarity search is performed on the embedded text chunks using FAISS.
- Relevant chunks are passed to the LangChain conversational chain, which generates a response using Google Generative AI models.

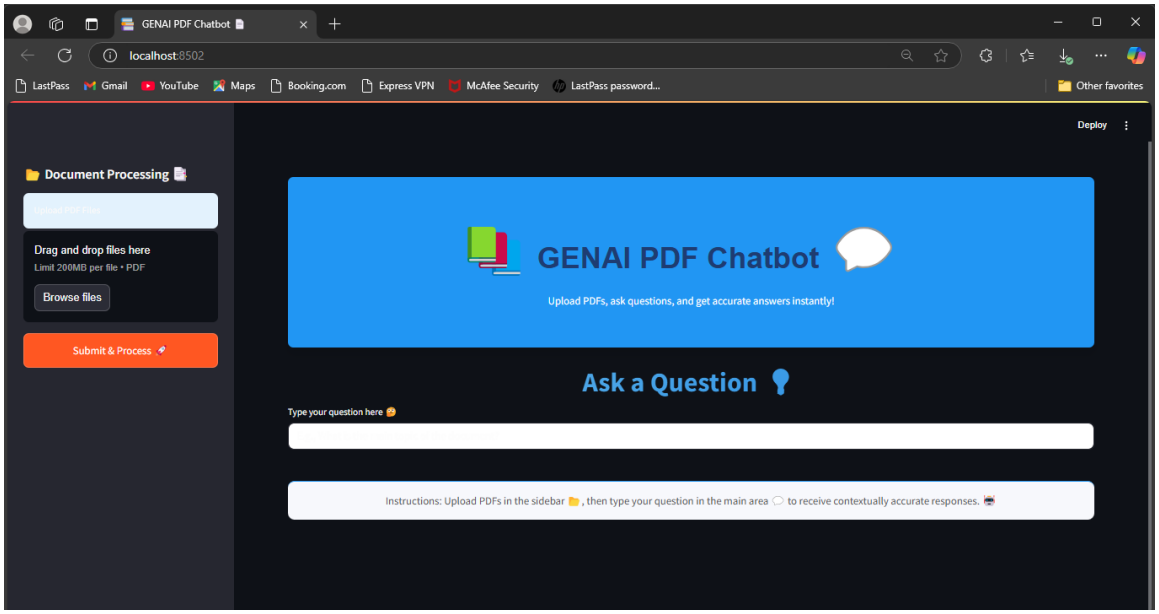
Input code:

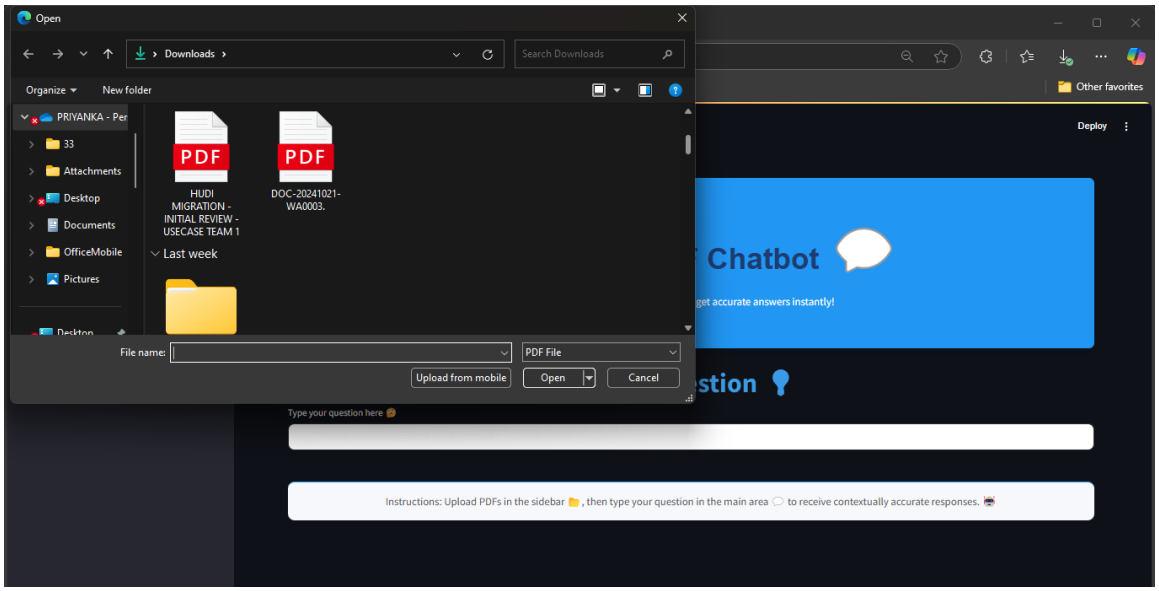


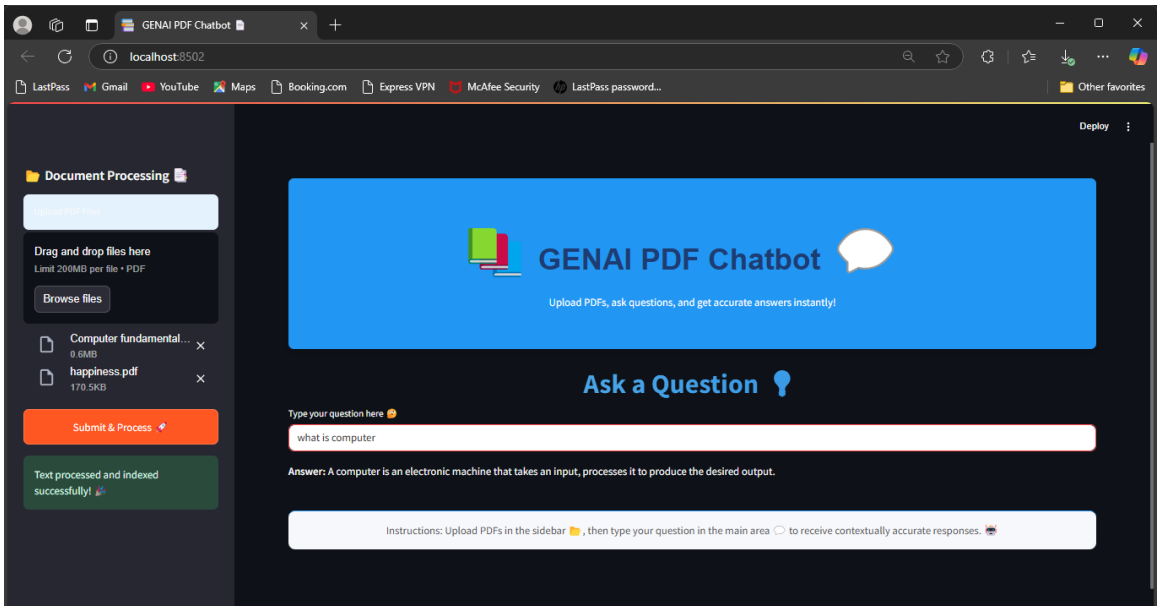


Output Snapshots:




GENAI PDF Chatbot
localhost:8502
LastPass
Gmail
YouTube
Maps
Booking.com
Express VPN
McAfee Security
LastPass password...
Other favorites
Deploy
Document Processing
Drag and drop files here
Limit 200MB per file • PDF
Browse files
Submit & Process
GENAI PDF Chatbot
Upload PDFs, ask questions, and get accurate answers instantly!
Ask a Question
Type your question here
Instructions: Upload PDFs in the sidebar , then type your question in the main area to receive contextually accurate responses.


Open
Downloads
Search Downloads
Organize
New folder
PRIYANKA - Per
33
Attachments
Desktop
Documents
OfficeMobile
Pictures
HUDI MIGRATION - INITIAL REVIEW - USECASE TEAM 1
DOC-20241021-WA0003.
Last week
File name:
PDF File
Upload from mobile
Open
Cancel
Other favorites
Deploy
Chatbot
Type your question here
Instructions: Upload PDFs in the sidebar , then type your question in the main area to receive contextually accurate responses.


GENAI PDF Chatbot
localhost:8502
LastPass
Gmail
YouTube
Maps
Booking.com
Express VPN
McAfee Security
LastPass password...
Other favorites
Deploy
Document Processing
Drag and drop files here
Limit 200MB per file • PDF
Browse files
Computer fundamental...
0.6MB
happiness.pdf
170.5KB
Submit & Process
Text processed and indexed successfully!
GENAI PDF Chatbot
Upload PDFs, ask questions, and get accurate answers instantly!
Ask a Question
Type your question here
what is computer
Answer: A computer is an electronic machine that takes an input, processes it to produce the desired output.
Instructions: Upload PDFs in the sidebar , then type your question in the main area to receive contextually accurate responses.

Example Use Cases:

1. Corporate Document Search:

    - Employees can quickly upload company policy PDFs and receive accurate answers to their queries.

2. Research Assistance:

    - Researchers can upload scientific papers, ask questions, and get detailed answers related to their research topic.

3. Educational Support:

    - Students can upload study materials and use the chatbot to clarify concepts.

Conclusion:

This GENAI PDF Chatbot demonstrates the potential of combining language models, vector databases, and streamlined web app development to create an interactive, intelligent assistant. By indexing document content and enabling conversational interactions, this solution transforms static text into a dynamic, searchable resource.