A PROJECT REPORT
on
"GenAI Chabot - Chat With PDF"

Submitted to
Bihar Engineering University

In Partial Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE AND ENGINEERING

BY

NAME	ROLL NUMBER
Shrejal Singh	21CSE15
Nikee Kumari	21CSE11
Kulsum Anjum	21CSE19
Bhanu Pratap Singh	21CSE06

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

SIWAN COLLEGE OF ENGINEERING & MANAGEMENT

SIWAN, BIHAR

May 2025

A PROJECT REPORT
on
"GenAI Chabot - Chat With PDF"

Submitted to
Bihar Engineering University

In Partial Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE AND ENGINEERING

BY

NAME	ROLL NUMBER
Shrejal Singh	21CSE15
Nikee Kumari	21CSE11
Kulsum Anjum	21CSE19
Bhanu Pratap Singh	21CSE06

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

SIWAN COLLEGE OF ENGINEERING & MANAGEMENT

SIWAN, BIHAR

May 2025

Bihar Engineering University

SIWAN COLLEGE OF ENGINEERING & MANAGEMENT

SIWAN, BIHAR 841226

CERTIFICATE

This is certify that the project entitled

"GenAI Chatbot - Chat With PDF"

submitted by

NAME	ROLL NUMBER
Shrejal Singh	21CSE15
Nikee Kumari	21CSE11
Kulsum Anjum	21CSE19
Bhanu Pratap Singh	21CSE06

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering) at Bihar Engineering University. This work is done during the year 2024-2025.

Date: 29/05/2025

Md. Saiffuddin
Project Guide

Acknowledgements

We are profoundly grateful to **MD. SAIFFUDDIN,** of **Affiliation** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We would like to thank everyone I learned with during this demanding time for making it joyful and efficient, and for answering my questions.

Lastly, I must express my profound gratitude to my parents for their unfailing support and encouragement throughout my years of study, especially for their sacrifices this year that allowed me to study abroad.

Shrejal Singh

Nikee Kumari

Kulsum Anjum

Bhanu Pratap Singh

ABSTRACT

In the evolving landscape of Generative AI and intelligent document processing, the ability to interact with unstructured data such as PDFs has become increasingly vital. The GenAI Chatbot – Chat With PDF, developed using Python, LangChain, Streamlit, and ChromaDB, serves as an innovative solution that empowers users to seamlessly interact with the contents of PDF files through natural language queries. This system addresses the challenge of information overload in static documents by enabling conversational access to document insights.

Built on top of a Retrieval-Augmented Generation (RAG) architecture, the chatbot leverages semantic search capabilities and large language models (LLMs) to provide contextually accurate responses based on PDF content. The project uses LangChain to orchestrate the flow between document loading, chunking, embedding generation, and querying. ChromaDB, a fast and scalable vector database, stores the embeddings generated from PDF chunks, allowing for efficient similarity searches based on user queries.

The frontend, designed using Streamlit, offers an intuitive and responsive user interface where users can upload PDF documents and interact with the chatbot in real-time. Behind the scenes, the system performs text extraction, chunking, and embedding using state-of-the-art models to ensure meaningful interaction. When a user poses a question, the most relevant text chunks are retrieved from the vector database and passed along with the query to the LLM for answer generation.

This project also emphasizes modularity, scalability, and extensibility. It demonstrates secure handling of file inputs, real-time feedback, and an interactive experience that minimizes user effort while maximizing value. By harnessing the power of modern NLP techniques, the chatbot not only enhances document understanding but also lays the groundwork for future integrations, such as multi-file chat, summarization, sentiment analysis, and voice-based querying.

In conclusion, the GenAI Chatbot – Chat With PDF represents a significant step forward in making static documents dynamically accessible. As AI continues to transform the way we interact with data, this project exemplifies the potential of combining retrieval-based systems with generative models to deliver smarter, faster, and more intuitive information access in academic, legal, corporate, and research domains.

The chatbot is particularly beneficial for students, researchers, and professionals who frequently work with lengthy documents, enabling them to quickly locate and understand specific information without manually scanning through pages. Its plug-and-play architecture makes it easy to integrate into existing workflows or enhance with new features like question summarization or document comparison.

Contents

List of Figures

Chapter 1

Introduction

In today's fast-paced digital era, the volume of unstructured textual data—particularly in the form of PDF documents—has grown exponentially. From research papers and legal contracts to technical manuals and academic notes, PDFs are the preferred medium for presenting well-formatted and organized content. However, navigating and extracting meaningful insights from large or multiple PDF documents can be time-consuming and inefficient, especially when users are looking for specific information. To address this challenge, the GenAI Chatbot – Chat With PDF project offers an intelligent solution that enables users to interact with PDF documents through natural language queries, simplifying document comprehension and enhancing productivity.

This project leverages the power of Generative AI and Retrieval-Augmented Generation (RAG) architecture to transform static PDFs into interactive knowledge sources. At the core of the system lies the LangChain library, which orchestrates the process of document loading, text chunking, embedding generation, and semantic retrieval. The embedded text chunks are stored in ChromaDB, a high-performance vector database designed for similarity search. When a user uploads a PDF and enters a query, the system retrieves the most relevant sections of the document and passes them to a Large Language Model (LLM) to generate a coherent and context-aware response. This enables users to "chat" with their documents as if they were speaking with a knowledgeable assistant.

The front-end interface of the chatbot is developed using Streamlit, a modern Python-based web framework known for its simplicity and interactivity. It provides a clean, user-friendly interface where users can upload their PDF files, ask questions, and receive instant answers—all in real time. The use of AJAX-like behavior in Streamlit ensures a smooth experience with minimal delays or page reloads. Additionally, the modular architecture of the system allows for easy extension, making it suitable for integrating more advanced features such as multi-document querying, summarization, and voice-based interaction in future iterations.

By combining natural language understanding with efficient information retrieval, the GenAI Chatbot empowers users from diverse domains—such as education, legal, corporate, and research—to access critical information quickly and accurately. Whether it's a student trying to extract key points from a study material or a legal professional scanning contracts for specific clauses, the chatbot simplifies the process and saves valuable time. Furthermore, the open-source nature of this project

encourages experimentation, collaboration, and enhancement, making it a valuable contribution to the field of AI-powered document intelligence.

In essence, the GenAI Chatbot – Chat With PDF is not just a tool but a smart assistant designed to enhance how we interact with knowledge locked inside documents. It embodies the growing synergy between artificial intelligence and real-world problem-solving, reflecting the shift toward more intuitive and accessible information systems in the digital age.

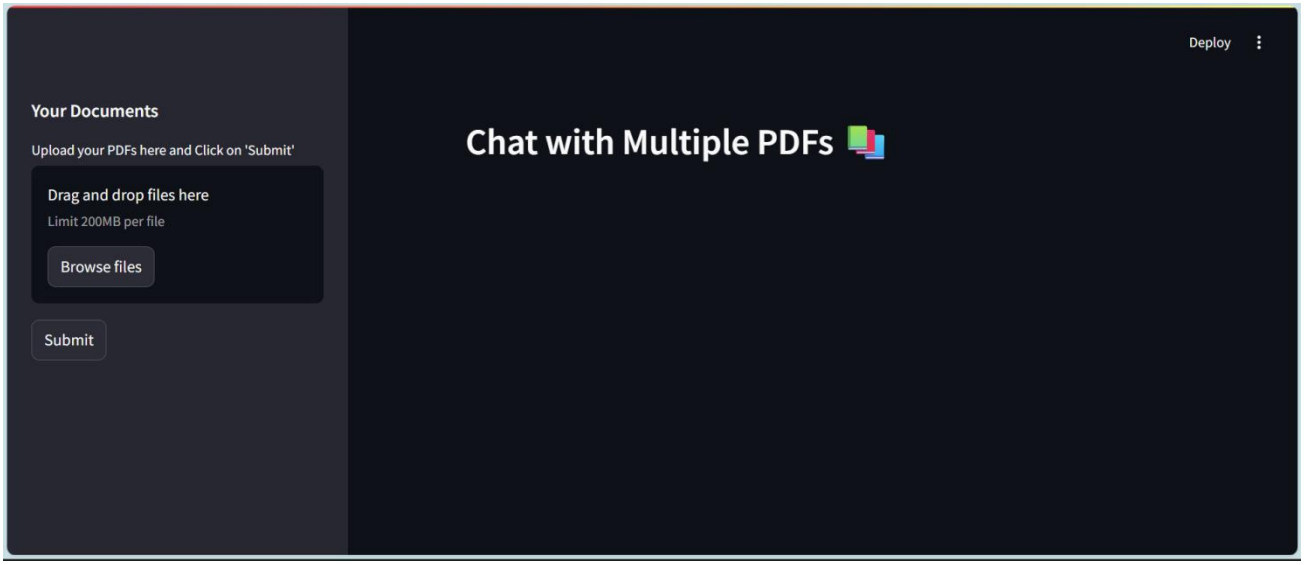

Fig. 1: Landding Page for GenAI Chatbot

Chapter 2

Objectives

The main objectives of the GenAI Chatbot – Chat With PDF project are:

1. To enable intelligent interaction with PDF documents: Allow users to ask natural language questions and receive accurate answers based on the content of uploaded PDF files, eliminating the need for manual reading and searching.

2. To automate document analysis and summarization: Reduce the time and effort required to extract insights from lengthy or complex PDFs through automated summarization and contextual response generation.

3. To implement Retrieval-Augmented Generation (RAG): Combine semantic search with generative AI to retrieve relevant content from the document and generate human-like responses using large language models (LLMs).

4. To store and search PDF content using vector embeddings: Break down the document into meaningful chunks, convert them into vector representations (embeddings), and store them in a vector database (ChromaDB) for efficient semantic search.

5. To build a user-friendly, interactive web interface: Use Streamlit to provide an intuitive and responsive front-end for document upload, query input, and response display, ensuring a smooth and engaging user experience.

6. To ensure modular and scalable system architecture: Design the project with modular components using LangChain and Python to support easy integration of additional features like multi-file support, real-time chat history, and role-based access control.

7. To support real-time question-answering without manual preprocessing: Allow the system to autonomously extract, preprocess, and index document content on-the-fly, minimizing user effort.

8. To enhance accessibility and usability across devices: Ensure the chatbot works seamlessly on various platforms such as desktops, tablets, and mobile devices through a responsive and lightweight UI.

9. To prioritize data privacy and security: Handle uploaded files and user inputs securely, ensuring that sensitive document content is not exposed or stored unnecessarily.

10. To provide a flexible foundation for future AI enhancements: Create a platform that can be expanded to include advanced features such as voice-enabled document querying, document comparison, chat history, multi-language support, and integration with external APIs or cloud storage systems.

In addition to these core objectives, the project also aims to democratize access to AI-powered document comprehension by offering an open-source, easy-to-use solution. By integrating real-time interactions, semantic understanding, and generative capabilities, the chatbot enhances productivity and simplifies knowledge extraction across domains like education, research, legal, and corporate sectors.
The system encourages continuous evolution, with potential for integration into enterprise systems, learning management platforms, or research tools. Ultimately, the GenAI Chatbot aims to redefine how users engage with static documents—making them dynamic, interactive, and intelligent.

Chapter 3

Tools and Technologies Used

The The GenAI Chatbot – Chat With PDF is built using a combination of cutting-edge technologies from the fields of web development, natural language processing, and vector-based search. These tools and frameworks work together to provide a seamless and intelligent user experience. The project architecture emphasizes modularity, responsiveness, and AI integration to allow users to interact meaningfully with PDF documents.

Backend Technologies

- Python : Python is the primary programming language used for developing the backend logic of the chatbot. Its extensive support for AI/ML libraries, natural language processing tools, and web frameworks makes it ideal for building intelligent applications. Python's readability and wide community support make development and maintenance efficient.

- LangChain : LangChain is a powerful framework designed to simplify the integration of large language models with external data sources. In this project, it orchestrates the flow from PDF ingestion to text chunking, embedding generation, semantic search, and LLM-powered response generation. LangChain enables building structured pipelines for document-based Q&A systems.

- PyMuPDF (fitz) : Used for extracting text content from PDF files efficiently. It supports accurate page-wise text parsing, which is crucial for indexing document content and maintaining context during retrieval.

- FAISS / ChromaDB (Vector Database) : ChromaDB is used to store vector embeddings of document chunks. It enables fast and accurate semantic search by matching the user’s query with the most relevant parts of the uploaded PDF. Vector similarity search is key to the retrieval-augmented generation (RAG) approach.

- Sentence Transformers / OpenAI Embeddings : Embeddings are generated using state-of-the-art models that convert text into numerical vectors representing semantic meaning. These embeddings are used to store and compare document content efficiently in the vector database.

Frontend Technologies

- Streamlit : Streamlit is a Python-based open-source framework used to create the interactive web application interface. It allows rapid prototyping of data-driven web apps with minimal code and includes widgets for file upload, chat input, and dynamic response rendering. Streamlit ensures a responsive, clean, and user-friendly experience.

- HTML & CSS : Basic HTML and CSS styling are utilized within Streamlit components to ensure well-structured formatting of PDF text output, chat messages, and user interactions.

Architectural and Design Tools

- Retrieval-Augmented Generation (RAG) : This AI architecture combines document retrieval with generative response generation. It retrieves relevant document sections using semantic search and passes them, along with the user's query, to a language model to generate context-aware answers.

- Embedding Models : Sentence-transformers or OpenAI embedding APIs are used to convert textual chunks into high-dimensional vectors. These embeddings capture semantic meaning and are stored in a vector database for similarity search.

- Virtualenv : Virtualenv is used to create an isolated Python environment for managing dependencies without affecting system-wide packages. This ensures reproducibility and stability across development setups.

- Pip : Pip, the Python package manager, is used to install and manage all project dependencies, including LangChain, Streamlit, ChromaDB, and PyMuPDF.

Version Control and Collaboration Tools

- Git : Git is used for source code version control, enabling tracking of changes, collaborative development, and rollback capabilities. The project is hosted on GitHub, allowing for open-source contributions and issue tracking.

- GitHub : GitHub serves as the project's code repository, facilitating public access, version control, and collaboration. It also hosts the project README, documentation, and deployment instructions.

These tools and technologies together form a robust ecosystem that supports the end-to-end functionality of the GenAI Chatbot – Chat With PDF system. From backend logic and document processing to AI-based response generation and frontend interaction, each component is selected to ensure performance, scalability, and an intuitive user experience. The project's modular design also allows for future expansion, such as integration with APIs, mobile platforms, or cloud-based storage and computing environments.

Chapter 4

System Features

The GenAI Chatbot – Chat With PDF system is designed to provide users with an intuitive, AI-powered interface that allows seamless interaction with the contents of PDF documents. The application uses Retrieval-Augmented Generation (RAG) to combine semantic search with generative capabilities, delivering accurate and context-aware responses. Below is a breakdown of key features available across different user functionalities.

4.1 Admin/Developer Features

Although the system is designed primarily for end-users, developers or admin users managing the application can utilize the following features:

- PDF Preprocessing and Chunking: Admins or developers can preprocess large PDF documents, breaking them into manageable text chunks for accurate embedding and efficient retrieval.
- Embedding Generation: The system uses sentence-transformers to convert text into semantic embeddings, which are automatically stored in a vector database (ChromaDB) for high-speed, similarity-based searching.
- Vector Store Management: The vector database (ChromaDB) is handled programmatically, and developers can clear, update, or inspect the stored document vectors for testing or fine-tuning.
- Custom LLM Integration: The system allows easy integration with different large language models via LangChain, giving developers the flexibility to switch providers like OpenAI, HuggingFace, or others.
- Environment Configuration: Secure and scalable environment management through .env file support enables safe handling of API keys and runtime variables.
- Modular Codebase: The project is structured into distinct functional modules (PDF loader, chunker, embedder, retriever, and generator), making it easy to enhance or modify individual components.

4.2 End-User Features

The system is built to offer an exceptional user experience through an intuitive Streamlit-based interface. Users can perform the following key actions:

- PDF Upload Interface: Users can upload one or more PDF documents directly through the interface. The system automatically extracts and processes the content in real time.
- Natural Language Query Input: A chat-like interface allows users to enter questions in plain English. The system interprets the query and responds based on the content of the uploaded PDF(s).
- Contextual Answer Generation: Using a Retrieval-Augmented Generation pipeline, the system identifies the most relevant chunks of text from the uploaded document(s) and sends them to the LLM to generate accurate, context-aware responses.
- PDF-Based Summarization (Optional): Depending on the LLM setup, the chatbot can also provide brief summaries of the PDF content, offering a quick overview of the document.
- Real-Time Response Rendering: The use of Streamlit ensures fast and seamless response generation without full page reloads, mimicking a conversational chat experience.
- Downloadable Chat Transcript (Future Scope): Users may be offered the ability to download their interaction history for reference or academic use in future versions of the system.

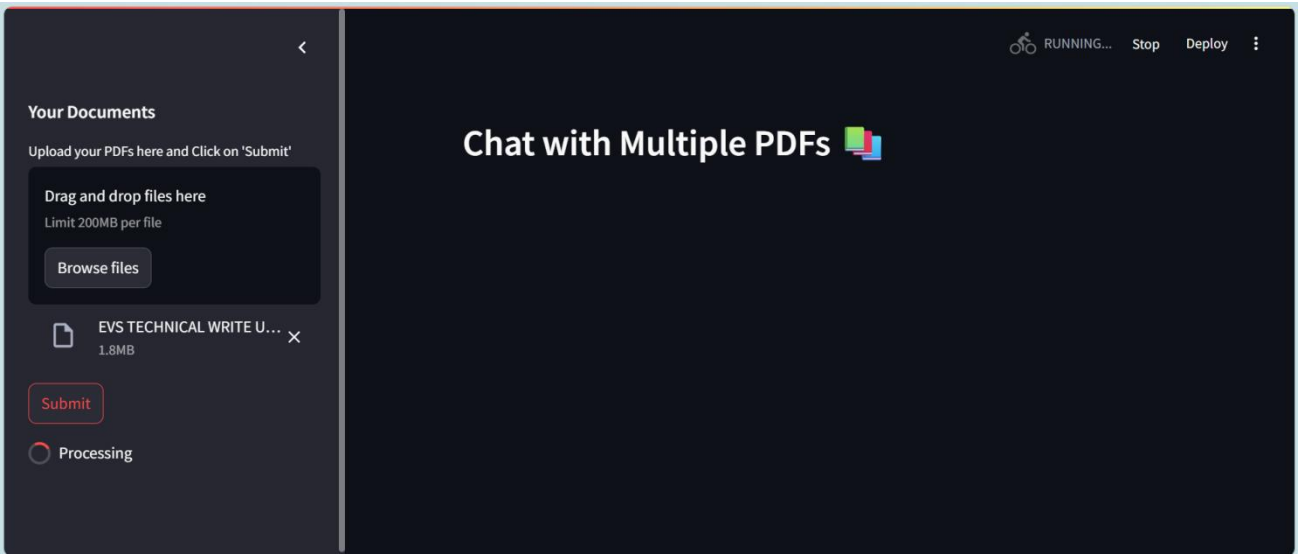

Fig. 2: File Uploading Section to upload any file

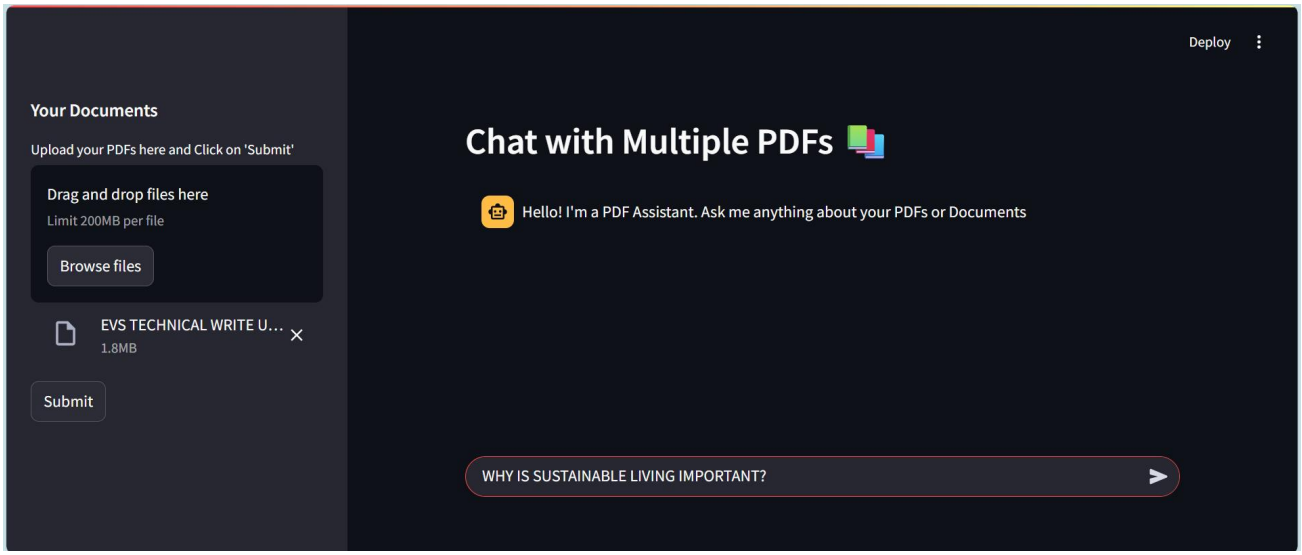

Fig. 3: User query box to enter the query

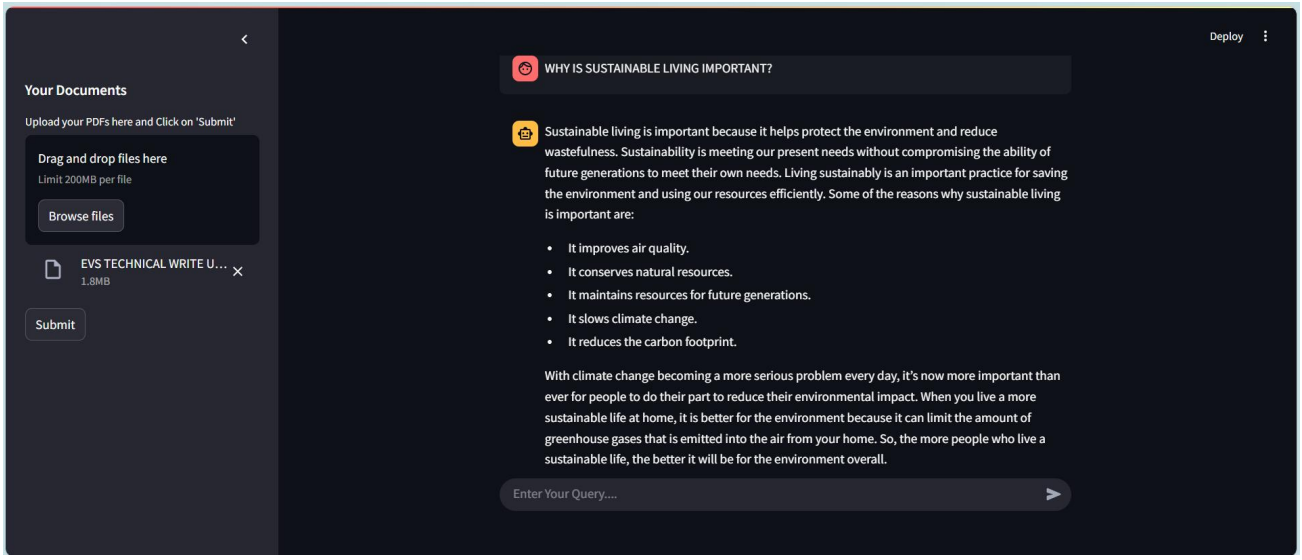

Fig. 4: Chatbot response for the user query

4.3 User Experience & Accessibility Features

- Responsive Design: The Streamlit app is mobile-friendly and adjusts to various screen sizes, offering flexibility across devices such as desktops, tablets, and smartphones.
- Error Handling and Feedback: The system provides clear feedback messages when errors occur (e.g., unsupported file types, missing queries), enhancing the reliability and usability of the application.

- Minimal Setup: Users can interact with the application locally or via cloud-hosted platforms with minimal setup—no advanced technical knowledge is required.
- Secure File Handling: Uploaded PDF files are handled securely in memory (not stored permanently), preserving user privacy and protecting sensitive content.

Chapter 5

System Design

5.1 Architectural Design

The GenAI Chatbot – Chat With PDF is architected using a modular and scalable approach that integrates modern web technologies with advanced AI capabilities. The system leverages Retrieval-Augmented Generation (RAG) architecture to combine document retrieval with language model-based generation, ensuring accurate, context-aware responses to user queries. The design emphasizes separation of concerns, extensibility, and maintainability, making it suitable for both small-scale and enterprise-level document intelligence solutions.

Key Components of the Architecture:

1. Document Preprocessing Layer (PDF Parsing & Chunking)

This layer is responsible for extracting and preparing raw content from uploaded PDF files.

- Text Extraction: Utilizes PyMuPDF (fitz) to extract clean and structured text from PDFs, preserving logical page flow.
- Chunking: Long texts are split into smaller, manageable chunks (e.g., 200–500 words) using intelligent chunking techniques to maintain context and improve retrieval accuracy.
- Metadata Handling: Optionally extracts document metadata such as title, author, and creation date for further use.

2. Embedding & Vectorization Layer (Semantic Representation)

This component converts text chunks into high-dimensional vector embeddings to enable semantic search.

- Embedding Models: Uses models like Sentence Transformers or OpenAI embeddings to represent textual meaning numerically.
- Vector Database (ChromaDB): Stores the embeddings in a vector index that supports similarity search via cosine distance or dot product.
- Index Optimization: Maintains an efficient index structure to ensure rapid, scalable retrieval even with large document collections.

3. Retrieval Layer (Semantic Search & Filtering)

This layer performs intelligent matching of user queries to relevant chunks of the document.

- Semantic Query Matching: Transforms user input into an embedding vector and retrieves top-k most similar document chunks from ChromaDB.
- Score Ranking: Ranks results based on similarity scores to ensure the most relevant contexts are passed to the generation model.
- Document Filtering: Supports multi-document filtering based on tags, filenames, or content type.

4. Generation Layer (LLM Integration & Response Crafting)

This component generates natural language responses using the retrieved context.

- Large Language Model (LLM): Uses OpenAI's GPT or other supported LLMs via LangChain to generate human-like responses.
- Prompt Engineering: Constructs structured prompts that include user queries and relevant document chunks to guide the LLM response.
- Answer Formatting: Presents responses in a readable, conversational style via the chat interface.

5. Application Layer (User Interface & Interaction)

The frontend, built using **Streamlit**, serves as the main interface for user interaction.

- File Upload: Allows users to upload one or multiple PDFs securely.
- Chat Interface: Enables users to type questions and view AI-generated answers in a real-time conversational format.
- Session Management: Maintains chat context throughout the session, enabling coherent multi-turn interactions.
- Responsive Design: Ensures smooth functionality across desktop, tablet, and mobile devices

Security Implementation

The system incorporates essential security practices to ensure safe document handling and user interactions.

- In-Memory File Handling: Uploaded PDFs are processed in memory, ensuring no files are stored on disk unless explicitly required.

- API Key Protection: Environment variables (.env) are used to securely manage and protect sensitive API keys and credentials.
- Input Validation: All user inputs and file uploads are validated to prevent injection attacks and malformed queries.
- Access Control (Optional): Can be extended to include authentication and role-based access in future versions.

Scalability and Extensibility

The system is designed to scale efficiently and support additional features through modular components.

- Model Flexibility: Easily switch between LLM providers like OpenAI, Cohere, HuggingFace, etc., using LangChain adapters.
- Multi-File Support: Architecture supports handling multiple PDFs in a single session with contextual separation.
- Cloud Deployment: The application can be deployed on platforms like Streamlit Cloud, Heroku, AWS, or GCP for broader accessibility.
- Future Enhancements:
    - Document summarization
    - Voice-based query input
    - Integration with cloud storage (Google Drive, Dropbox)
    - Chat history storage and download

5.2 Database Design

While the system does not rely on a traditional relational database for primary operations, it uses vector databases and optionally structured stores for managing document metadata, user queries, or sessions.

Key Data Structures and Components:

1. ChromaDB (Vector Store for Embeddings)

- Stores: High-dimensional embeddings of document chunks.
- Enables: Fast semantic similarity search based on user input.
- Supports: Filtering, scoring, and batch retrieval.
- Scalable: Can handle large volumes of document chunks efficiently.

2. Document Metadata (Optional Structured Storage)

For future expansion or multi-user systems, a lightweight document metadata table can be added using SQLite, PostgreSQL, or MongoDB to store:

- Filename
- Upload timestamp
- User session ID
- Document tags or categories
- Content summary or description

3. User Query Logs (Optional Logging Table)

- Query text
- Matched document ID
- Response text
- Timestamp
- Session metadata

These logs can be used for analytics, audit trails, and personalized improvements.

Database Integrity and Query Optimization

- In-memory operations: Ensure speed and privacy in lightweight use cases.
- Index refreshing: Keeps the vector database up to date when documents are reprocessed.

- Future Integration: Supports database abstraction if migrating to external storage (e.g., Redis, PostgreSQL, or cloud-based vector stores like Pinecone or Weaviate)

Scalability and Maintainability

The Django ORM provides an abstraction layer that simplifies database interactions, making the system highly maintainable and scalable.

- Migration Support: Django's migrations allow database schema updates without data loss.
- Multiple Database Compatibility: Though SQLite is used, the system can easily switch to PostgreSQL, MySQL, or Oracle for large-scale deployments.
- API and Third-Party Integration: The database structure supports RESTful APIs, enabling integration with external applications, analytics tools, and mobile platforms.
- Cloud Deployment Readiness: The database can be deployed on AWS RDS, Google Cloud SQL, or Azure SQL, ensuring high availability and performance.

The GenAI Chatbot – Chat With PDF system follows a clean and flexible architecture that bridges advanced language models with real-world document interaction. Through its modular design, it offers a scalable and extensible platform that can evolve into a full-fledged AI-powered document assistant capable of transforming how users engage with and extract insights from unstructured data.

Chapter 6

Installation and Setup Guide

This section provides a step-by-step guide to installing and setting up the GenAI Chatbot - Chat With PDF using Python.

Prerequisites
Before proceeding with the installation, ensure that your system meets the following requirements:

- Python (3.x) installed – Download from Python.org
- pip (Python package manager) – Comes pre-installed with Python
- Streamlit Framework – Installable via pip
- Google Gemini API Token
- Git – Required for cloning the project repository
- VS Code or Any Code Editor
- SQLite (default Django database) or PostgreSQL (optional)

Step 1: Clone the Project from GitHub
First, open your terminal or command prompt and navigate to your preferred directory. Run the following command to clone the repository:

```
git clone <repository-url>
cd Chat-With-Pdf
```

Replace <repository-url> with the actual GitHub repository URL.

Step 2: Create a Virtual Environment (Recommended)
Setting up a virtual environment ensures dependencies do not interfere with system-wide Python packages.
Run the following commands:

```
python -m venv venv
```

Activate the virtual environment:

```
venv\Scripts\activate
```

Step 3: Install Required Dependencies
Once inside the virtual environment, install the required dependencies by running:

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

If requirements.txt is not available, manually install Django:

```
pip install streamlit
```

Step 4: Run the Development Server
Start the Django server using:

```
Streamlit run app.py
```

Open http://127.0.0.1:8501/ in your web browser to access the application.

Chapter 7

Future Enhancement

The GenAI Chatbot – Chat With PDF is a powerful prototype that simplifies user interaction with unstructured PDF content using Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs). While the system currently enables single-document Q&A with real-time interaction, several enhancements can be implemented to improve usability, scalability, and intelligence. These improvements focus on multimodal access, cloud scalability, multi-document support, AI-driven analytics, and broader platform integration to transform the tool into a full-fledged document intelligence platform.

1 Multi-Document Support and Chat History

Enhancing the Contextual Understanding Across Documents

- Multi-PDF Upload and Chat Context Handling: Users will be able to upload and interact with multiple PDFs in a single session, with the chatbot differentiating and referring to the right context dynamically.
- Persistent Chat Sessions: Save chat history for each document session to allow users to revisit previous queries and answers.
- Document-Based Tagging: Organize documents by tags or topics, enabling better filtering and targeted information retrieval.

2 Mobile and Desktop Application Development

Cross-Platform Access for Enhanced Usability

- Mobile App (Flutter/React Native): A mobile-friendly version will allow users to query documents directly from their smartphones, making document analysis more accessible on the go.
- Desktop Client (Electron or Tauri): For researchers and professionals working offline, a native desktop app can provide a richer and more stable interface.
- Offline Document Processing: Include the ability to process and query PDFs locally without the need for an internet connection, useful in secure or low-connectivity environments.

3 Integration with Cloud and External Platforms

Scalable Infrastructure and Storage Capabilities

- Cloud Storage Integration: Integrate Google Drive, Dropbox, and OneDrive for easy access to personal or team document repositories.
- Cloud Deployment (AWS, GCP, Azure): Enable global access with elastic compute scalability and real-time collaboration.
- API as a Service: Expose the functionality as RESTful APIs or GraphQL services for easy integration with CRMs, document management systems (DMS), and LMS platforms.

4 Advanced AI Features

Moving Beyond Q&A to Intelligent Document Understanding

- Summarization and Keyword Extraction: Add features to auto-summarize documents and highlight key points or topics based on user queries.
- Sentiment and Intent Analysis: Extract sentiment from legal, HR, or review documents, enabling context-aware emotional tone detection.
- Voice-Based Interaction: Integrate speech recognition to allow users to ask questions using voice commands.
- Multilingual Support with NLP Models: Enable document querying in multiple languages, expanding accessibility globally.

5 UI/UX and Accessibility Enhancements

Improved User Experience Across Devices

- Customizable Themes and Layouts: Allow users to personalize the UI (dark/light mode, font size, etc.).
- Real-Time Highlighting in PDF Viewer: Highlight exact document segments used for generating a response.
- Right-to-Left (RTL) Language Support: Enhance localization by supporting languages such as Arabic and Hebrew.

6 Security and Privacy Enhancements

Ensuring Secure Document Handling and API Usage

- End-to-End Encryption for File Uploads: Secure transmission and temporary in-memory processing of files.
- Role-Based Access Control (RBAC): In a multi-user platform, assign roles like admin, editor, or reader to protect data access.
- Token-Based Authentication (JWT/OAuth2): Enable secure authentication for cloud or enterprise-level deployments.
- Audit Logging: Log user activity for transparency, traceability, and compliance in regulated environments.

Containerization with Docker & Kubernetes

Implementing Docker containerization ensures seamless deployment, portability, and maintainability.

- Microservices Architecture: Divide the application into independent services for better performance.
- Kubernetes Orchestration: Automate scaling, load balancing, and deployment across cloud instances.

References

https://medium.com/firebird-technologies/chat-with-your-pdfs-using-langchain-e57866b7926d

https://aws.amazon.com/what-is/retrieval-augmented-generation/

https://www.ibm.com/think/topics/generative-ai