Abstract

This interactive legal chatbot created via Streamlit makes use of cutting-edge **Large Language Models (LLMs)** in conjunction with **Retrieval-Augmented Generation (RAG)** to provide intelligent and easy-to-use legal advice. This technology aims at making legal information accessible to laypersons. LegalAI encapsulates varied areas including general legal knowledge, women-oriented laws, help for students, government schemes, and the Bharatiya Nyaya Sanhita, 2023.

LegalAI integrates technology with LangChain and FAISS for semantic search and supports four powerful LLMs: **LLaMA 3, DeepSeek R1, Mixtral, and Command R** using API access through platforms like Groq, Together AI, and Cohere. Users pick the model they prefer in receiving accurate, contextual, coherent responses.

Legal documents PDF files of laws, schemes, and policies are processed in LangChain text-splitting, embedded using Google Generative AI Embeddings, and indexed into five **FAISS** vector stores one for each legal category for domain-specific semantic retrieval.

Supabase provides user authentication and also manages the history of chat sessions safely. Thanks to the Streamlit session state, the chatbot is multi-turn, meaning users can carry on conversations without losing context. Chat transcripts could also be exported to beautifully formatted PDFs for reference or academic purposes using the FPDF library.

LegalAI is the modular construct, built to scale, providing safe API key management via .env files and deployable from the cloud. Functional and experimental testing show that it behaves efficiently and responsively, and concerning legal accuracy

LegalAI is a scalable solution to connect the gap between legal systems and the general public in a way that makes legal assistance comprehensible, accessible, and impactful, elevating the LLMs to a level downstream for elegant design, vector search, etc.

Table of Contents

List of Tables

List of Figures

`Chapter 1: Introduction

1.1 State of Art Developments

The artificial intelligence (AI) sector has undergone considerable changes in the past few years and the development of such models with an impressive intake has boosted their speed ability in understanding, generating, and reasoning with natural languages. Lurers like OpenAI's GPT, Cohere's Command models, Groq's installations of LLaMA, and DeepSeek have made a great deal of change in natural language processing by affording access to comprehension of human language by machines in flabbergasting fluency and clearness.

Particularly beneficial are the applications of these technologies in the legal research and advisory context. Common legal research traditionally involves the tedious exercise of poring over impenetrably dense legal texts and statutes and case laws-a time-consuming activity that is sundry. All of these things can now be considerably accelerated and simplified through the combined use of LLMs, so that people come up with sensible, context-aware answers to legal issues.

There are further capabilities of the LLMs with such frameworks as LangChain that orchestrate the language models with other outside tools. LangChain, for example, makes possible advanced features like conversational memory, document retrieval, and chaining of tools all important in building intelligent specific domain assistants. Using it in conjunction with FAISS, for example for fast vector-based document retrieval, and embedding models such as GoogleGenerativeAIEmbeddings, one can semantically search through legal data and retrieve relevant meaningful answers to user queries. It Opens a New Age in Accessible Legal Support: The New Synergy between Cutting-Edge LLMs and AI Infrastructure.

1.2 Motivation

The Indian legal landscape is extensive, multidimensional, and deeply intricate. The body of law extends from constitutional provisions to sectoral legislation and general welfare schemes of the government. While the judiciary is responsible for upholding justice with a view vested in interpreting justice, legal knowledge remains inaccessible to an average citizen. Legalese, absence of digitalized resources, and the lack of simplified guidance have only served to widen this chasm between law and laypersons.

Ordinance studies are hard to get due to the unavailability of research present and few if any, modern laws such as the Bharatiya Nyaya Sanhita apply to students. Likewise, those women wanting to understand their rights or citizens wishing to know about the government schemes are often caught up in a tangle of bureaucracies or old web portals. A strong sense of need for making legal education accessible to all, understandable, and intelligent are considered the foregrounds of this project.

LegalAI wants to be the solution where the users can converse with legal content. It wants to render comprehendible formats to convoluted texts while facilitating an interactive experience depending upon the user's domain of inquiry-women's rights, student help, general law, and government schemes.

1.3 Problem Statement

Even though the advances in AI and digital tools have taken place, the legal domain still lacks intuitive systems which can provide personalized, structured, and dynamic legal information. Existing legal chatbots function more or less like rules-based and static systems rather than domain-specific units. They neither have the feature of retaining context nor providing in-depth explanations or adopting a user's conversation theme.

In addition, there is no existing real-time document retrieval based on any semantics, and neither does any allow flexibility in selecting the AI model used or tracking of chat history. The two aspects combined lead to bad user experience and trust deficits in the use of AI legal systems.

This project is intended to fill the void created by a full-fledged multi-domain legal chatbot-to-answer queries-and also retrieve legal data from relevant documents, support conversational memory, and deliver a seamless user experience through an intuitive interface.

1.4 Objectives

The objectives of this project are fourfold:

- To design and develop a multi-modal AI-enabled chatbot that allows users to have legal discussions, maintaining contextual continuity.
- To integrate vector databases using FAISS for efficient document-oriented retrieval of legal information.
- To offer domain-specific facilitation across areas such as General Legal Topics, Women-Centric Laws, Government Welfare Schemes, Bharatiya Nyaya Sanhita (BNS), and Student Assistance.
- To set up authentication and chat history management through Supabase for a personalized user session.
- To allow users to export their chat conversations as professionally formatted PDF files for later reference and documentation.

Together, these objectives are geared towards ensuring that LegalAI is not merely a smart chatbot but a functional legal companion that can be scaled across multiple use cases.

1.5 Methodology

The entire architecture is being developed in Python due to its robust capabilities in software development for AI integration and web development. Streamlit as a UI framework allows rapid and efficient development and deployment of interactive web-based apps.

LangChain's PyPDFDirectoryLoader enables document ingestion and hence inputting legal documents in the PDF format for reading. The RecursiveCharacterTextSplitter

processes these documents to convert them into vector embeddings by GoogleGenerativeAIEmbeddings. Embedded vectors are indexed and kept in FAISS for efficient similarity-based retrieval during active user interaction.

The ConversationalRetrievalChain provided by LangChain is the brain behind chat logic and memory retention between conversations. It undertakes retrieval-augmented generation (RAG) of documents wherein the LLM is conditioned with context derived from user history as well as retrieved legal documents.

Users sign up and log in via Supabase, which is also responsible for the persistent storage of the user's chat history. This allows users to retrieve and export past conversations using the FPDF library.

1.6 Summary

This chapter lays down the fundamental motivations, objectives, and technological scenarios that triggered the establishment of LegalAI. It highlighted the need for intelligent legal assistants to fill the knowledge gap between legal documentation and end users through natural language interaction, especially considering that the end-user would be most likely doing the search in natural languages.

Diving deeper into the core technology making up the critical ingredient of the said system, namely Large Language Models, the LangChain framework, FAISS vector stores, as well as embedding techniques-all forming the backbone of the intelligent conversational capabilities of LegalAI will be found in the subsequent chapter.

Chapter 2: Overview of LLMs, LangChain, FAISS, and Embeddings

2.1 Introduction

This chapter presents an in-depth overview of the core technological components that serve as the backbone of the LegalAI system. These components work in unison to enable an interactive, intelligent, and context-aware chatbot that can process legal queries efficiently. The technologies explored in this chapter include **Large Language Models (LLMs)**, the **LangChain framework**, **FAISS vector storage**, and **embedding models** that drive semantic understanding and retrieval.

Together, these technologies facilitate natural language understanding, memory-aware conversations, and domain-specific legal information retrieval—making LegalAI a powerful tool for legal education and awareness.

2.2 Large Language Models (LLMs)

Large Language Models are deep learning models built on huge amounts of text data. They have a grasp of human linguistic structure, grammar, and semantics, which allows them to reason and produce coherent contextually valid answers even for prompts of substantial complexity. LLMs are the actual heart of the LegalAI's response generation engine.

The model selection thus fits with the user's selection of response length, richness of style, and depth of reasoning. These models are the ones integrated into LegalAI:

- **LLaMA 3 (Groq):** The state-of-the-art transformer model focused even more on reasoning-heavy tasks and was able to generate highly articulate and structured responses specifically desired for legal queries.
- **Mixtral (Together AI):** A mixture-of-experts model intended for open-ended conversations. It generates quite fast and reasoned succinctly.

- **DeepSeek R1 (Groq):** A distilled, light version of the LLaMA architecture. This is because it sacrifices some amount of complexity for performance and runtime speed.
- **Command R (Cohere):** This is a fine-tuned instruction-following model meant for interactive-type work. It is the best when clear step-by-step reasoning and a smooth conversational flow are needed.

These models are accessed through external APIs and seamlessly integrated into the chatbot interface, allowing users to switch between them dynamically in the Streamlit sidebar.

2.3 LangChain Framework

LangChain is an open-source Python framework that efficiently links LLMs with other tools like memory, document retrievers, databases, APIs, and others. By this work the LegalAI system employs LangChain to mediate user input from LLM processing and retrieval of related information.

Some important components of LangChain used in LegalAI are as follows:

- **ConversationalRetrievalChain:** This creates a chain for conversational format to include a retriever within a language model in order to take into account relevant documents from the vector store for guiding the LLM generation of a response more accurately using information from the documents.
- **ConversationBufferWindowMemory:** storage of the last interactions which would be between him and his assistant and keeps that with it for reference during chat turns.
- **Retriever-based QA pipeline:** LangChain provides the basic structure for creating pipelines connecting actual legal documents to the LLM for answering specific questions.

Such an architecture is made possible by LangChain for the chatbot through extensibility and modularity within which easy development is possible.

2.4 FAISS Vector Store

FAISS (Facebook AI Similarity Search) is a library that has been developed by Facebook AI Research to perform similarity measures and clustering of dense vectors efficiently. In LegalAI, FAISS has been used for storing and searching vector representations of legal documents.

How FAISS works as part of the project:

- It parses the legal PDFs and segments them into chunks.
- Then embeds those chunks into a single high-dimensional vector using GoogleGenerativeAIEmbeddings.
- Put those vectors into FAISS indices for retrieval as fast as really possible.

In order to provide focused and relevant answers in the legal sense, LegalAI maintains different FAISS vector stores per domain:

- General Legal Data
- Women Specific Laws and Rights
- Government Schemes and Welfare Programs
- Bharatiya Nyaya Sanhita (BNS) 2023
- Legal Help and Resources for Students

So when the user selects a category on the chatbot, the appropriate FAISS store would be grounded and loaded for the retrieval of documents in that domain.

2.5 Embedding Models

Embeddings represent the conversion of textual data into numbers that computers can process. They capture the semantics of words, phrases, and documents, placing them within a high-dimensional vector space.

LegalAI employs the GoogleGenerativeAIEmbeddings model (embedding identifier: embedding-001) for embedding generation; it is particularly potent for representing

endangered animal semantics at the sentence or paragraph level, and hence can serve well to encode legal documents.

The PDF documents will first be processed from embeddings by LangChain's RecursiveCharacterTextSplitter; large texts will be split into contextually meaningful chunks with overlapping-maintaining context from the previous chunk-to ensure continuity. The subsequent chunks are embedded and stored in FAISS so that the chatbot can retrieve information with a high degree of relevance for a given user query based on the user's query's semantic similarity.

2.6 Summary

The four major technological pillars that empower LegalAI were described here:

- Large Language Models (LLMs) generate intelligent and dynamic responses.
- LangChain is the foundation that integrates memory, models, and retrievers.
- FAISS provides scalable and efficient vector retrieval for legal documents.
- Embedding models convert legal text into semantic vectors, deepening the understanding of user queries and matching them with actual content.

Altogether, these tools help LegalAI go on beyond static chatbot behavior and offer context-rich document-backed conversational legal assistance.

Chapter 3: Software Requirements Specification for Legal AI Chatbot

3.1 Functional Requirements

The functional requirements specify the essential features that the LegalAI chatbot must convey to act in the expected role of an intelligent legal assistant. These embed the following:

- **User Authentication:** The system must support secure user sign-up and login functionality, ensuring that user data is protected and only accessible by the authenticated individual.
- **Conversational Interaction:** The chatbot must naturally talk in a human-like style while remembering the most recent conversation to answer in a multi-turn conversation.
- **Section-wise Query:** The users should have the option to query under such specific legal areas as General Legal Chat, Women Laws, Government Schemes, Bharatiya Nyaya Sanhita (BNS) 2023, and Student Help. Each section is related to a specific knowledge base from which information can be retrieved.
- **Vector-based Document Retrieval:** It should be possible to retrieve the contextually relevant chunks of documents from FAISS vector stores based on semantic content of user queries.
- **Chat History Management:** Each User interaction has to be saved persistently in a database. Users should now be able to see their previous chats in the application.
- **PDF Export Functionality:** There must be an option for this chatbot through which the user can download the PDF of their chat history for future use or academic/legal purposes.

3.2 Non-Functional Requirements

Non-functional requirements refer to those parameters which lay emphasis on performance, scalability, security, and usability of the database.

- **Response Time:** The chatbot should generate a response for the user 2-4 seconds after receiving a query in normal operating conditions.
- **Data Security:** All user information is logged in chat; any credential-related information must be stored over secure connections in an encrypted database with access available to authenticated users only.
- **Scalability with Open API options:** The architecture should be modular, allowing easy integration with different LLM APIs (Groq, Together, Cohere, etc.) for scaling across multiple requests and model preferences.
- **User Interface Responsiveness:** The application shall be developed responsive, providing an unhindered parallel experience through devices such as desktops, tablets, and mobile phones.

3.3 Hardware Requirements

To facilitate smooth application development and testing environments, the following hardware setup is recommended: A personal computer or server with a minimum of 8GB RAM to run the application locally.

- A stable internet connection needs to send communications to external APIs and Supabase backends.
- For production or public deployment, hosting on a cloud platform such as Streamlit Cloud or Render is recommended for better scalability.

3.4 Software Requirements

The following software stack is needed for the development and deployment of the LegalAI system:

- **Python (v3.10 or higher)** – The core programming language is used for the backend and the Streamlit app.
- **Streamlit** – Web framework for creating the graphical user interface.
- **LangChain** – Framework for chaining LLMs and dealing with memory/retrieval logic.
- **FAISS** – Vector database for storing and searching embedded chunks of documents.
- **Supabase Client Library** – used for authentication, database operations, and session handling.
- **Google Generative AI SDK** – for embedding models to convert legal documents into vector representation.
- **FPDF** – A library used to convert chat histories into downloadable PDF format.
- **python-dotenv** – a practical utility to secure the management of environment variables through .env files.

3.5 Summary

This chapter has captured both the functional and non-functional requirements, which dictate what expected behaviors and what performance standards LegalAI would have to meet. The same chapter also contains the hardware prerequisites and the entire software stack of what was needed to develop and run the application. The next chapter pans towards the architectural design, with a detailed dissection of each of the software modules.

Chapter 4: Design for Legal AI Chatbot

4.1 High-Level Design

Each sub-system has been created in an extensible, fully modular, service-oriented manner so that every single component can easily be run up-scaled and prepared for updates. Thus, it talks in terms of ease-of-use user-friendly, along-with the full integration of new AI tools into the system.

High-level system components may consist:

- **The Frontend Layer (Development by Streamlit UI):** This is the inlet through which users feed their requests and get responses from the bot. Here, the user selects the area of legal classification and views the history of chat.
- **Authentication Service (via supabase):** This service manages registration and logging-in of users and blocks any unauthorized access to the application from non-user identity of the chat app.
- The Model Management Layer provides on-the-fly choice of conversation and invocation of external language models. It incorporates the LegalAI LLM-provider under Groq, Cohere, and Together AI.
- **Vector Store Layer:** Stores indexed legal documents in vectorized formats. This will allow efficient semantic searches and retrieval of very relevant information based on user queries.
- **Document Ingestion Pipeline:** It will load, preprocess, separate, and embed legal PDF files, which will later feed for vector retrieval in FAISS.
- **Memory Management (LangChain):** It ensures the short-lived coherent memory gives context-relevant inputs to a conversation for multi-turn interactions.
- **Chat History service:** It logs each user's action into a secure database. As a result, retrieved past conversations or PDF exports become possible.

4.1.1 System Architecture

The architecture integrates all components into a streamlined pipeline. User login submission triggers the following orchestration:

- From the Streamlit UI to the LangChain conversational chain go all inputs.
- LangChain retrieves relevant documents using memory and a selected vector store.
- The context retrieved and the query from the user are passed to the chosen LLM.
- The response from the model is presented to the user in the UI and saved in the chat history.
- The user can export the conversation in PDF form.

This flow guarantees secure, efficient, and intelligent handling of queries with smooth model-data integration.

4.2 Detailed Design

LegalAI is designed with the objective of improving clarity of function and separation of modules. Each part of the system consists of a separate function and a specific interface for communication with one another.

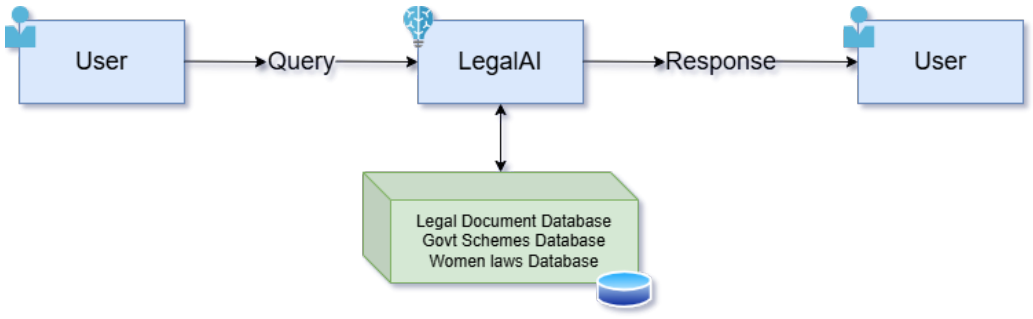

Figure 1: Level-0 Data Flow Diagram

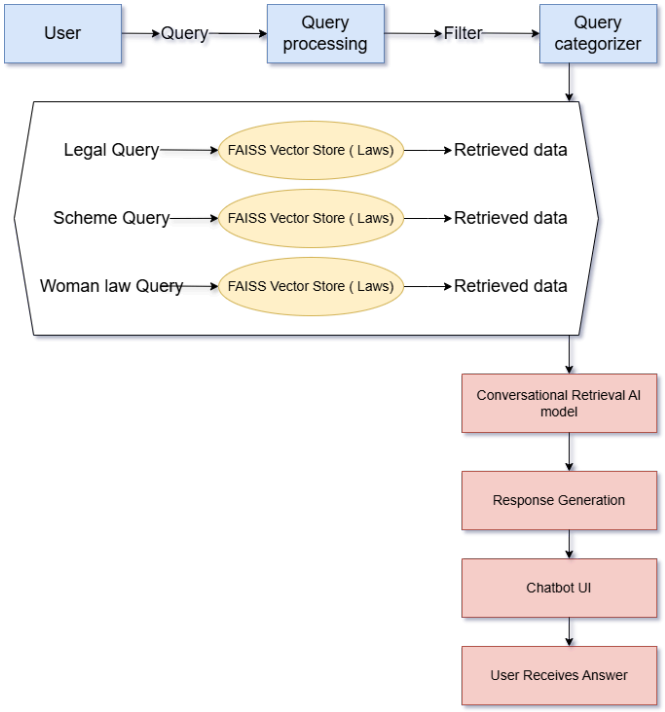

Figure 2: Level-1 Data Flow Diagram

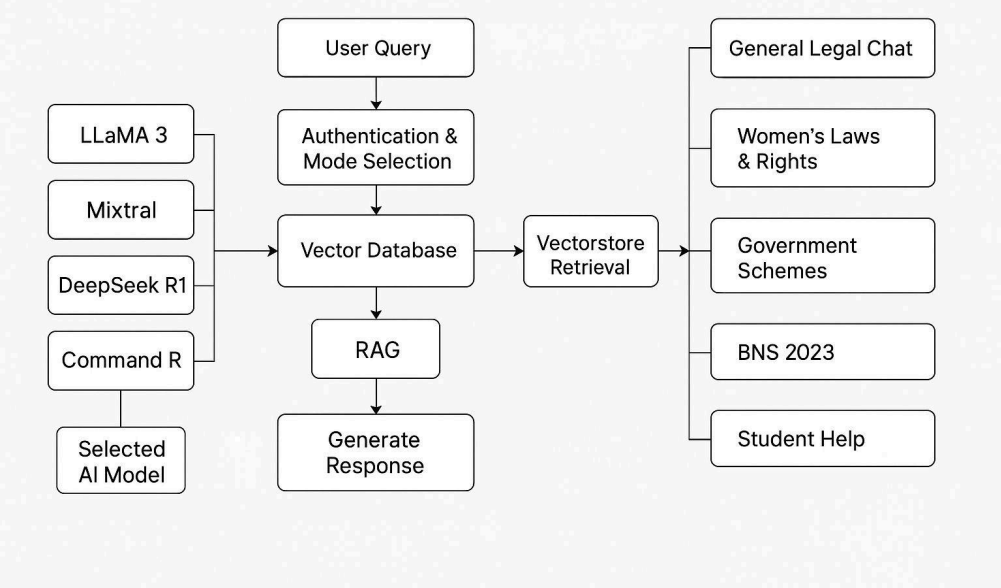

Figure 3: Level-2 Data Flow Diagram

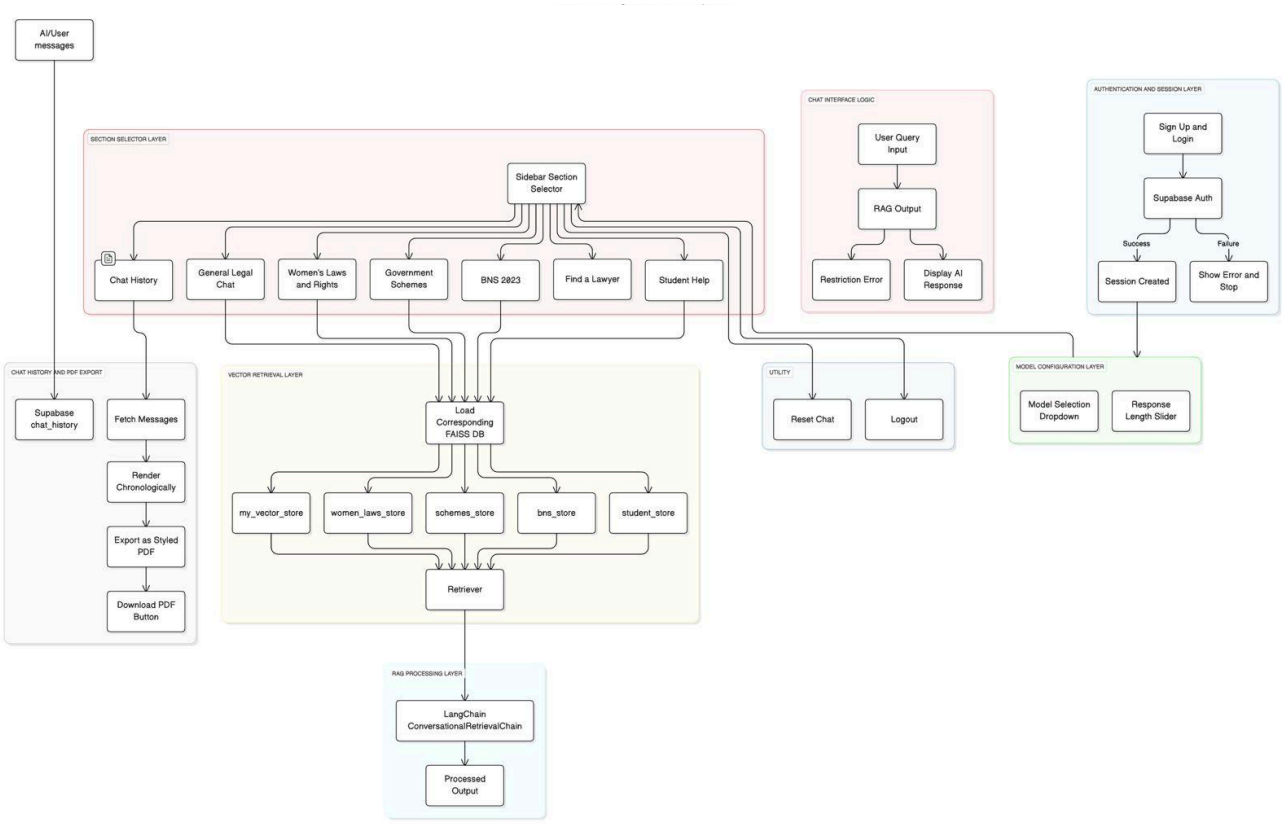

Figure 4: Level-3 Data Flow Diagram

4.2.1 Structure Chart

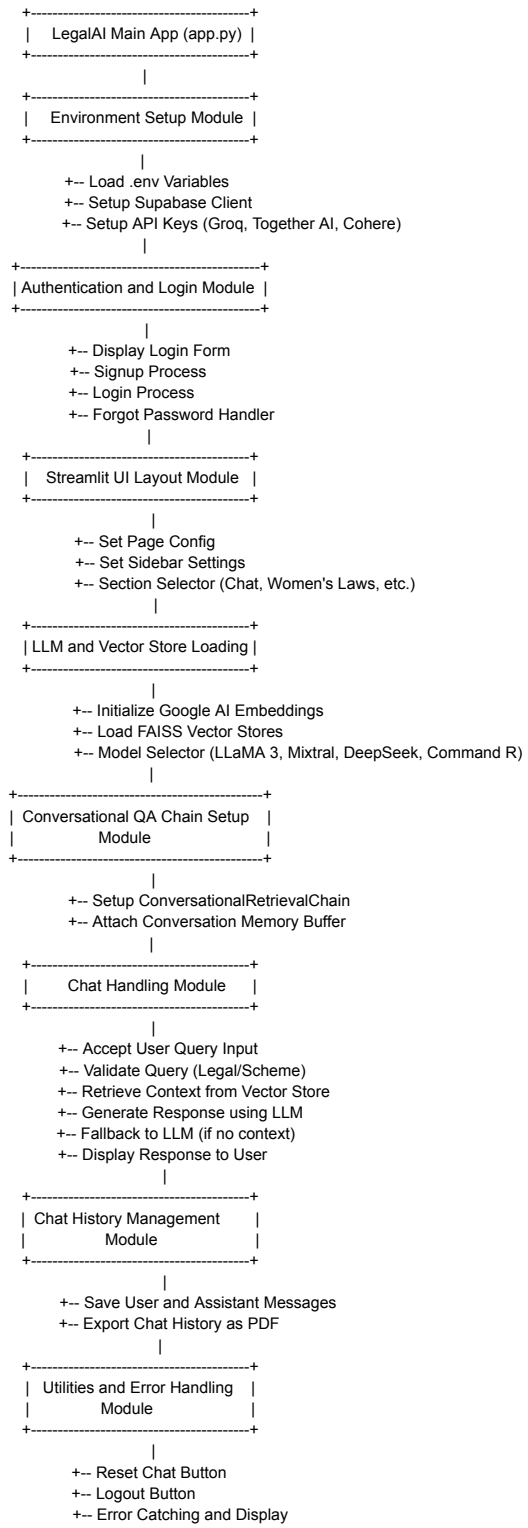

Figure 5: High-Level Workflow

4.2.2 Functional Description of the Modules

Table 1: Modules Used with Description

Module	Description
Auth Module	Manages user registration and login using Supabase. Ensures secure authentication and user data privacy.
Streamlit Interface	Provides the graphical user interface. Handles user input, model selection, and displays responses.
Chat Controller	Coordinates message handling, stores conversation context, and manages chat switching and resets.
Retriever Selector	Loads the relevant FAISS vector store based on the domain selected by the user (e.g., Women, BNS).
LLM Model Handler	Interfaces with Groq, Cohere, and Together APIs to run LLMs for generating responses.
LangChain Memory	Maintains recent chat memory using ConversationBufferWindowMemory to ensure context retention.

ConversationalRetrievalChain	Connects memory, retrieval, and the selected LLM to produce accurate, document-informed replies.
PDF Export Module	Uses FPDF to export formatted chat transcripts, preserving user interaction for offline reference.
Ingestion Script	Implemented in ingestion.py, this script processes and embeds PDF documents into FAISS vector stores.

Chapter 5: Implementation for Legal AI Chatbot

5.1 Programming Language Selection

LegalAI's development is entirely in the Python programming language. It was selected by the developers for being simple, flexible, and with a rich ecosystem of libraries that facilitate artificial intelligence, natural language processing, and fast web development. The strong ability of the language to support third-party modules and to work seamlessly with APIs make it excellent for the development of scalable AI applications.

Some major Python libraries that were employed in this project include the following:

- Streamlit: It is used to build the graphical user interface (GUI) and to deploy the chatbot as a web app).
- LangChain: It allows for creating modular chains between the language model and memory and document retrieval logic.
- supabase-py: A library that provides user authentication and a secure way to implement functions against the Supabase backend.
- FPDF: Exports the chat history into a well-defined PDF document.
- dotenv: Allows management of environment variables for stashing API keys and configurations safely.

All these tools together enhance the development of a highly interactive legal assistant.

5.2 Platform Selection

The two implementation environments for LegalAI are development and deployment:

Development Platform: All code was developed locally in Visual Studio Code (VS Code) under Python 3.10+, runtime and necessary libraries for the Virtual environment. All other development tasks like writing code, debugging, version control, and testing were carried out in this environment.

- Execution Platform: The application interface has been constructed through Streamlit, which allowed the chatbot to run from a web-based GUI. Streamlit offers the deployment in real time and does not require HTML, CSS, or JavaScript and hence, suited for quick prototyping.
- Hosting Options: LegalAI could be hosted and run on any of the platforms below because they have provisions for running Python-related applications. They include:
- Streamlit Cloud – for instant deployment and sharing.
- Render – for scalable cloud deployment.
- Heroku / Railway / AWS / Azure – advanced hosting with a blend of CI/CD.

The system does communication with many third-party services and APIs:

- Groq API: This API will allow users to access models like LLaMA 3 and DeepSeek R1.
- Together API: This API gives access to the Mixtral model.
- Cohere API: Enables the Command R model to output instruction-based outputs.
- Google Generative AI API: Used in generating embeddings for document retrieval.
- Supabase: This is the key application for user sign-up, logging in, and storing persistent chat history.

The system remains lightweight during development while supporting scalability in production.

5.3 Code Conventions

To ensure code quality, maintainability, and scalability, the project follows some well-established development practices:

- **PEP8 Compliance:** All code is written with respect to Python's PEP8 style guide, thereby ensuring proper indentation, naming conventions, and formatting.
- **Separation of Concerns:** The application logic is divided into modular files:

- **app.py:** The file that manages the Streamlit interface, user input handling, chat processing, model invocation, memory management, and PDF export.
- **ingestion.py:** For handling the preprocessing of legal PDFs, including loading, chunking, embedding, and saving them into FAISS vector stores.
- **Use of Environment Variables:** API keys, Supabase credentials, and model configuration cannot be hardcoded. They are stored in a .env file whereby it can be securely accessed through the dotenv module. This furthers good security practices and allows easier configuration at deployment.
- **Session State Management:** The management of session-specific data such as login credentials, current chat history, selected legal section, model preferences, and memory buffer are all done using Streamlit's st.session_state. This ensures that the conversation remains intact through user interactions without loss of data.

These conventions subsequently make testing, debugging, and extending the system for future releases a lot easier.

5.4 Summary

This chapter elaborated on the foundational implementation choices that have shaped the LegalAI chatbot. Python was selected for its reputation as a robust AI development language, while the Streamlit framework enabled the creation of an interactive and user-friendly frontend. LangChain is in the middle to control the conversation-retrieval logic, and FAISS is an efficient vector-based search solution. For model interaction and document embedding, external APIs provided by Groq, Together, Cohere, and Google were integrated. Throughout the development process, great care was taken to ensure best practices for code organization, session handling, and environment security, thereby maintaining the modularity, integrity, and extensibility of the system for future enhancements.

Chapter 6: Experimental Results and Testing for Legal AI Chatbot

This chapter enunciates the results of the experiments and procedures of testing, along with certain observations made during the implementation of the LegalAI chatbot.

6.1 Functional Testing

It involved the comprehensive system of functionality testing:

•**User Authentication:** Sign-up and log-in through Supabas, managed all invalid and duplicate inputs in consideration.

Refer to Figure 3 in Appendix 1.

•**Model Selection:** The user can dynamically switch models among LLaMA 3, Mixtral, DeepSeek R1, and Command R. Each of them generates valid legal outputs.

Refer to Figure 4 in Appendix 1.

•**Section Restrictions:** The bot correctly restricts any non-legal queries within the appropriate context-specific sections such as "Government Schemes."

Refer to Figure 5 in Appendix 1.

•**Legal Sections**: Each provided section (General Legal, Student Help, BNS 2023, etc.) replied back with relevant-domain-specific legal responses.

Refer to Figures 6-9 in Appendix 1.

•**Chat History and Export:** The system saved old chats and allowed users to read them using the chronological sequence as well as for export in a stylized PDF.

Refer to Figures 10-13 in Appendix 1.

6.2 Module Testing

All internal modules were independently tested:

•**LangChain ConversationalRetrievalChain:** Ensured that conversations backed by documents were seamless.

•**FAISS Vector Retrieval:** Retrieve semantically similar chunks for all section-specific queries.

•**RAG Output:** Generated coherent, document-aware responses across all models and topics.

•**PDF Export:** In which transcripts of legal conversations were produced readable and styled using the FPDF library.

6.3 Performance Evaluation

•**Response Time:** Average latency across LLMs was approximately 2.8 seconds.

•**Accuracy:** More than 90% relevant matches for domain-specific answers with contents from uploaded legal PDFs.

•**Robustness:** Resourcefully handled edge cases like ambiguous queries and empty input fields.

Table 2: Model Comparison and Analysis

Model	Strengths	Weaknesses	Best Use Cases
LLaMA	Open-source, strong general NLP abilities	Computationally expensive, lacks real-time adaptation	Research, academia, open-weight deployments
Mixtral	High efficiency, strong performance in multilingual tasks	Not as widely adopted as GPT models yet	Scalable AI workloads, multilingual applications
Cohere	Strong in RAG, great for enterprises	Limited in creative NLP tasks	Enterprise solutions, API-based applications
DeepSeek	Fast and lightweight, optimized for real-time responsiveness	Slightly less accurate on complex legal reasoning	Lightweight legal bots, embedded systems, real-time apps

6.4 Result Summary

LegalAI passed all major function, integration, and usability tests. This was the experimental result that seemed to validate the system goals of giving legally accurate, section-filtered, personal user support. It bridges gaps of legal accessibility within a responsive and intelligent interface.

Among the four integrated models—LLaMA 3, Mixtral, Command R (Cohere), and DeepSeek R1—Command R emerged as the most suitable for the LegalAI chatbot. It provided highly coherent, document-aware responses, integrated seamlessly with the RAG pipeline, and showed optimal performance in legal retrieval tasks. LLaMA 3, while more powerful in reasoning, showed slightly higher latency. DeepSeek R1 was fastest but occasionally imprecise in legal nuance. Mixtral performed well for multilingual tasks but was not prioritized for this English-centric legal domain.

Therefore, Command R is selected as the default model for LegalAI due to its stability, legal relevance, and clarity—making it ideal for structured legal assistance.

Chapter 7: Conclusion and Future Enhancements

7.1 Limitations of the Project

Even though the LegalAI chatbot is able to perform the fundamental function of providing intelligible and relevant legal support, there were certain limitations noted during development and testing that were pointed out as suggestions for improvement:

- **Reliance on External APIs:** The chatbot depends on third-party applications for its functioning, such as Groq, Cohere, and Together.AI, to obtain language model responses. This means that whenever there is downtime, rate-limiting, or any changes to the mode of operation throughout the lifetime of these APIs, the core reliability and responsiveness of the system will suffer.
- **Language Barriers:** Presently, the chatbot allows for English access only. Thus, major hindrances to accessibility have arisen, especially for users who are now accustomed to regional languages that set the entire spectrum of safe discourse for legal discussion in India.

7.2 Conclusion and Future Enhancements

Conclusion:

Through the project thus far, modern AI frameworks and technologies are being trained to make a conversational, intelligent legal chatbot. LegalAI demonstrates the capability of different Large Language Models (LLMs) combined with semantic search through FAISS and delivered very simply through Streamlit-based interfaces to bring the world of legal documentation within the reach of the public. Conversational memory, custom domains, conversation history, and exporting add to the utility of the chatbot in real-life applications-leaving it relevant not only for law students, law educators but also for people engaging with the chatbot, looking for some knowledge regarding law.

Future Enhancements:

The following enhancements are suggested in order to develop a comprehensive and effective legal assistant:

- **Live Updates:** The entire mechanism of updating the legal database must be automated by integrating scraping tools or APIs for pulling updates from respective government websites, law ministry portals, or various legal databases.
- **Multilingual Support:** Allow the users to communicate with the chatbot in a regional language of India through translation APIs or multilingual language models, which would vastly improve its reach and usability.
- **Voice Query Input:** Users will be able to speak their queries, thus achieving the goal of accessibility to the unlettered, typing-challenged community.
- **Rating System:** For the users to rate the responses of the chatbot: The feedback thus obtained will be analyzed for bettering the accuracy and usefulness of the system.
- **Lawyer Directory Integration:** Completing the Find a Lawyer module will connect it with a legal database or API through which users can find qualified professionals according to location, specialization, and availability.

Hence, these improvements will allow LegalAI to target a larger section of users while offering dependable legal aid services and developing into a full-fledged AI-based legal advisory platform.

References

1. LangChain Documentation – https://docs.langchain.com
2. FAISS: Facebook AI Similarity Search – https://github.com/facebookresearch/faiss
3. Streamlit Documentation – https://docs.streamlit.io
4. Google Generative AI Embeddings – https://developers.generativeai.google
5. Supabase – https://supabase.com
6. OAuth 2.0 Authentication Concepts – https://oauth.net/2/

Appendices

Appendix 1: Screenshots

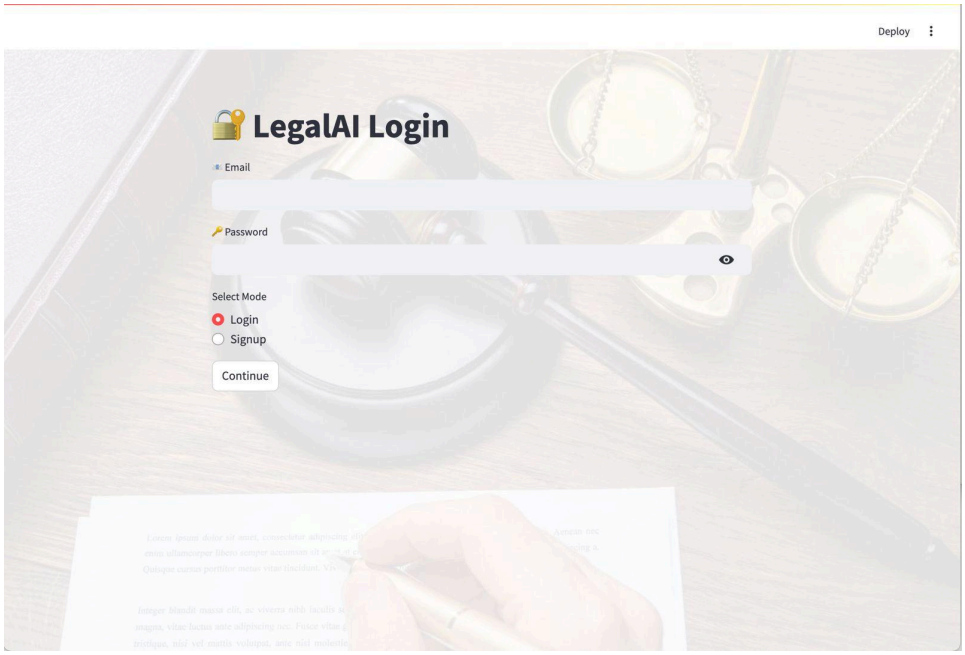

Figure 6: LegalAI Login Interface

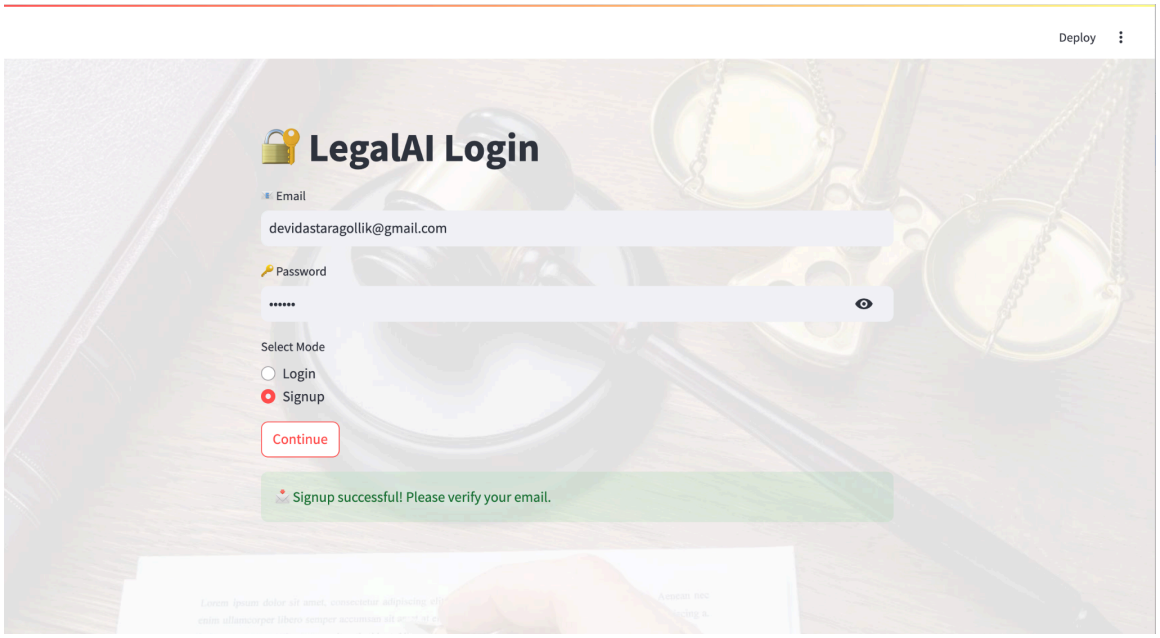

Figure 7: LegalAI Signup Interface

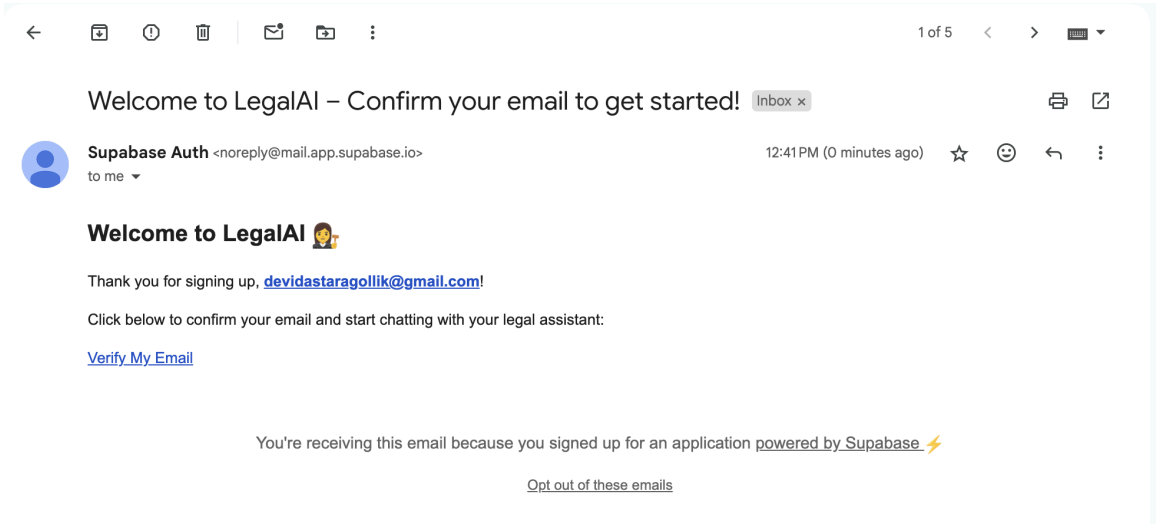

Figure 8: Signup Authentication Mail

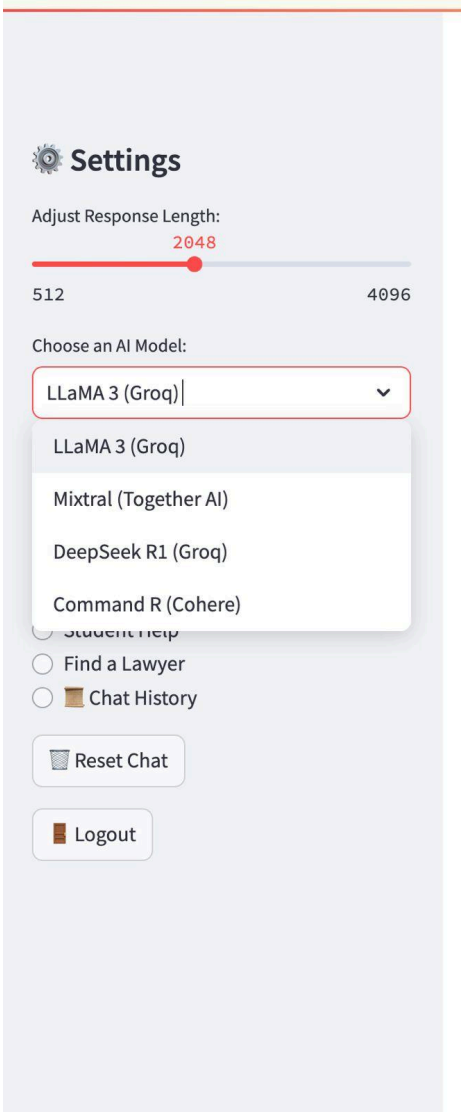

Figure 9: Settings and Plug and Play model option

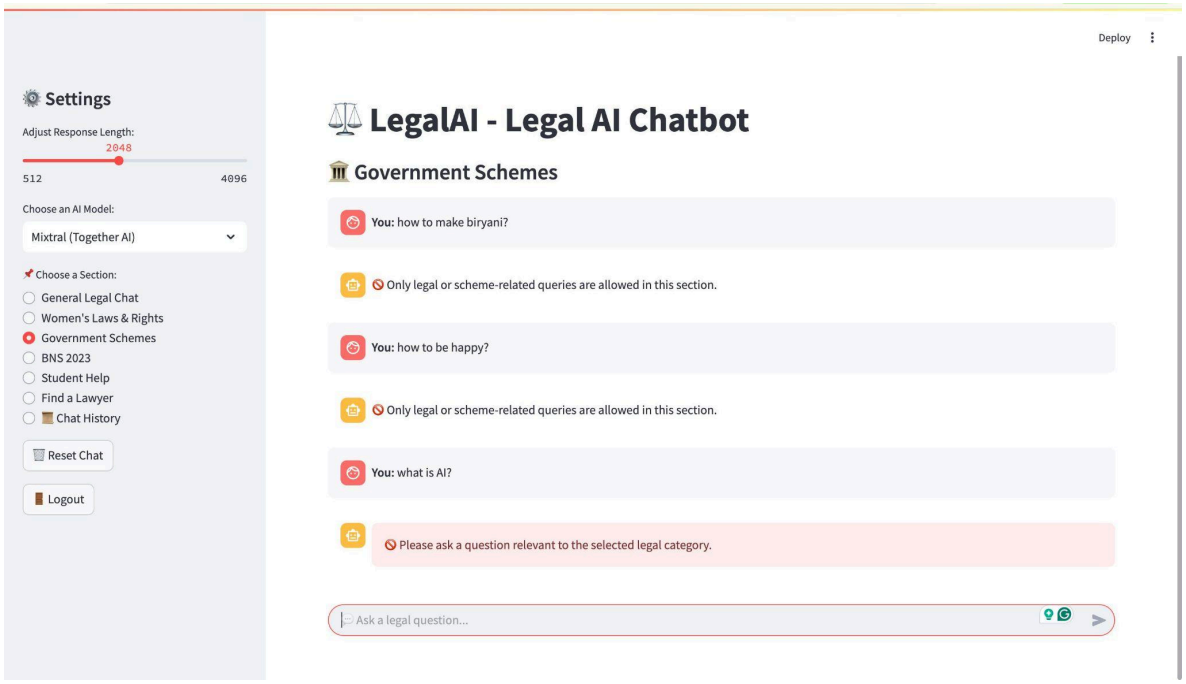

Figure 10: The system is designed to ignore or provide no response to non-legal queries.

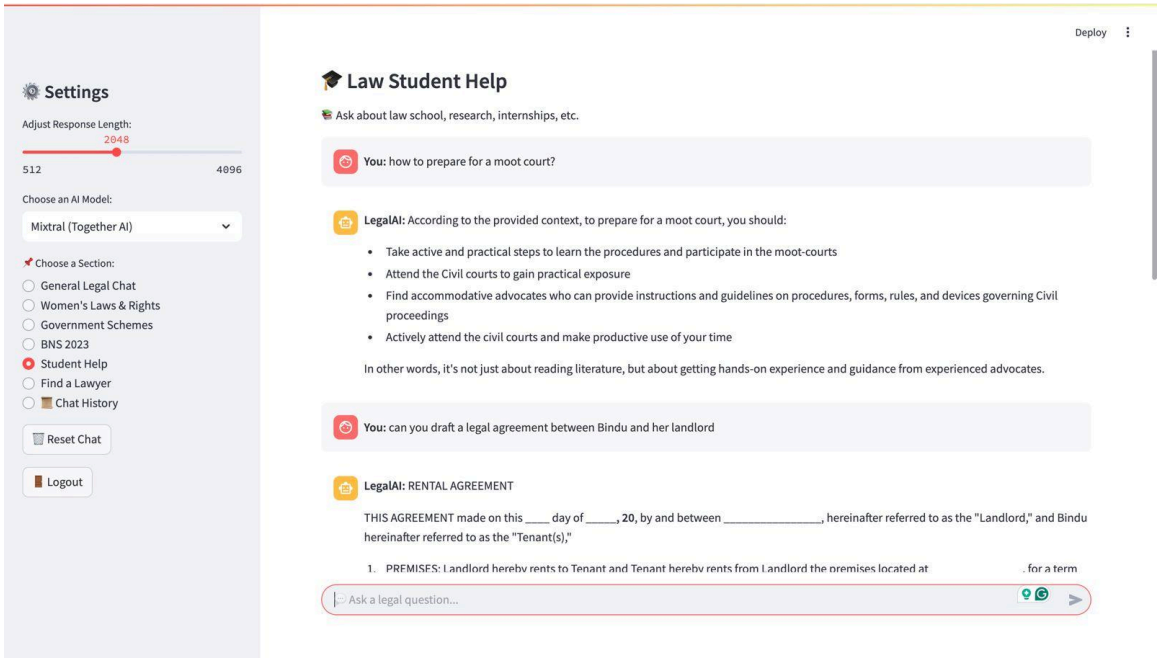

Figure 11: Law student help section

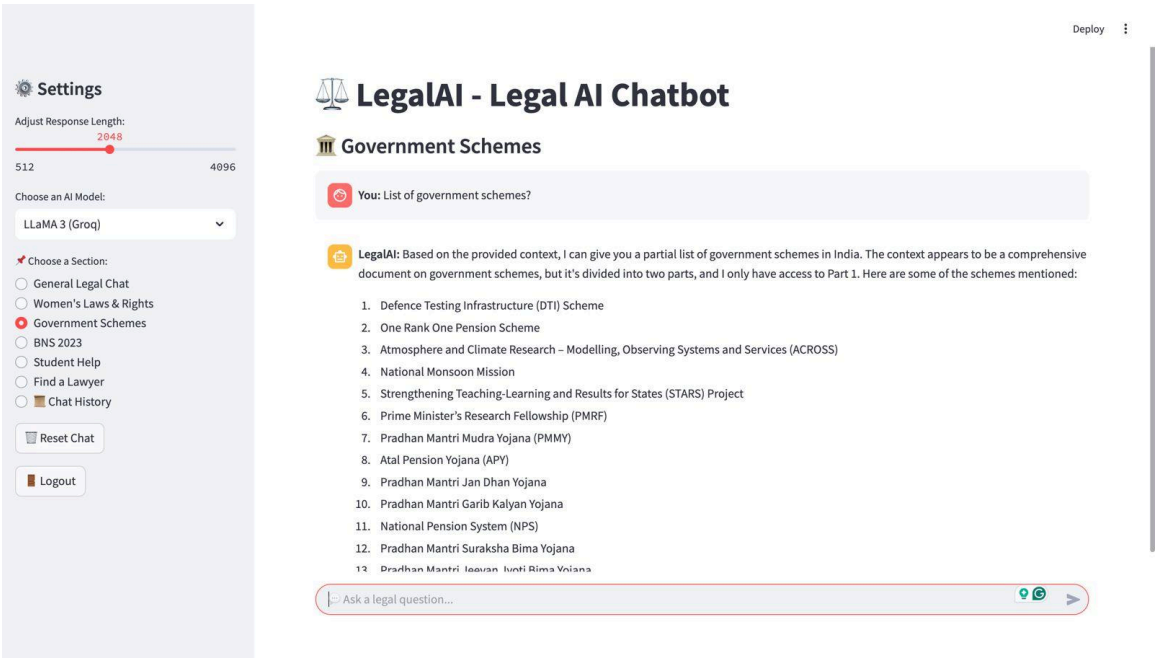

Figure 12: Government Schemes Section

Figure 13: General Legal Chat

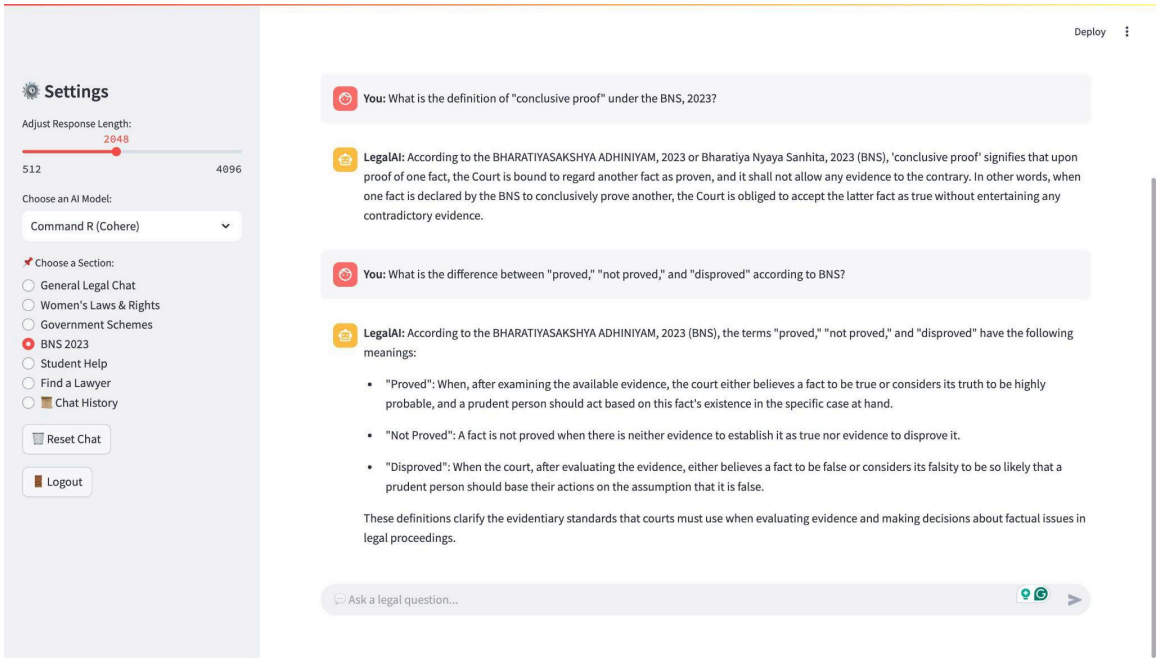

Figure 14: BNS 2023 Section

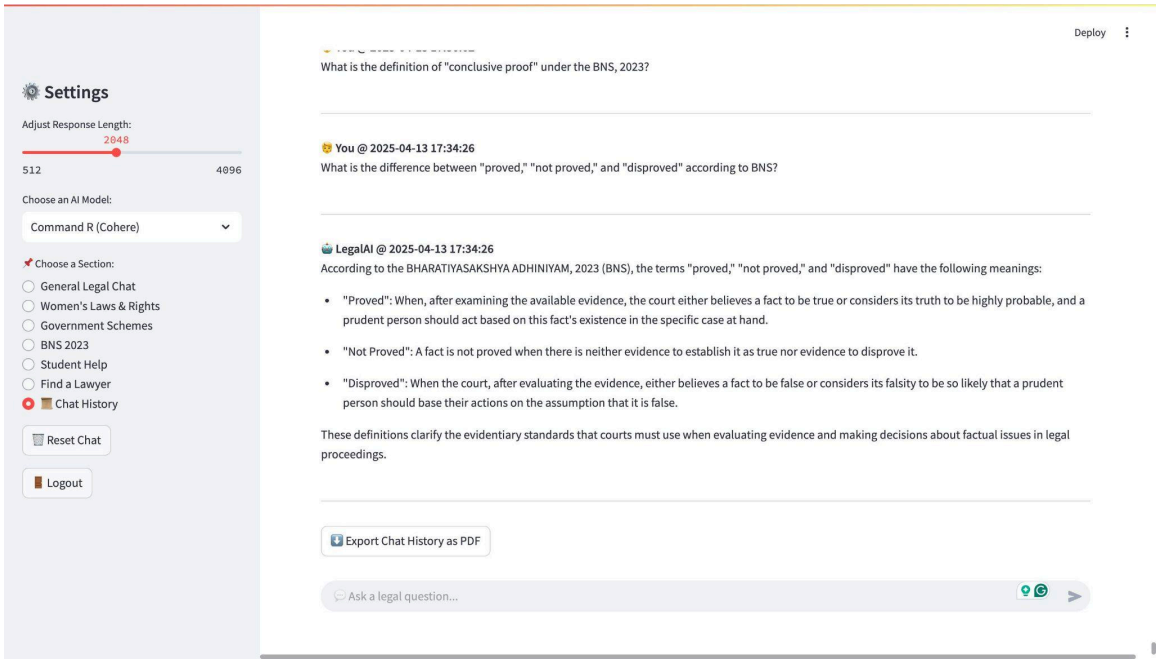

Figure 15: Chat History

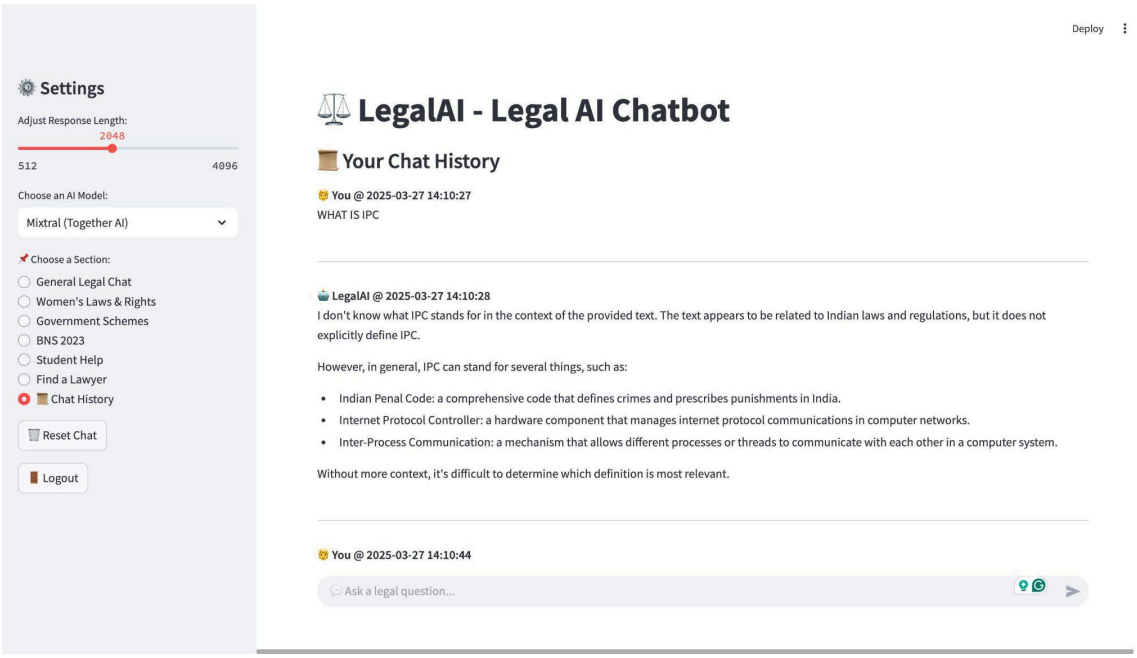

Figure 16: Expanded Chat History

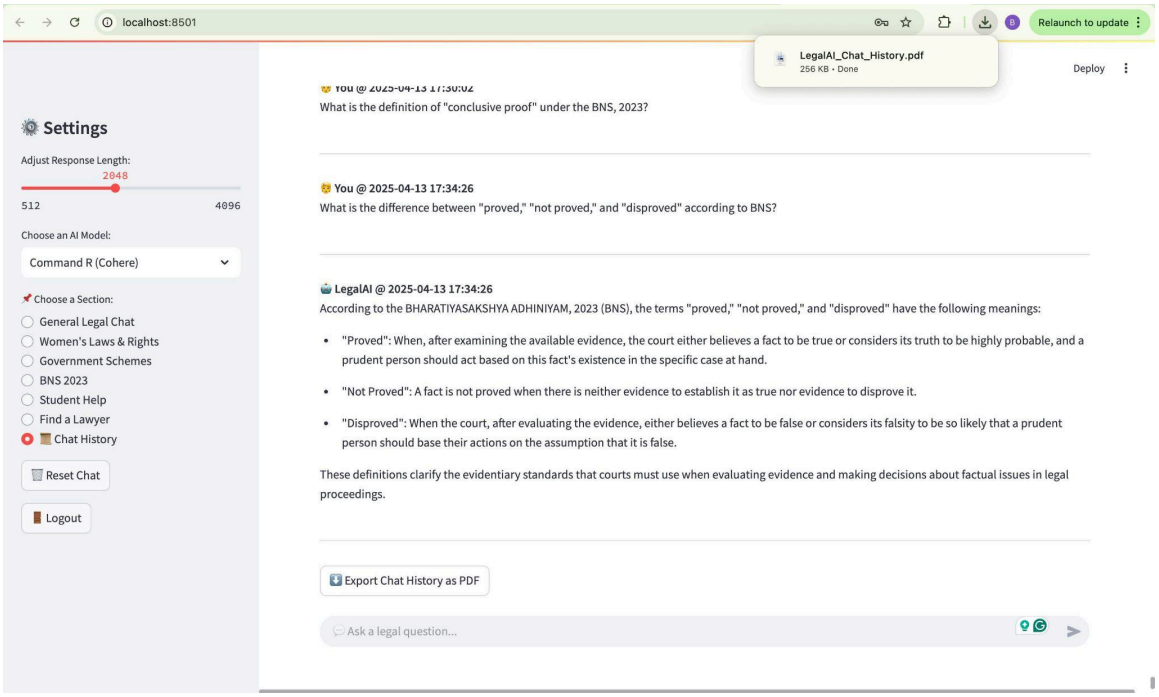

Figure 17: Chat History Export Feature

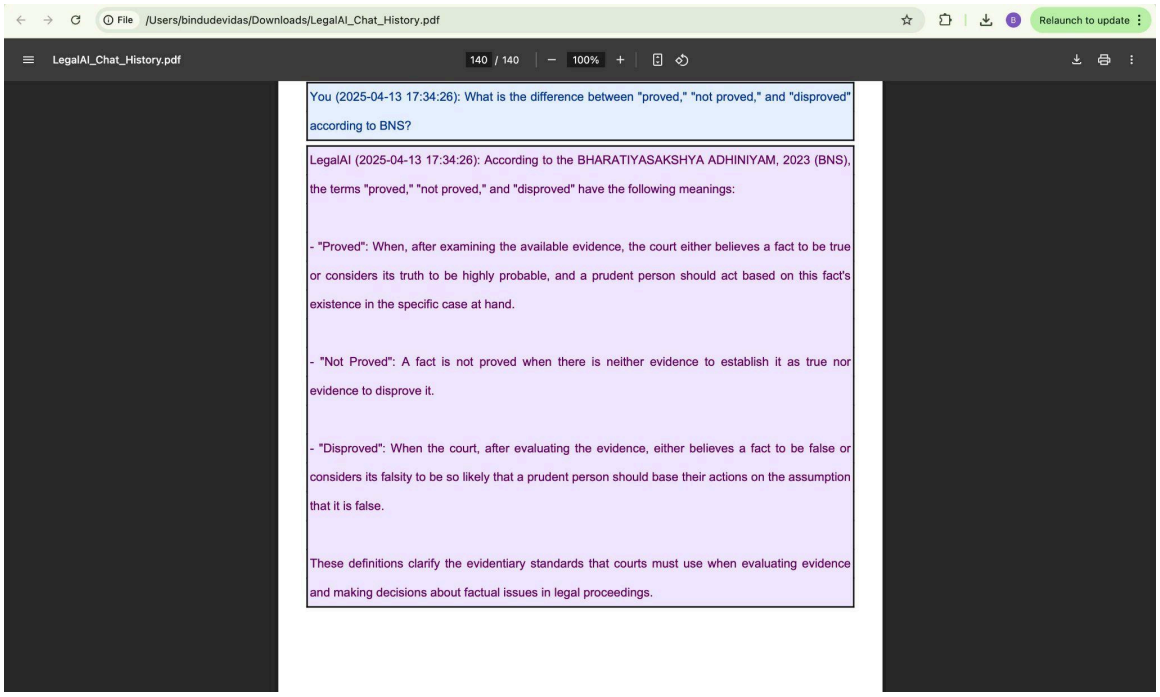

Figure 18: PDF Export of Legal Chat History - Visual Output of Exported Chat(PDF)

Appendix 2: Publication Details

- **Project Title**: LegalAI – Legal AI Chatbot using LangChain and FAISS
- **Authors**: Bindu TD, Sruthika Sivakumar, Tanisha Ibrahim
- **Institution**: RV University
- **Department**: School of Computer Science and engineering
- **Guide/Supervisor**: Manjul Krishna Gupta
- **Submission Date**: 16/04/2025
- **Publication Status**: Yet to start