

A Project Report On

AI ATTORNEY

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

by

PRATHAPANI SATWIKA (20BCD7160)

MEKALA POORNIMAI (20BCD7167)



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

MAY 2024

DECLARATION

I here by declare that the thesis entitled “AI ATTORNEY” submitted by me, for the award of the degree of Specify the name of the degree VIT is a record of bonafide work carried out by me under the supervision of Dr. Chirra Venkata Ramireddy.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Amaravati

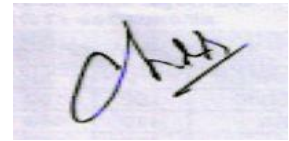
Date:22/05/2024

P. Satwika,

Signature of the Candidate

CERTIFICATE

This is to certify that the Senior Design Project titled “**AI ATTORNEY**” that is being submitted by **Prathapani Satwika(20BCD7160)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

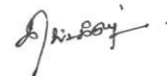


Dr. Chirra Venkata Ramireddy
Guide

The thesis is satisfactory / unsatisfactory



Dr. Srinivasa Reddy
Internal Examiner1



Dr. Selva Kumar S
Internal Examiner2

Approved by

Dr. G. Muneeswari

HoD, DataScience and Engineering
School of Computer Science and Engineering

ABSTRACT

Natural language processing (NLP) and Machine Learning components are integrated into Streamlit, an AI-powered document chat system. Users can interact with the system through interactive conversations where they can discuss uploaded PDF documents or query a language model that has already been trained. In order to enable intelligent responses and document understanding, the application makes use of OpenAI's language models for text embeddings. When users upload PDFs, the system reads the files and extracts the text. After that, it uses OpenAI's API to create embeddings and breaks the text up into digestible chunks. Because these embeddings are kept in a vector store, retrieving them quickly during user interactions is made possible. An OpenAI chat model handles the conversational part, and a conversation buffer memory keeps track of previous conversations for context-aware replies. Streamlit was used in the design of the user interface to provide a seamless experience. The application has different sections for home, about, news, document search, code explanation, and interactive chat. The concept of AI Attorney presents a novel approach to legal document comprehension, showcasing the potential of sophisticated language models in legal contexts. The system seeks to improve accessibility and comprehension of legal content by providing users with an intelligent, document-centric chat interface.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. Chirra Venkata Ramireddy, Assistant Professor Sr Grade I, SCOPE, VIT-AP, for his/her constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Computer Science.

I would like to express my gratitude to Dr. G. Viswanathan, Dr. Sekar Viswanathan, Dr. S. V. Kota Reddy and Dr. Madhusudhana Rao N, SCOPE, Saroj Panigrahy, SCOPE for providing with an environment to work in and for his inspiration during the tenure of the course. In jubilant mood I express ingeniously my whole-hearted thanks to Dr. G. Muneeswari, HOD / Department of DataScience and Engineering all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Amaravati

Date: 22/05/2024

Name of the student

Prathapani Satwika

CONTENTS

CONTENTS.....	iii
LIST OF FIGURES	iv
LIST OF TABLES	iv
LIST OF ACRONYMS	iv
CHAPTER 1	
INTRODUCTION	
1.1 OBJECTIVES	2
1.2 BACKGROUND AND LITERATURE SURVEY.....	4
1.3 ORGANIZATION OF THE REPORT.....	6
CHAPTER 2	
PROPOSED SYSTEM AND METHODOLOGY	
2.1 PROPOSED SYSTEM	7
2.2 WORKING METHODOLOGY	9
2.3 SOFTWARE DETAILS.....	10
CHAPTER 3	
RESULTS AND DISCUSSIONS.....	20
CHAPTER 4	
CONCLUSION AND FUTURE WORK.....	22
CHAPTER 5	
APPENDIX.....	23
REFERENCES.....	44

LIST OF FIGURES

1. LEGAL ANALYSIS OF INDIA	1
2. COMPARISON OF KEYWORD EXTRACTION MODELS	4
3. SYSTEM BLOCK DIAGRAM	7
4. GET PDF TEXT FLOW DIAGRAM.....	11
5. WEB DIAGRAM.....	12
6. HOME PAGE.....	13
7. ABOUT PAGE.....	14
8. CHAT WITH ME PAGE.....	15
9. CHAT WITH DOCUMENT PAGE.....	16
10. SEARCH PAGE.....	17
11. UNDERSTAND ME PAGE.....	18
12. TODAYS NEW PAGE.....	19
13. RESULT FROM CHAT WITH ME PAGE.....	20
14. RESULT FROM CHAT WITH DOCUMENT.....	21

LIST OF TABLES

1. KEYWORDS EXTRACTION MODELS COMPARISION TABLE.....	5
--	---

LIST OF ACRONYMS

NLP	NATURAL LANGUAGE PROCESSING
AI	ARTIFICIAL INTELLIGENCE

CHAPTER 1

INTRODUCTION

The "AI Attorney" project is an innovative investigation into the fields of Natural Language Processing (NLP) and Artificial Intelligence (AI), with the goal of transforming the comprehension and interaction of legal documents. This creative project combines the power of multiple frameworks and technologies to produce a clever and intuitive AI-powered legal assistant system. The project's main component is the Streamlit framework, a potent instrument for quickly and easily building web applications. The user interface exhibits a high degree of versatility and breadth, as it allows users to easily navigate through various sections such as home, about, code explanation, news, document search, and an engaging "Chat with Me!!" feature.

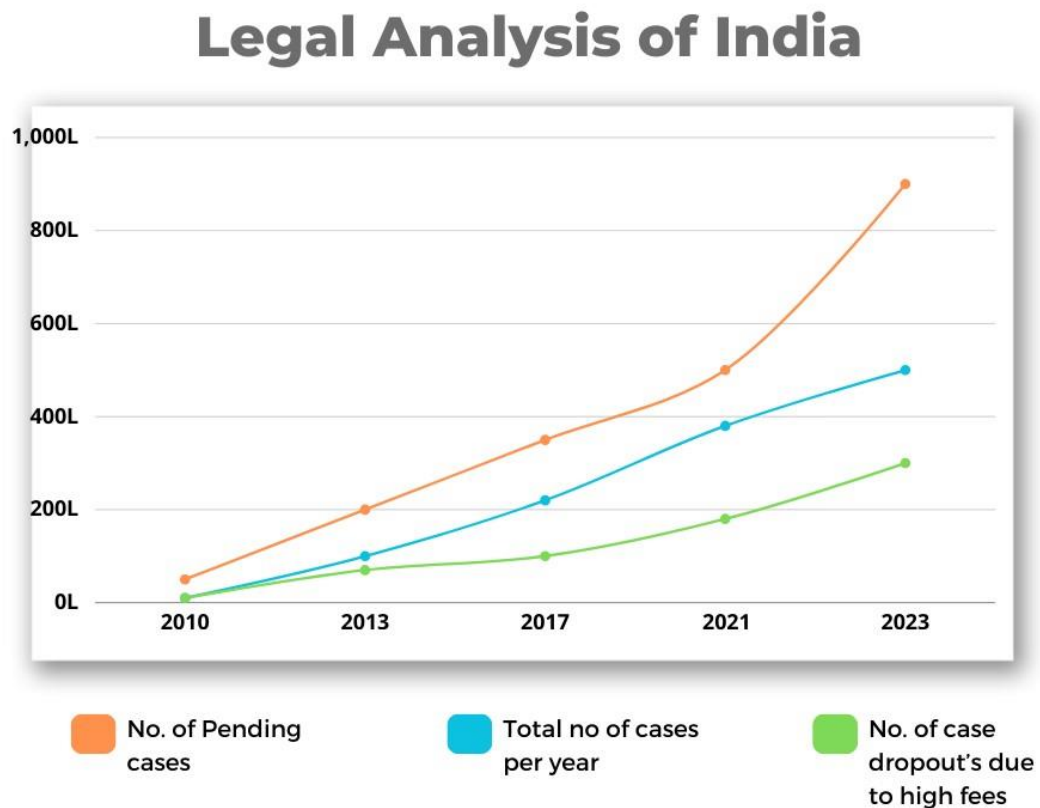


Figure 1: Legal Analysis of India

The AI Attorney's main objective is to improve legal document accessibility and comprehension through interactive conversations. Either by uploading PDF documents for contextualized conversations or by querying a language model that has already been trained, users can participate in dynamic discussions. The system makes use of advanced text embeddings and OpenAI's language models to comprehend legal language and intelligently respond to user inquiries.

One notable feature is the document-centric chat interface, which enables users to converse meaningfully and gain a deeper understanding of the content while also searching through a large corpus of legal documents. By recognizing the potential of AI in legal applications, the project demonstrates a forward-thinking approach and paves the way for a more intelligent and effective legal future.

In addition, the system uses intelligent document processing, chunking and extracting text from uploaded PDFs, utilizing OpenAI's API to create embeddings, and storing them in a vector store for quick retrieval during talks. By combining AI, NLP, and user-centric design, the "AI Attorney" is a step toward changing the legal landscape rather than merely being a tool. This introduction lays the groundwork for an ambitious project that aims to redefine legal information access and comprehension in the digital age, while also showcasing technical prowess.

1.1 Objectives

All of these goals work toward developing an AI attorney system that can comprehend and evaluate legal documents, as well as have meaningful conversations with users, offering insightful legal advice and staying current on legal developments.

- Conversational Legal Assistance:
 - Create a conversational user interface to give users information and legal support.
 - Give users the ability to communicate with the AI Attorney by using natural language inquiries.

- Document Understanding and Analysis:
 - Establish a system for processing documents that can extract valuable information from legal documents, particularly those in PDF format.
 - Create algorithms that can comprehend and evaluate legal text to find pertinent information.
- Real-time Interaction:
 - Provide users requesting prompt legal advice with the ability to communicate in real-time with the AI Attorney.
 - Develop a system that is responsive to user inquiries and responds with prompt and accurate information.
- Multi-Modal Document Support:
 - Extend the system's capabilities to support various types of legal documents beyond text, including images and scanned documents.
 - Enhance document understanding to handle diverse legal content.
- User-Friendly Experience:
 - Create an interface that is easy to use and intuitive to enable smooth communication between users and the AI Attorney.
 - Give top priority to usability and accessibility for people with different levels of legal expertise.
- Integration of News and Updates:
 - Incorporate a feature that provides users with relevant legal news and updates.
 - Enhance the AI Attorney's knowledge base with the latest legal information to keep users informed.

1.2 Background and Literature Survey

Although AI technologies have shown they can speed up the work of lawyers by handling routine tasks and technical tasks, they are not yet capable of completely replacing legal specialists (Deltsova, 2020) [1]. This idea offers a solid and condensed basis for assessing developments in the application of AI in the legal field. Legal practitioners can use this perspective to evaluate the impact of AI on legal practice and the achievement of desired results, while academics studying the role of AI in legal reasoning can use it to track progress (Eliot, 2020) [2]. It is argued that investigating AI within the context of law will play a crucial role in encouraging the creation of AI systems that support human welfare, underscoring the increased importance of this intersection (Verheij, 2020) [3]. Simultaneously, it is acknowledged that artificial intelligence has the potential to serve as a crucial "assistant" to legal professionals, providing a way to improve the effectiveness and caliber of legal services provided (Vasiliev & Pechatnova, 2020) [4].

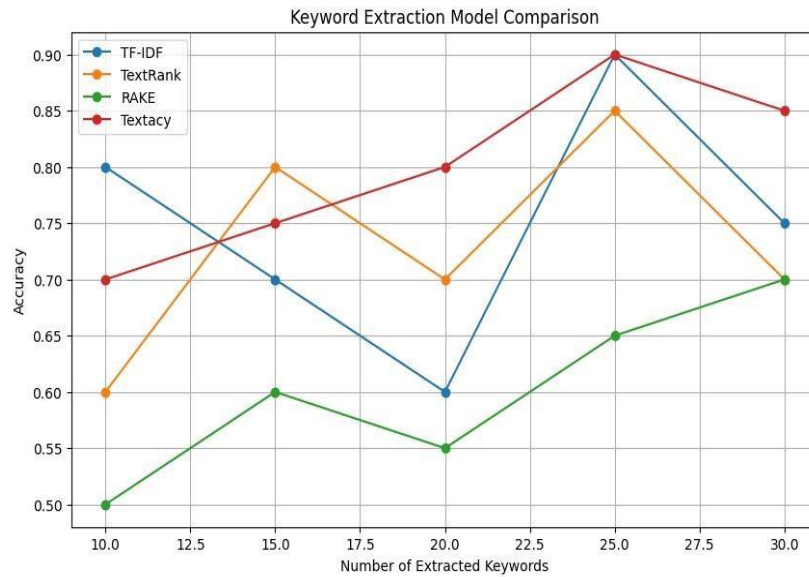


Figure 2: Comparison of Keyword Extration Models

The literature has proposed a number of novel keyword extraction methods, demonstrating developments in this area. In order to improve news retrospective event detection accuracy and efficiency, Li et al. introduced a Chinese news document keyword extraction method based on tf/idf with multiple strategies [5]. This method outperformed baseline methods. Wang and Ning proposed a TF-IDF method that combines context and semantic classification to address problems with keyword extraction [6]. When compared to conventional TF-IDF and TextRank methods, improved TextRank algorithms for keyword extraction showed superior generality and accuracy [7].

Additionally, a newer version of TextRank was created for extracting keywords from text streams. It offers notable speed improvements and is useful for research projects [8]. The efficiency of keyword extraction was found to be improved by the combination of Named Entity Recognition (NER) and RAKE [9]. Additionally, domain-specific adaptations like RAKEB showed improved keyword extraction performance in the case of Bengali language [10]. The traditional TextRank algorithm was also improved by a cutting-edge method that combined keyword extraction with text network analysis.

Table 1: Keyword extraction models comparison table

Model	Method	Dependency	Algorithm
1	POS Tagging + Lemmatization	NLTK	Custom Rules
2	TF-IDF	NLTK	TF-IDF
3	TextRank + Co-occurrence Graph	NLTK, NetworkX	TextRank
4	YAKE	YAKE	Statistical
5	RAKE	RAKE	Statistical
6	TextRank (Textacy)	Spacy	TextRank

The table presented compares various keyword extraction models, highlighting their methods, dependencies, and algorithms. There are six models listed:

1. Model 1 uses a combination of Part-of-Speech (POS) tagging and lemmatization. It relies on the Natural Language Toolkit (NLTK) and operates based on custom rules to extract keywords.
2. Model 2 implements the Term Frequency-Inverse Document Frequency (TF-IDF) method, also depending on NLTK. The TF-IDF algorithm identifies keywords by evaluating their importance across documents.
3. Model 3 employs TextRank in conjunction with a co-occurrence graph. This model depends on both NLTK and NetworkX, utilizing the TextRank algorithm for keyword extraction.
4. Model 4 is based on the YAKE method, which is a statistical approach for keyword extraction. It depends solely on the YAKE library.
5. Model 5 uses the RAKE (Rapid Automatic Keyword Extraction) method, another statistical approach, relying on the RAKE library.
6. Model 6 also uses the TextRank algorithm but is implemented through the Spacy library.

This comparative table provides a concise overview of the different approaches to keyword extraction, illustrating the diversity in methods, dependencies, and underlying algorithms.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 Proposed System and Methodology.
- Chapter 3 Results and Discussion.
- Chapter 4 Conclusion and Future work.
- Chapter 5 Code repository.
- Chapter 6 References.

CHAPTER 2

PROPOSED SYSTEM AND METHODOLOGY

This Chapter describes the System Architecture, working methodology and software details.

2.1 Proposed System

The following block diagram (figure 3) shows the system architecture of this project.

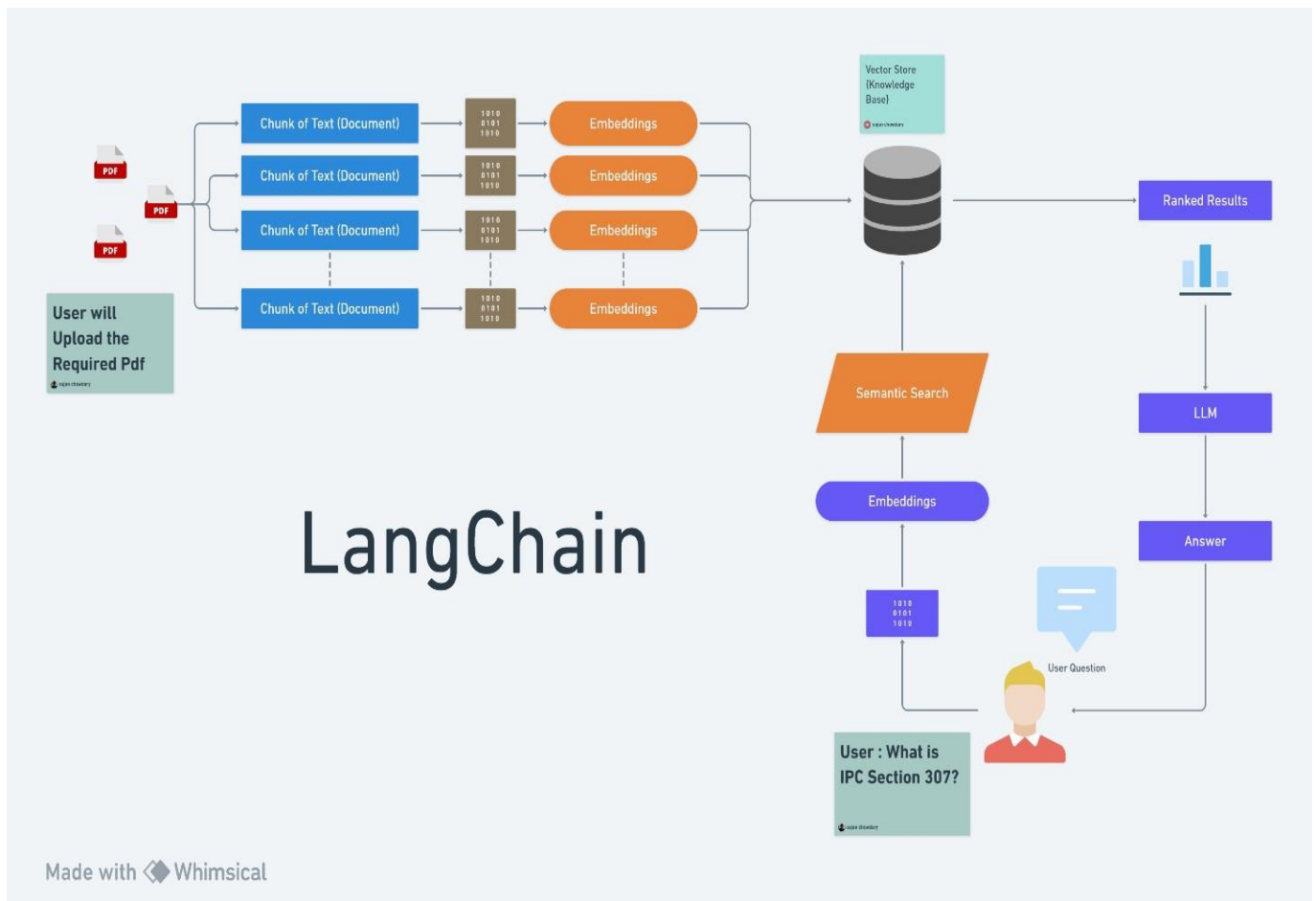


Figure 3 System Block Diagram

PROPOSED SYSTEM

An open-source framework called LangChain assists programmers in using large language models (LLMs) to create applications. It offers end-to-end chains for common applications, standard interfaces for chains, and integrations with other tools.

- Among LangChain's use cases are: Document analysis and summarization
- Chatbots
- Code analysis

Because LangChain is agentic and data-aware, it can connect to different data sources to provide individualized experiences. Agents can be used to decide which tasks to complete and to manage the chain's flow. By offering external integrations, prompt management, and standardized interfaces, LangChain streamlines development.

In addition to its versatility, LangChain supports a modular approach, allowing developers to build and customize specific components according to their project needs. This flexibility is crucial for applications that require specialized handling, such as legal document processing or complex conversational AI. LangChain's robust ecosystem includes pre-built modules and templates, which can significantly reduce development time and enhance the scalability of applications.

The framework also emphasizes security and compliance, ensuring that sensitive data can be handled safely and in accordance with industry standards. By integrating seamlessly with various APIs and databases, LangChain enables the creation of sophisticated, data-driven AI applications that can evolve with changing requirements. This makes it an ideal choice for projects like AI Attorney, where accuracy, reliability, and adaptability are paramount.

2.2 Working Methodology

The AI Attorney project makes use of a sidebar navigation system and Streamlit-based user interface to provide users with a variety of features like code understanding, information exploration, AI-driven document interaction, search capabilities, and access to the most recent news. Processing PDF documents and allowing users to ask legal questions about the content is the main function of the project. Through the sidebar, users can upload multiple PDFs. This initiates a processing mechanism that uses OpenAI's GPT-based embeddings stored in an FAISS vector store, to vectorize the documents and extract text from them.

It also segments the documents into manageable chunks. A ChatGPT model integrated into a Conversational Retrieval Chain facilitates the conversational aspect, and chat history is stored in a buffer memory. With distinct pages for home, about, code explanation, news, search, and AI-based chatting with both general and document-specific questions, the user interface offers a seamless experience.

The application's visually appealing layout, which makes use of CSS styling to improve the sidebar's appearance, complements its user-centric design. The code makes use of different modules for different functionalities, such as vectorization, conversational AI interactions, and text extraction from PDFs. A responsive interface and obvious navigation options make it easier for users to move between the application's various sections. The approach captures the complexity of the project by combining general conversational AI and document-centric AI features into a unified and intuitive Streamlit application.

To ensure robust performance, the system employs efficient backend processes for handling large volumes of data and maintaining fast response times. The integration of FAISS for vector storage and retrieval enhances the speed and accuracy of document searches. Additionally, the use of buffer memory for chat history allows for context-aware responses, making interactions more meaningful and relevant.

The modular design also allows for easy updates and scalability, ensuring that new features and improvements can be incorporated without disrupting the user experience. Regular updates to the AI models and embeddings ensure that the system remains current with legal language and evolving

user needs. By leveraging a combination of advanced AI technologies and user-centric design principles, the AI Attorney project aims to provide a comprehensive and efficient tool for legal document analysis and interaction.

2.3 Software Details

The AI Attorney project leverages a comprehensive software stack to ensure efficient development, robust functionality, and seamless integration of advanced technologies. The software components encompass various frameworks and libraries tailored to specific tasks within the system.

- **Programming Language:**
 - Python is widely used as the primary programming language due to its extensive libraries, versatility, and support for machine learning and natural language processing.
- **Conversational AI Module:**
 - Streamlit: Streamlit is a Python web application development library that makes it easier to create an intuitive user interface for dynamic dialogues.
 - langchain.chat_models: The integration of the Langchain library enables conversational modeling and offers sophisticated language comprehension functionalities.
- **Document Processing Engine:**
 - PyPDF2: An essential Python library for reading and modifying PDF documents is one that is used to extract text from legal documents.
 - langchain.text_splitter: Text splitting using the langchain library, which enables effective management of substantial amounts of legal text.
 - OpenAIEmbeddings: Contextual information within legal texts can be captured and generated through integration with OpenAIEmbeddings.
 - FAISS: FAISS (Facebook AI Similarity Search) integration is used to store and retrieve vectors efficiently.
- **Knowledge Base Integration:**
 - Hugging Face Hub: Hugging Face Hub is used to stay up to date with legal information and access pre-trained language models.
 - External APIs: Using APIs, the system can be integrated with outside news and legal databases to expand its knowledge base.

- Web Application Development:
 - HTML and CSS: fundamental web technologies for UI design and organization.
 - Streamlit Option Menu: In the Streamlit Option Menu library has been integrated to improve user interaction features.
- Data Storage and Management:
 - Utilization of databases such as SQLite or MongoDB for efficient storage and retrieval of user data, preferences, and session information.
 - Integration of data management frameworks like Pandas for handling structured data and performing data analysis tasks.

Together, these software specifics create a seamless stack that enables the AI Attorney to provide a strong, perceptive, and easy-to-use legal assistance system. To guarantee the project's success and future scalability, a balance between cutting-edge technologies and well-established libraries was struck in the selection of these tools.

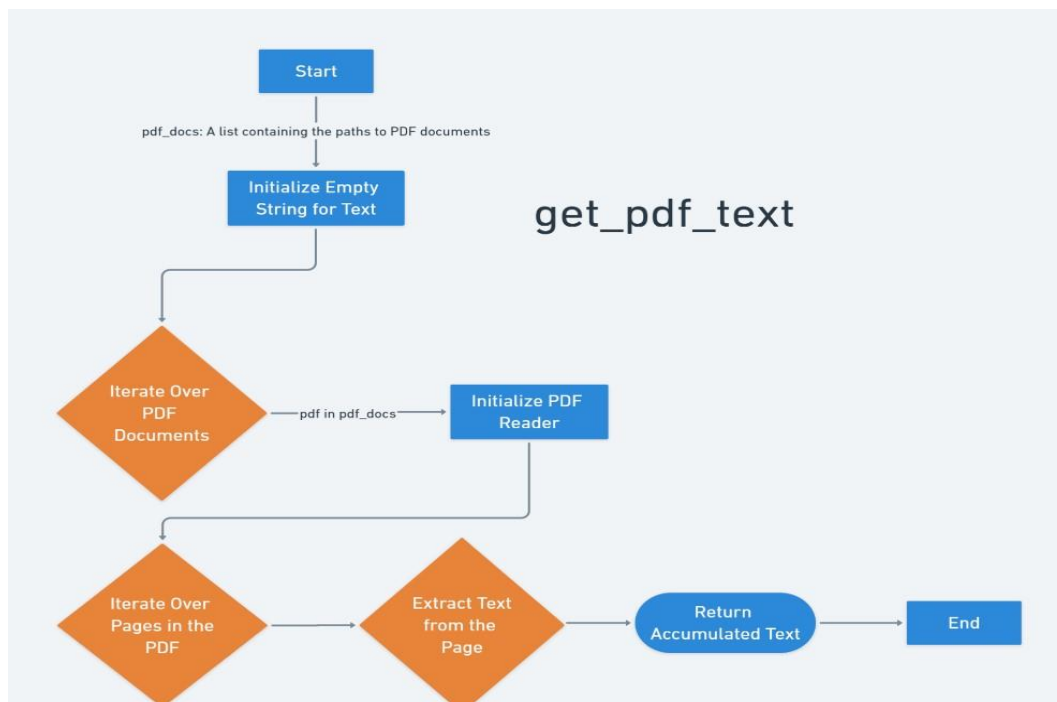


Figure 4: Get Pdf Text Flow Diagram

The flowchart Figure 4 illustrates the process of extracting text from PDF documents through a function called `'get_pdf_text'`. The procedure begins at the "Start" node and involves initializing an empty string to hold the accumulated text. This initial step prepares the function to store the text extracted from the PDFs.

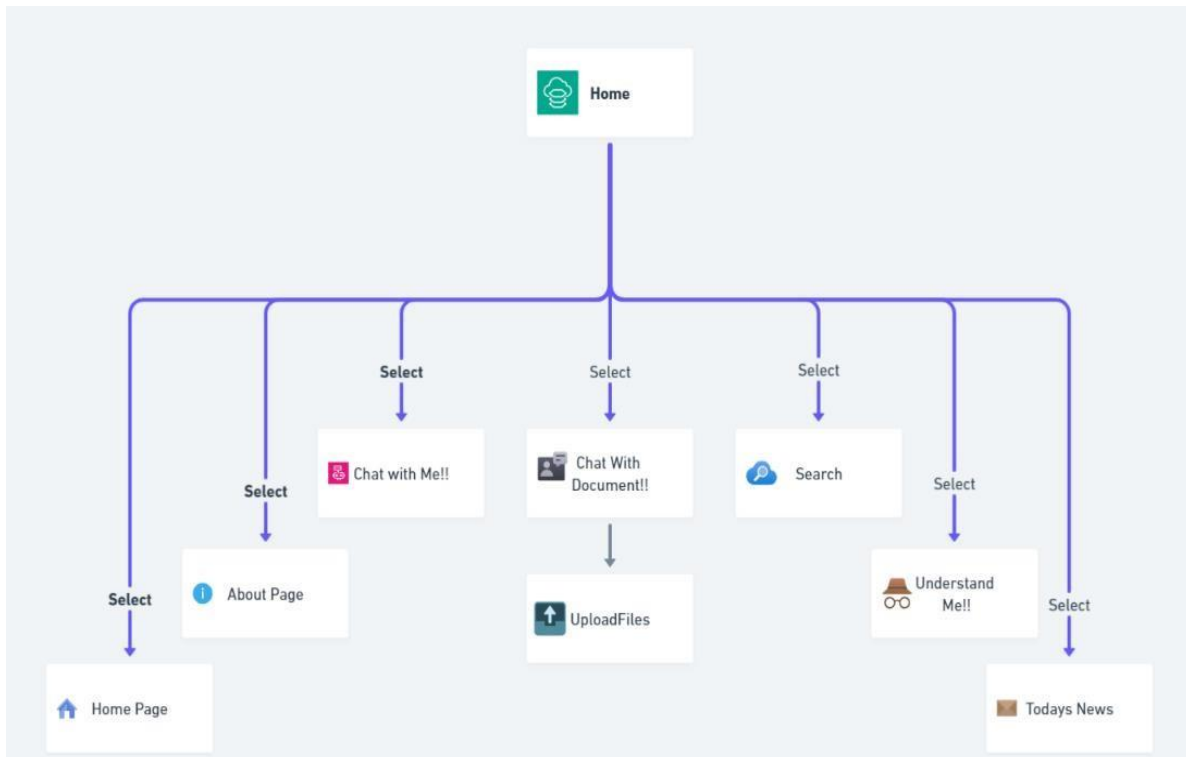


Figure 5: Web Diagram

The flowchart Figure 5 illustrates the navigation structure of a website or application, starting from the main "Home" page. From this central hub, users can access various sections tailored to different functionalities. One prominent feature is the "Chat with Me!!" option, which likely provides a chatbot interface for users to interact and seek assistance. Another key feature is "Chat With Document!!," where users can upload files via the "UploadFiles" option to facilitate document-based conversations. Additionally, there is a "Search" function that allows users to find specific information across the site. The "Understand Me!!" section suggests a customization or settings area to enhance user interaction by better understanding their preferences. For keeping users informed, the "Todays News" section offers the latest updates and news. There is also an "About Page" that provides background information about the website or application, ensuring users can learn more about its purpose and creators. Lastly, the flowchart includes direct navigation back to the "Home Page" from various sections, ensuring a user-friendly

experience by maintaining easy access to the starting point. This structure effectively organizes the site's features, ensuring users can navigate efficiently and find the information or services they need.

Home Page:

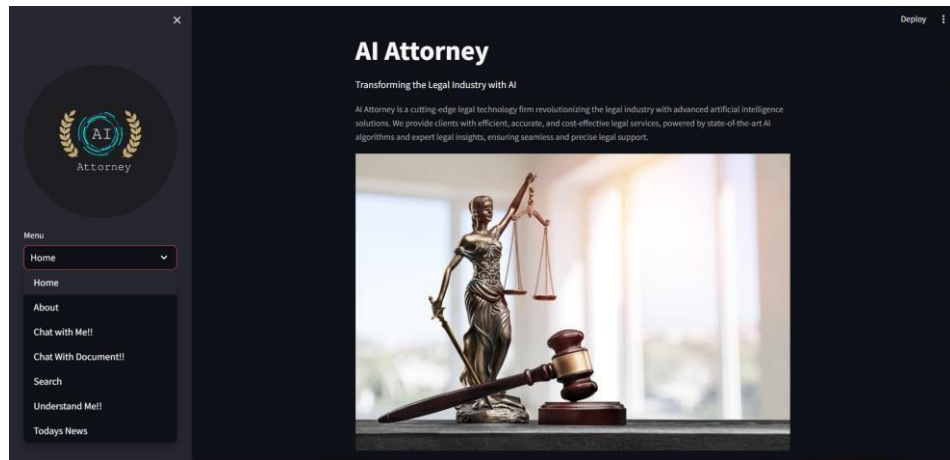


Figure 6: Home Page

The AI Attorney project's home page code makes use of Streamlit to present the ground-breaking AI-powered legal research system on a visually appealing platform. With an engaging mix of multimedia components, such as photos and interactive maps, the page clearly conveys the goals of the project, the inspiration behind them, and the makeup of the team. The code uses a two-column layout to highlight the system's cutting-edge Natural Language Processing (NLP) algorithms for accurate legal material retrieval, while also briefly showcasing the system's revolutionary capabilities. Additionally, the home page features a user-friendly navigation menu, allowing visitors to easily explore different sections like about, code explanation, news, document search, and interactive chat. The layout is designed to provide a seamless and intuitive user experience, ensuring that users can quickly access the information they need. By combining aesthetic appeal with functional design, the AI Attorney home page effectively introduces users to the innovative features and benefits of the platform.

About Page:



Figure 7: About Page

The "About page" code for AI Attorney on Streamlit presents the project in detail, showcasing its mission of empowering legal excellence through cutting-edge AI solutions by fusing educational text with eye-catching visuals like animations and images. It highlights the salient features of AI Attorney, such as its innovative approach, client-centric focus, efficiency in legal processes, unmatched accuracy in research, and integration of legal expertise and AI capabilities. In order to improve readability, the page also includes a thorough summary of the features of the product. It does this by using a two-column layout with images to illustrate how the system makes use of sophisticated Natural Language Processing (NLP) technology, scenario-based search functionality, PDF file analysis, keyword extraction knowledge, adaptive learning, and multilingual text translation. The contact form, which is animated with a fade-in effect, encourages user interaction and engagement and adds to the "About page" of the AI Attorney project's overall thorough and user-friendly design.

Chat With Me!! Page:

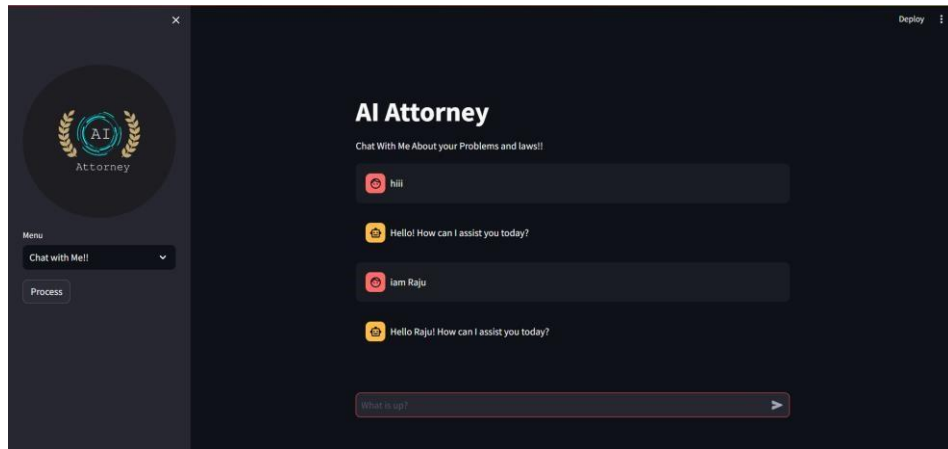


Figure 8: Chat With Me Page

Using Streamlit and OpenAI's GPT-3.5-turbo model, the "Chat With Me!!" code for AI Attorney generates an interactive chat interface where users can have dynamic conversations about legal issues and laws. This real-time integration of user inputs with AI-generated responses makes the platform easy to use and intuitive for legal questions and assistance within the AI Attorney project. The chat interface is designed to understand and respond to natural language queries, making it accessible to users without a legal background. By maintaining a conversational context, the system can provide more accurate and relevant information, ensuring users receive comprehensive answers to their legal questions. The feature also includes a conversation history, allowing users to refer back to previous interactions. This innovative approach not only enhances user engagement but also demonstrates the potential of AI in providing accessible legal support.

Chat With Document!! Page:

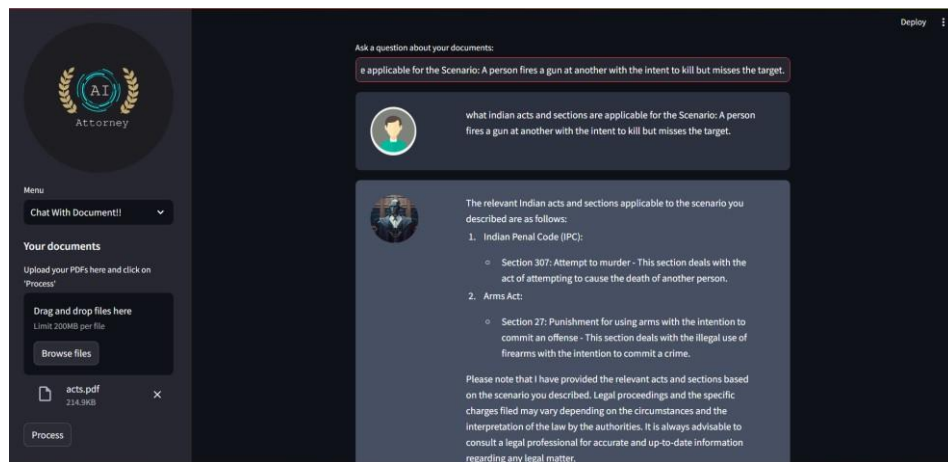


Figure 9: Chat With Document Page

Using key technologies like OpenAI's GPT-3.5-turbo model, PyPDF2 for PDF processing, and LangChain's language and conversation processing modules, the "Chat With Document!!" code in the AI Attorney project uses Streamlit and LangChain to create an interactive platform that will improve the efficiency and accuracy of legal document understanding and interaction within the AI Attorney framework. Users will be able to upload legal documents in PDF format, ask questions, and receive dynamic AI-generated responses. The system processes and analyzes the text within these documents, breaking them down into manageable chunks to ensure comprehensive and detailed responses. By leveraging advanced embeddings and vector storage, the platform can quickly retrieve relevant sections of the document during the conversation. This functionality not only streamlines legal research but also makes it accessible to individuals without extensive legal knowledge. Furthermore, the interface is designed to be user-friendly, allowing for seamless document uploads and intuitive querying, ultimately enhancing the overall user experience. This powerful combination of AI and document processing technologies represents a significant step forward in legal technology, making legal information more accessible and easier to navigate.

Search Page:

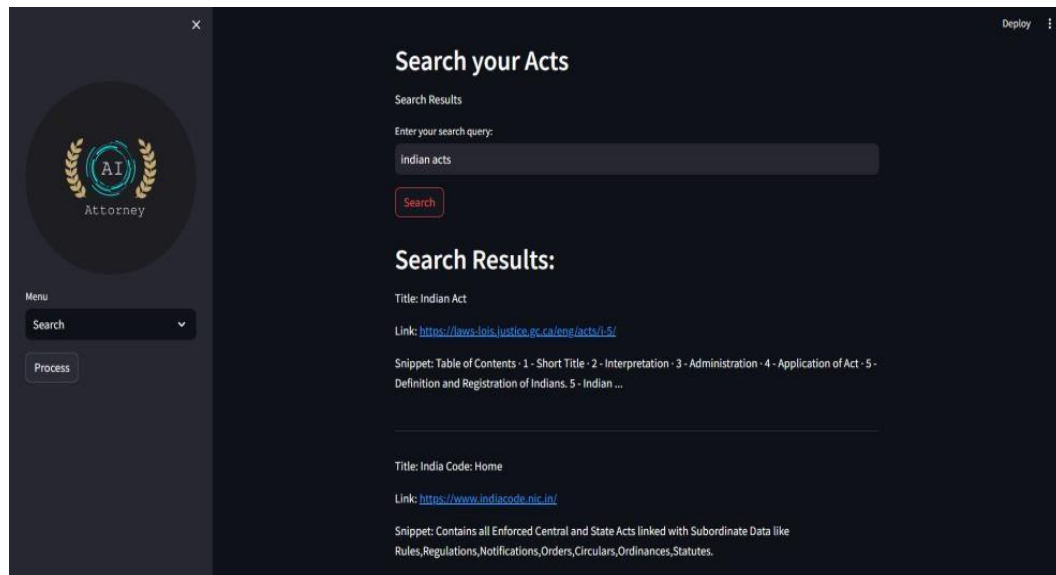


Figure 10: Search page

Utilizing Google's search engine infrastructure to give users rapid and easy access to relevant information and resources about legal acts, the "Search" code in the AI Attorney project streamlines the research process and adds to the overall efficacy of the project by providing precise and pertinent legal insights and materials. The "Search" code is implemented using Streamlit and Google Custom Search JSON API and allows users to input search queries, fetching and displaying relevant search results, including titles, links, snippets, and images. This integration ensures that users can quickly find authoritative and up-to-date legal information from a vast array of sources. The interface is designed to be intuitive, with a clean layout that highlights search results for easy navigation. Advanced filtering options enable users to refine their searches based on specific legal criteria, ensuring that the information retrieved is both relevant and specific to their needs. By incorporating this powerful search functionality, the AI Attorney project significantly enhances users' ability to conduct thorough legal research efficiently, saving time and improving the quality of their legal inquiries. This feature underscores the project's commitment to leveraging modern technology to make legal information more accessible and user-friendly.

Understand Me!! Page:

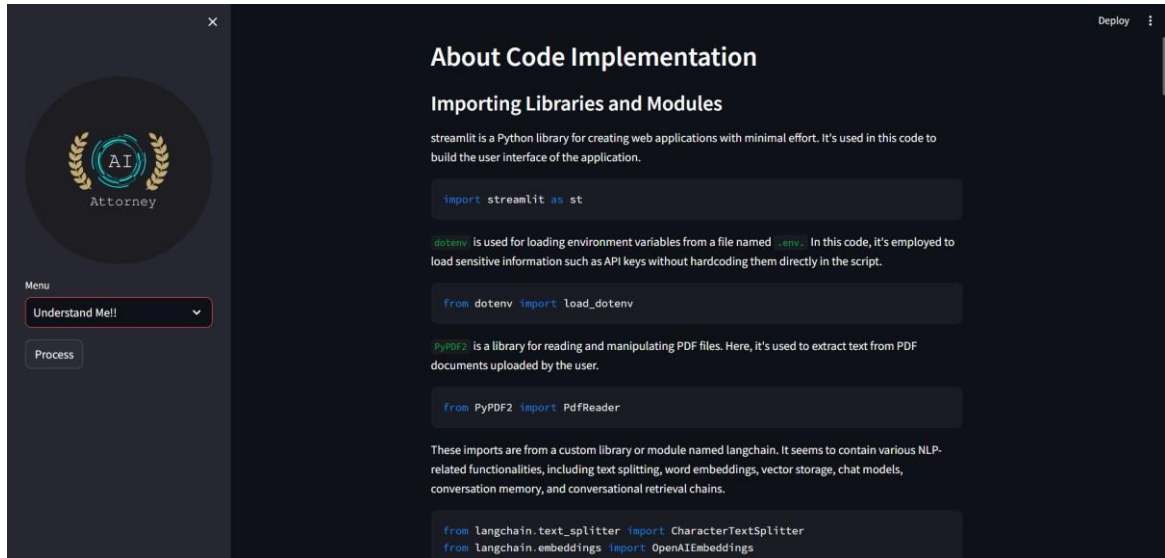


Figure 11: Understand Me Page

This page offers an in-depth guide to an AI-driven web application developed using Streamlit. It begins by detailing the various libraries and modules employed, such as Streamlit for the user interface, 'dotenv' for handling environment variables, and 'PyPDF2' for extracting text from PDF documents. It also explains the use of custom modules from 'langchain' for natural language processing tasks. The guide breaks down key functions within the application: extracting text from PDFs, splitting large texts into smaller chunks, creating a vector store with text embeddings from OpenAI, and establishing a conversational AI model using 'ChatOpenAI'. The 'main()' function orchestrates the application's operations, including setting up the Streamlit page, managing user navigation, processing uploaded PDFs, and handling user input to generate AI responses. Additionally, the page includes a contact form for user inquiries and applies custom CSS for improved styling. This comprehensive explanation not only documents the code but also provides users with the knowledge to understand and extend the application, offering both technical insights and practical implementation advice.

Today's News Page:

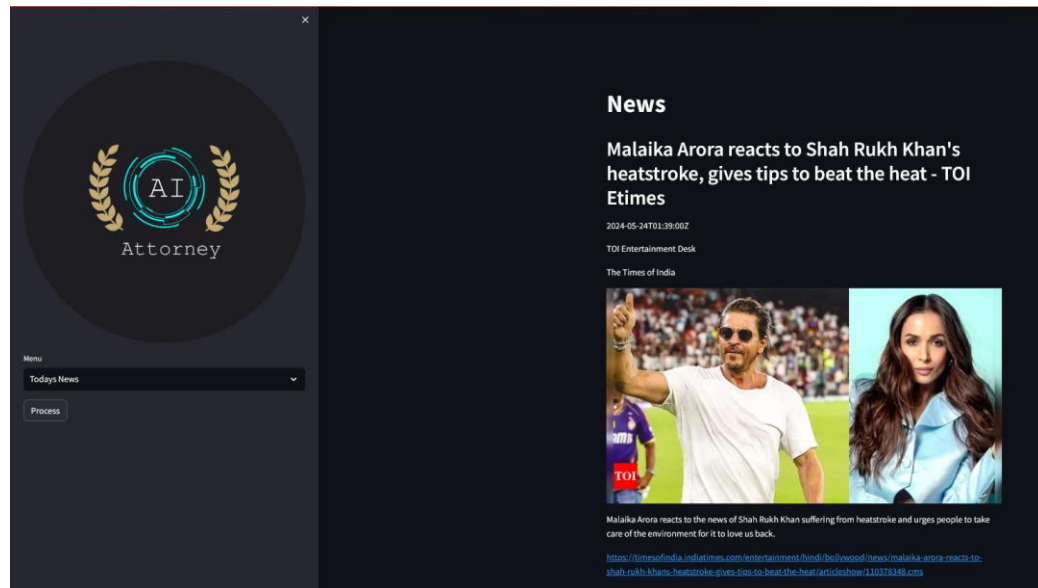


Figure 12: Today's News Page

Using the News API, the "Today's News" code, which is implemented in Streamlit, dynamically fetches and displays the top headlines from India. It provides real-time news updates with information about titles, publication dates, authors, news sources, article images, descriptions, and direct links for more reading. This makes the project's "Today's News" feature easily accessible and educational, keeping users up to date on current events. The news section is designed to present information in a clean, organized manner, ensuring that users can quickly scan through the latest headlines and find stories of interest. By providing direct links, users can easily navigate to the full articles for more in-depth reading. Additionally, the feature can be customized to filter news based on specific legal topics or regions, making it a valuable tool for legal professionals who need to stay informed about relevant developments. This dynamic and responsive news feed not only enhances the overall user experience but also ensures that users are always equipped with the most current information, further solidifying the AI Attorney project as a comprehensive resource for legal knowledge and updates.

CHAPTER 3

RESULTS AND DISCUSSIONS

a. Chat With Me!!

User: What is IPC Section 307?

Bot Output:

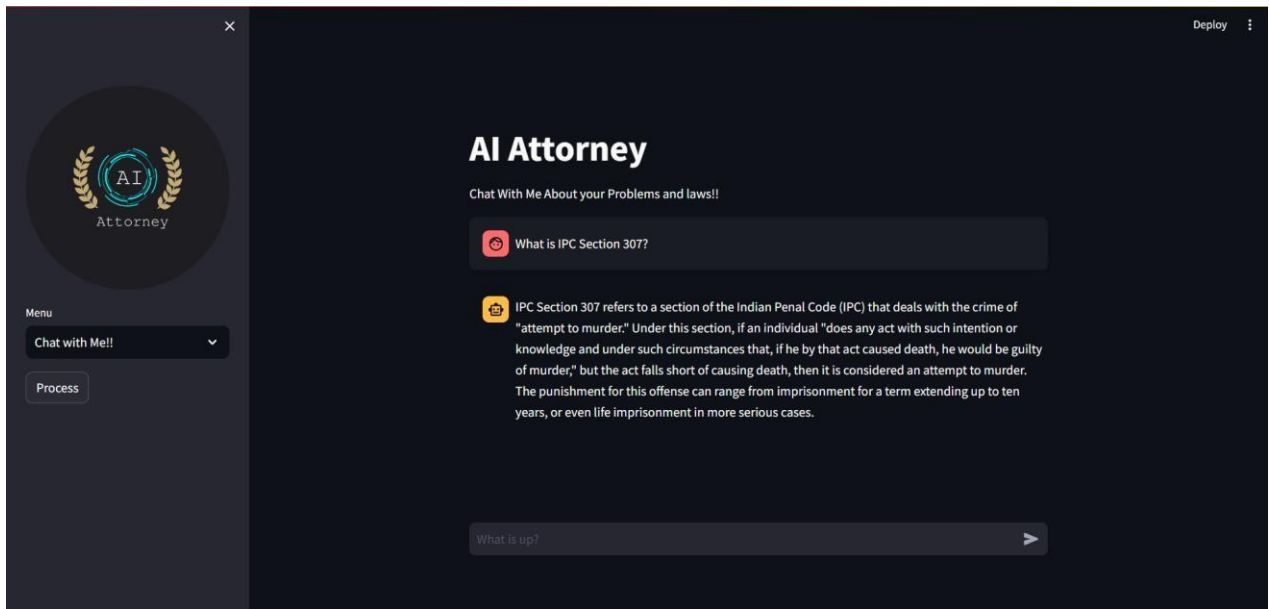


Figure 13: Result From Chat with Me

b. Chat With Document!!

First user has Uploaded the Document called Acts

User: what Indian Acts and Sections are applicable for Scenario: A person fires a gun at another with the intent to kill but misses the target.

Bot Output:

The relevant Indian Act in this scenario would be the Indian Penal Code (IPC). The specific sections that would apply are:

Section 307: Attempt to murder - This section deals with the act of attempting to commit murder. It states that whoever does any act with the intention of causing the death of another person, but the act does not result in the death of that person, shall be punished with imprisonment for a term which may extend to life imprisonment or for a term which may extend to ten years, and shall also be liable to fine.

Additionally, depending on the circumstances, other relevant sections of the IPC may also apply, such as:

Section 34: Acts done by several persons in furtherance of common intention - This section deals with situations where a criminal act is done by several individuals in furtherance of a common intention. It states that when a criminal act is done by several persons, each person is liable for the act as if he/she had done it alone.

Section 120B: Criminal conspiracy - This section deals with the offense of criminal conspiracy, which is an agreement between two or more persons to commit a criminal act. It states that whoever is a party to a criminal conspiracy to commit an offense punishable with death, imprisonment for life, or rigorous imprisonment for a term of two years or upwards, shall be punished in the same manner as if he/she had actually committed the offense.

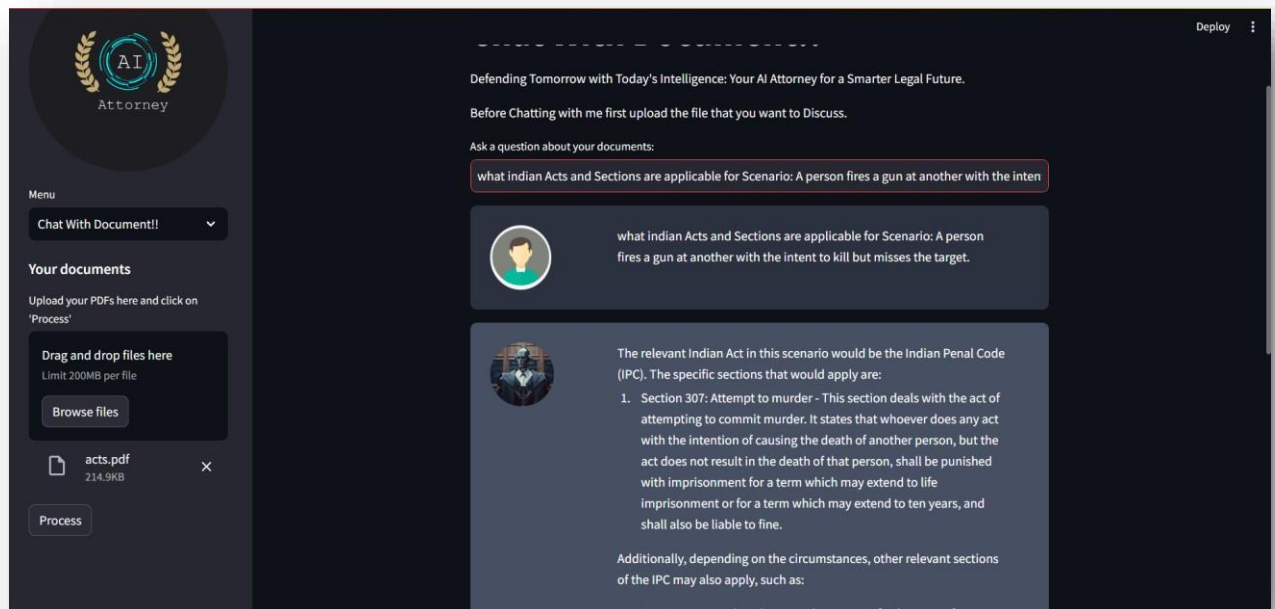


Figure 14: Result from Chat with Document

CHAPTER 4

CONCLUSION AND FUTURE WORK

To sum up, the AI Attorney project's development and execution represent a major advancement toward the creation of a more sophisticated and approachable legal assistance system. The incorporation of OpenAI models, specifically the ChatOpenAI model, has shown encouraging results in comprehending and reacting to user inquiries, creating a stimulating and educational atmosphere. The document processing feature is very effective at retrieving data from uploaded PDFs, which makes legal documents easier for users to access and use. The project successfully validates the potential of advanced natural language processing in the legal domain by offering a conversational interface for legal inquiries.

There are numerous opportunities for further improvements and modifications in the future. The ChatOpenAI model's contextual understanding and responsiveness can be further improved by increasing the amount of training data available to it. Furthermore, investigating domain-specific embeddings and customized training strategies may improve the model's accuracy for legal jargon and nuance.

Expanding the scope of document formats handled and increasing the precision of text extraction are potential areas of future development for the document processing module. Assuring the AI Attorney's applicability and accuracy in a variety of legal scenarios would require working with legal experts to customize the system for particular use cases. Iterative updates and constant user feedback will be essential for improving and developing the system, which will ultimately contribute to the ongoing development of AI in the legal sector.

Overall, continuous innovation and collaboration with legal experts, technology partners, and end-users will be crucial for driving the AI Attorney project forward and ensuring its continued evolution to meet the evolving needs of the legal community. By embracing these opportunities for growth and refinement, the AI Attorney project has the potential to become an indispensable tool for legal professionals worldwide.

CHAPTER 5

Appendix

HOME PAGE CODE:

```
# home.py
import streamlit as st
import os
import numpy as np
import pandas as pd
def home_page():
    st.title("AI Attorney")
    st.markdown("Transforming the Legal Industry with AI")
    st.caption("AI Attorney is a cutting-edge legal technology firm revolutionizing the legal industry with advanced artificial intelligence solutions. We provide clients with efficient, accurate, and cost-effective legal services, powered by state-of-the-art AI algorithms and expert legal insights, ensuring seamless and precise legal support.")
    st.image(r"..\\ask-multiple-pdfs-main\\ask-multiple-pdfs-main\\images\\AiLaw.jpeg")

    st.markdown(
        "AI Attorney is at the forefront of revolutionizing the legal industry by harnessing the power of artificial intelligence. "
        "Our commitment is to provide innovative and efficient solutions that transform traditional legal processes. "
        "With cutting-edge AI algorithms and expert legal insights, we strive to deliver accurate and cost-effective legal services to our clients."
    )

    st.markdown(
        "Our team of dedicated professionals is passionate about leveraging technology to address the complexities of the legal landscape. "
```

```

        "We believe in the seamless integration of AI into legal workflows,
empowering legal professionals and organizations to work smarter and achieve
better outcomes."
    )

    st.markdown(
        "At AI Attorney, we constantly push the boundaries of what AI can achieve
in the legal domain. Our mission is to redefine how legal services are delivered,
making them more accessible, precise, and adaptable to the evolving needs of our
clients."
    )

    st.text(" ")
    st.text(" ")
    st.text(" ")
    st.text(" ")

    st.header("OBJECTIVE")
    col1, col2 = st.columns(2)
    with col1:
        st.image(r"..\\ask-multiple-pdfs-main\\ask-multiple-pdfs-main\\images\\law
2.png")
    with col2:
        st.write("""The objective of this presentation is to demonstrate the
groundbreaking capabilities and significant contributions of Ai Attorney, an
advanced AI powered system, in revolutionizing legal research. Through its
utilization of cutting-edge Natural Language Processing (NLP) algorithms, Ai
Attorney offers users precise and pertinent legal materials based on input
scenarios or case PDF files. This presentation aims to highlight the system's
meticulous keyword extraction process, its ability to swiftly analyze and
recognize relevant legal sections within uploaded case files, and its ongoing
enhancement through user interactions. The presentation seeks to underscore how Ai
Attorney substantially improves the efficiency and accuracy of legal research,
outperforming conventional tools.""")

    st.text(" ")

```

```

st.text(" ")
st.text(" ")
st.text(" ")
st.header("Motivation: Revolutionizing Legal Assistance with AI")
st.write("""Welcome to AI Attorney, where cutting-edge artificial intelligence
meets legal documentation!

    Our motivation stems from a deep commitment to making legal processes more
accessible, efficient, and user-friendly. Here's why AI Attorney is poised to
revolutionize the legal assistance landscape:""")

st.markdown("""

    ### Bridging the Gap

    Legal documents can be complex and intimidating, often creating a
significant barrier

    between individuals and their understanding of legal matters. AI Attorney
bridges this gap by

    providing an intuitive platform where users can ask questions about their
documents in plain language.

    ### Empowering Users

    We believe that everyone has the right to comprehend and engage with legal
documents that concern them.

    AI Attorney empowers users by offering a conversational interface,
enabling them to seek clarification,

    explanations, and guidance on legal jargon, contracts, and other legal
texts.

    ### Enhancing Efficiency

    Traditional legal processes can be time-consuming. AI Attorney streamlines
the document review

    and understanding process, offering quick and accurate responses to user
queries. This efficiency

    ensures that users can make informed decisions promptly.

    ### Leveraging Advanced AI Technology

```


AI Attorney leverages state-of-the-art natural language processing and conversational AI models.

Our technology can understand and respond to a wide range of legal questions, providing users with valuable insights and information.

Continuous Improvement

We are committed to constant improvement. AI Attorney's AI models learn and adapt over time, ensuring that the platform evolves to meet the diverse and dynamic needs of its users.

Explore AI Attorney and experience the future of legal assistance - where technology meets accessibility, empowering individuals to navigate legal complexities with confidence.

Remember, AI Attorney is here to assist, simplify, and make legal matters more understandable for everyone.

""")

st.text(" ")

st.text(" ")

st.text(" ")

st.text(" ")

st.subheader("Current Location")

```
df = pd.DataFrame(  
    np.random.randn(1000, 2) / [50, 50] + [16.4942796, 80.5007368],  
    columns=['lat', 'lon'])
```

st.map(df)

```

st.text(" ")
st.text(" ")
st.text(" ")
st.text(" ")

st.header(":mailbox: Get In Touch With Me!")
contact_form = """
<form action="https://formsubmit.co/YOUREMAIL@EMAIL.COM" method="POST">
    <input type="hidden" name="_captcha" value="false">
    <input type="text" name="name" placeholder="Your name" required>
    <input type="email" name="email" placeholder="Your email" required>
    <textarea name="message" placeholder="Your message here"></textarea>
    <button type="submit">Send</button>
</form>
"""

st.markdown(contact_form, unsafe_allow_html=True)

# Use Local CSS File
def local_css(file_name):
    with open(file_name) as f:
        st.markdown(f"<style>{f.read()}</style>", unsafe_allow_html=True)

local_css(r"..\\ask-multiple-pdfs-main\\style\\style.css")

```

ABOUT PAGE CODE:

```

import streamlit as st
import time

def fade_in_animation(element, duration=0.5):
    st.markdown(
        f"""
        <style>
        @keyframes fade-in {{

```

```

        from {{ opacity: 0; }}
        to {{ opacity: 1; }}
    }}
    {element} {{
        animation: fade-in {duration}s ease-in-out;
    }}
</style>
""",
    unsafe_allow_html=True
)

def about_page():
    st.title("About AI Attorney")
    st.subheader("Empowering Legal Excellence Through AI")
    col1, col2 = st.columns(2)
    with col2:
        st.image(r"..\\ask-multiple-pdfs-main\\ask-multiple-pdfs-
main\\images\\ailogo.png")
    with col1:
        st.text(" ")
        st.text(" ")
        st.write("At AI Attorney, we are at the forefront of the legal technology
revolution. Our mission is to empower legal professionals with cutting-edge AI
solutions that redefine the practice of law. With a relentless commitment to
innovation, we have developed a suite of services designed to enhance the legal
industry's efficiency, accuracy, and client service. Here's what sets us apart:")
        st.write(" Expertise: Our team combines legal expertise with AI prowess to
create solutions that truly understand the intricacies of the law.")
        st.write(" Innovation: We constantly push boundaries, developing custom AI
tools that adapt to the ever-changing legal landscape.")
        st.write(" Efficiency: AI Attorney's solutions streamline legal processes,
allowing you to focus on what matters most - your clients.")
        st.write(" Accuracy: Our AI algorithms deliver unparalleled precision in legal
research, analysis, and document review.")
        st.write(" Client-Centric: We prioritize your success, aiming to elevate your
practice to new heights while ensuring your clients receive top-notch service.")
        st.write("Join AI Attorney on the journey to redefine legal excellence in the
digital age.")
        st.image(r"..\\ask-multiple-pdfs-main\\ask-multiple-pdfs-main\\images\\about.png")

    st.header("Product Features")
    st.markdown("AI Attorney stands as a pioneering force in the legal technology
landscape, harnessing the power of Artificial Intelligence (AI) and Natural
Language Processing (NLP) techniques to revolutionize legal research. Here are the
key features that set AI Attorney apart")
    col1, col2 = st.columns(2)
    with col2:

```

```

        st.image(r"..\ask-multiple-pdfs-main\ask-multiple-pdfs-main\images\img-
1.png")
        with col1:
            st.subheader("Cutting-Edge NLP Technology")
            st.markdown(
                """
                - Utilizes advanced NLP algorithms for precise legal document
analysis.
                - Ensures accurate keyword extraction, improving the relevance of
search results.
                - Integrates the powerful Natural Language Toolkit (NLTK) library for
robust language processing.
                - Enhances legal research efficiency with AI-driven language
understanding.
                """
            )

        with col1:
            st.image(r"..\ask-multiple-pdfs-main\ask-multiple-pdfs-main\images\img-
2.png")

        with col2:
            st.subheader("Scenario-Based Search")
            st.markdown(
                """
                - Empowers users to input specific legal scenarios for tailored
research.
                - Aligns research outcomes with real-world legal situations.
                - Provides contextually relevant legal articles and sections.
                - Streamlines the process of finding pertinent legal information.
                """
            )

        with col2:
            st.image(r"..\ask-multiple-pdfs-main\ask-multiple-pdfs-main\images\img-
3.png")
        with col1:
            st.subheader("PDF File Analysis")
            st.markdown(
                """
                - Accelerates legal research by analyzing and interpreting case PDF
files.
                - Intelligently identifies act names and relevant sections within
uploaded documents.
                - Significantly reduces manual effort in reviewing case materials.
                - Enhances productivity for legal experts dealing with extensive case
files.
            """

```

```

        """
    )

    with col1:
        st.image(r"..ask-multiple-pdfs-main\ask-multiple-pdfs-main\images\img-4.png")
    with col2:
        st.subheader("Keyword Extraction Expertise")
        st.markdown(
            """
            - Meticulously crafted keyword extraction process for precision and relevance.
            - Guarantees contextually accurate and meaningful search results.
            - Improves the overall effectiveness of legal research.
            - Minimizes irrelevant content, saving users valuable time.
            """
        )

    with col2:
        st.image(r"..ask-multiple-pdfs-main\ask-multiple-pdfs-main\images\img-3.png")
    with col1:
        st.subheader("Adaptive Learning")
        st.markdown(
            """
            - Continuously refines keyword extraction based on user interactions.
            - Adapts to evolving legal language and terminology.
            - Ensures ongoing improvement in search accuracy.
            - Customizes results to match the evolving needs of legal professionals.
            """
        )

    with col1:
        st.image(r"..ask-multiple-pdfs-main\ask-multiple-pdfs-main\images\img-1.png")
    with col2:
        st.subheader("Multilingual Text Translation")
        st.markdown(
            """
            - Breaks language barriers by providing multilingual text translation.
            - Expands accessibility for legal professionals working in diverse linguistic contexts.
            - Enables seamless research in international legal scenarios.
            - Enhances collaboration and understanding in global legal environments.
            """
        )

```

```

)

st.header(":mailbox: Get In Touch With Me!")

contact_form = """
<form action="https://formsubmit.co/YOUREMAIL@EMAIL.COM" method="POST">
  <input type="hidden" name="_captcha" value="false">
  <input type="text" name="name" placeholder="Your name" required>
  <input type="email" name="email" placeholder="Your email" required>
  <textarea name="message" placeholder="Your message here"></textarea>
  <button type="submit">Send</button>
</form>
"""

st.markdown(contact_form, unsafe_allow_html=True)

# Use Local CSS File
def local_css(file_name):
    with open(file_name) as f:
        st.markdown(f"<style>{f.read()}</style>", unsafe_allow_html=True)

local_css(r"..\\ask-multiple-pdfs-main\\style\\style.css")

```

CHAT WITH ME! CODE:

```

import openai
import streamlit as st

def chat_page():
    st.title("AI Attorney")
    st.markdown("Chat With Me About your Problems and laws!!")

    openai.api_key = "sk-proj-UiFjaMCRAPoWn8PAM7bxT3B1bkFJlFATLXxaPYxSR5sFKWY"
    if "openai_model" not in st.session_state:
        st.session_state["openai_model"] = "gpt-3.5-turbo"

    if "messages" not in st.session_state:
        st.session_state.messages = []

    for message in st.session_state.messages:
        with st.chat_message(message["role"]):
            st.markdown(message["content"])

    prompt = st.chat_input("What is up?")
    if prompt:
        st.session_state.messages.append({"role": "user", "content": prompt})
        with st.chat_message("user"):

```

```

st.markdown(prompt)

with st.spinner("Thinking..."):
    # Removed stream=True from the openai.ChatCompletion.create call
    response = openai.ChatCompletion.create(
        model=st.session_state["openai_model"],
        messages=[
            {"role": m["role"], "content": m["content"]}
            for m in st.session_state.messages
        ],
    )
    full_response = response.choices[0].get("message", {}).get("content", "")
    st.session_state.messages.append({"role": "assistant", "content":
full_response})

with st.chat_message("assistant"):
    st.markdown(full_response)

```

CHAT WITH DOCUMENT CODE:

```

import streamlit as st
from dotenv import load_dotenv
from PyPDF2 import PdfReader
from langchain.text_splitter import CharacterTextSplitter
from langchain_community.embeddings import OpenAIEmbeddings
from langchain_community.vectorstores import FAISS
from langchain_community.chat_models import ChatOpenAI
from langchain.memory import ConversationBufferMemory
from langchain.chains import ConversationalRetrievalChain
from htmlTemplates import css, bot_template, user_template
from streamlit_option_menu import option_menu
from about import about_page
from home import home_page
from codeexplain import code_page
from news import news_page
from ActsSearch import search_page
from chat import chat_page
import os

def get_pdf_text(pdf_docs):
    text = ""
    for pdf in pdf_docs:
        pdf_reader = PdfReader(pdf)
        for page in pdf_reader.pages:
            text += page.extract_text()
    return text

```

```

def get_text_chunks(text):
    text_splitter = CharacterTextSplitter(
        separator="\n",
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len
    )
    chunks = text_splitter.split_text(text)
    return chunks

def get_vectorstore(text_chunks):
    openai_api_key = "sk-proj-UiFjaMCRApOwN8PAM7bxT3BlbkFJUlFATLXxaPYxSR5sFKWY"
    embeddings = OpenAIEmbeddings(openai_api_key=openai_api_key)
    vectorstore = FAISS.from_texts(texts=text_chunks, embedding=embeddings)
    return vectorstore

def get_conversation_chain(vectorstore):
    openai_api_key = "sk-proj-UiFjaMCRApOwN8PAM7bxT3BlbkFJUlFATLXxaPYxSR5sFKWY"
    llm = ChatOpenAI(openai_api_key=openai_api_key)
    memory = ConversationBufferMemory(
        memory_key='chat_history', return_messages=True)
    conversation_chain = ConversationalRetrievalChain.from_llm(
        llm=llm,
        retriever=vectorstore.as_retriever(),
        memory=memory
    )
    return conversation_chain

def handle_userinput(user_question):
    response = st.session_state.conversation({'question': user_question})
    st.session_state.chat_history = response['chat_history']

    for i, message in enumerate(st.session_state.chat_history):
        if i % 2 == 0:
            st.write(user_template.replace(
                "{{MSG}}", message.content), unsafe_allow_html=True)
        else:
            st.write(bot_template.replace(
                "{{MSG}}", message.content), unsafe_allow_html=True)

def main():
    load_dotenv()
    st.set_page_config(page_title="AI Attorney", page_icon=r"..\\ask-multiple-pdfs-
main\\ask-multiple-pdfs-main\\images\\cropped logo.PNG")
    st.write(css, unsafe_allow_html=True)

```



```

if "conversation" not in st.session_state:
    st.session_state.conversation = None
if "chat_history" not in st.session_state:
    st.session_state.chat_history = None

st.sidebar.image(r"..\ask-multiple-pdfs-main\ask-multiple-pdfs-
main\images\cropped logo.PNG", use_column_width=True)
st.markdown(
    """
        <style>
            @keyframes fadeIn {
                from {
                    opacity: 0;
                    transform: translateY(-10px);
                }
                to {
                    opacity: 1;
                    transform: translateY(0);
                }
            }

            .sidebar .sidebar-content {
                background-color: #f8f9fa;
                box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.1);
                border-radius: 10px;
                animation: fadeIn 0.5s ease-out;
            }

            .sidebar .sidebar-content .block-container {
                padding: 1em;
            }
        </style>
    """,
    unsafe_allow_html=True
)

# Sidebar navigation
navigation = st.sidebar.selectbox("Menu", ["Home", "About", "Chat with Me!!",
"Chat With Document!!", "Search", "Understand Me!!", "Todays News"])

if navigation == "Home":
    home_page()
elif navigation == "About":
    about_page()
elif navigation == "Understand Me!!":
    code_page()

```

```

elif navigation == "Todays News":
    news_page()
elif navigation == "Search":
    search_page()
elif navigation == "Chat with Me!!":
    chat_page()
elif navigation == "Chat With Document!!":
    st.title("Chat With Document!!")
    st.markdown("Defending Tomorrow with Today's Intelligence: Your AI
Attorney for a Smarter Legal Future.")
    st.write("Before Chatting with me first upload the file that you want to
Discuss.")
    user_question = st.text_input("Ask a question about your documents:")
    st.sidebar.subheader("Your documents")
    pdf_docs = st.sidebar.file_uploader("Upload your PDFs here and click on
'Process'", accept_multiple_files=True)
    if user_question:
        handle_userinput(user_question)

# File upload section outside the "Chat" condition

if st.sidebar.button("Process"):
    with st.spinner("Processing"):
        # get pdf text
        raw_text = get_pdf_text(pdf_docs)

        # get the text chunks
        text_chunks = get_text_chunks(raw_text)

        # create vector store
        vectorstore = get_vectorstore(text_chunks)

        # create conversation chain
        st.session_state.conversation = get_conversation_chain(vectorstore)

if __name__ == '__main__':
    main()

```

SEARCH ACT CODE:

```

import streamlit as st
from googleapiclient.discovery import build

def fetch_google_search_results(query, cx, api_key):
    resource = build("customsearch", 'v1', developerKey=api_key).cse()
    result = resource.list(q=query, cx=cx).execute()
    return result['items']

```

```

def search_page():
    # Your Google API key and Custom Search Engine ID
    api_key = "AIzaSyC4xwBIApKA8XJ-gLUKOL04Ms6a4X0BLIg"
    cx = "807d5be983eda4de8"
    st.header("Search your Acts")

    st.write("Search Results")

    # Input box for search query
    query = st.text_input("Enter your search query:")

    if st.button("Search"):
        results = fetch_google_search_results(query, cx, api_key)

        # Display results
        st.header("Search Results:")
        for item in results:
            st.write(f"Title: {item['title']}")
            st.write(f"Link: {item['link']}")
            st.write(f"Snippet: {item['snippet']}")
            if 'cse_image' in item['pagemap']:
                st.image(item['pagemap']['cse_image'][0]['src'],
caption="Image", use_column_width=True)
            st.write("-" * 50)

```

UNDERSTAND ME! CODE:

```

import streamlit as st

def code_page():
    st.header("About Code Implementation")
    st.subheader("Importing Libraries and Modules")

    st.write("streamlit is a Python library for creating web applications with
minimal effort. It's used in this code to build the user interface of the
application.")
    st.code("""
import streamlit as st
""")

    st.write("`dotenv` is used for loading environment variables from a file named
`.env`. In this code, it's employed to load sensitive information such as API keys
without hardcoding them directly in the script.")
    st.code("""
from dotenv import load_dotenv
""")

```

```

st.write("`PyPDF2` is a library for reading and manipulating PDF files. Here,
it's used to extract text from PDF documents uploaded by the user.")
st.code("""
    from PyPDF2 import PdfReader
    """)

st.write("These imports are from a custom library or module named langchain.
It seems to contain various NLP-related functionalities, including text splitting,
word embeddings, vector storage, chat models, conversation memory, and
conversational retrieval chains.")
st.code("""
    from langchain.text_splitter import CharacterTextSplitter
    from langchain.embeddings import OpenAIEmbeddings
    from langchain.vectorstores import FAISS
    from langchain.chat_models import ChatOpenAI
    from langchain.memory import ConversationBufferMemory
    from langchain.chains import ConversationalRetrievalChain
    """)

st.write("These imports are from a module named `htmlTemplates`. It appears to
include HTML templates for styling the user interface of the web application. css
is likely a style sheet, while `bot_template` and `user_template` might be
templates for displaying bot and user messages in the chat.")
st.code("""
    from htmlTemplates import css, bot_template, user_template
    """)

st.write("This import is from `langchain.llms` and seems to involve a language
model (LLM) from Hugging Face's model hub. Hugging Face is a platform for sharing
natural language processing models.")
st.code("""
    from langchain.llms import HuggingFaceHub
    """)

st.write("This import is for an additional Streamlit component that provides
an option menu. It might be used to create a dropdown menu or similar interactive
features in the web application.")
st.code("""
    from streamlit_option_menu import option_menu
    """)

st.write("These imports are from modules named about and home. They likely
contain functions that define the content for the About and Home sections of the
web application, respectively.")
st.code("""
    from about import about_page

```

```

from home import home_page
from codeexplain import code_page
    """

    st.write("This import is for the os module, which provides a way of using
operating system-dependent functionality, such as reading or writing to the file
system. Its specific use in this code may not be clear without additional
context.")
    st.code("import os")

    st.subheader("Function to Extract Text from PDFs")
    st.write("This function takes a list of PDF documents (`pdf_docs`) as input
and extracts text from each page of every PDF using `PyPDF2`. The extracted text
is concatenated and returned.")
    st.code(
        """
        def get_pdf_text(pdf_docs):
            text = ""
            for pdf in pdf_docs:
                pdf_reader = PdfReader(pdf)
                for page in pdf_reader.pages:
                    text += page.extract_text()
            return text
        """
    )

    st.subheader("Function to Split Text into Chunks")
    st.write("This function takes a large text and splits it into smaller chunks
using a custom `CharacterTextSplitter` class. The chunks have a specified size
with some overlap.")
    st.code("""
        def get_text_chunks(text):
            text_splitter = CharacterTextSplitter(
                separator="\n",
                chunk_size=1000,
                chunk_overlap=200,
                length_function=len
            )
            chunks = text_splitter.split_text(text)
            return chunks
        """)

    st.subheader("Function to Create Vector Store")
    st.write("This function creates a vector store using the `FAISS` library. It
takes the text chunks and represents them as vectors using OpenAI `GPT-3.5's` word
embeddings.")

```

```

st.code(
    """
    def get_vectorstore(text_chunks):
        openai_api_key = "API_KEY"
        embeddings = OpenAIEmbeddings(openai_api_key=openai_api_key)
        vectorstore = FAISS.from_texts(texts=text_chunks,
embedding=embeddings)
        return vectorstore

    """
)

st.subheader("Function to Create Conversation Chain")
st.write("This function creates a conversational chain, combining the
conversational model (`ChatOpenAI`), vector retriever, and conversation memory
(`ConversationBufferMemory`).")
st.code(
    """
    def get_conversation_chain(vectorstore):
        openai_api_key = "API_KEY"
        llm = ChatOpenAI(openai_api_key=openai_api_key)
        memory = ConversationBufferMemory(
            memory_key='chat_history', return_messages=True)
        conversation_chain = ConversationalRetrievalChain.from_llm(
            llm=llm,
            retriever=vectorstore.as_retriever(),
            memory=memory
        )
        return conversation_chain

    """
)

st.subheader("Function to Handle User Input")
st.write("This function takes a user's question, interacts with the
conversation chain, and updates the chat history in the application's session
state. It then displays the conversation in the UI using HTML templates for user
and bot messages.")
st.code(
    """
    def handle_userinput(user_question):
        response = st.session_state.conversation({'question': user_question})
        st.session_state.chat_history = response['chat_history']

        for i, message in enumerate(st.session_state.chat_history):
            if i % 2 == 0:
                st.write(user_template.replace(

```

```

        "{{MSG}}", message.content), unsafe_allow_html=True)
    else:
        st.write(bot_template.replace(
            "{{MSG}}", message.content), unsafe_allow_html=True)

    """
)

st.subheader("Main Function")
st.markdown(
    """
    - The `main()` function is the entry point of the script.
    - It initializes the Streamlit page, sets configurations, and defines the
    UI structure.
    - It handles user input, navigation, and the processing of uploaded PDF
    documents.
    - The functions defined earlier are called within the main function to
    orchestrate the application's behavior.
    """
)
st.code(
    """
    def main():
        load_dotenv()
        st.set_page_config(page_title="AI Attorney", page_icon=r"..\\ask-
multiple-pdfs-main\\ask-multiple-pdfs-main\\images\\AI-Attorny-Logo.png")
        st.write(css, unsafe_allow_html=True)

        # Initialization of session state variables
        if "conversation" not in st.session_state:
            st.session_state.conversation = None
        if "chat_history" not in st.session_state:
            st.session_state.chat_history = None

        st.header("AI Attorney")
        user_question = st.text_input("Ask a question about your documents:")

        # Create a sidebar navigation
        navigation = st.sidebar.radio("Menu", ["Home", "About", "Chat"])

        # Handling different sections based on navigation
        if navigation == "Home":
            home_page()
        elif navigation == "About":
            about_page()
        elif navigation == "Chat":
            if user_question:

```

```

        handle_userinput(user_question)

    # File upload section outside the "Chat" condition
    st.sidebar.subheader("Your documents")
    pdf_docs = st.sidebar.file_uploader("Upload your PDFs here and click
on 'Process'", accept_multiple_files=True)
    if st.sidebar.button("Process"):
        with st.spinner("Processing"):
            # get pdf text
            raw_text = get_pdf_text(pdf_docs)

            # get the text chunks
            text_chunks = get_text_chunks(raw_text)

            # create vector store
            vectorstore = get_vectorstore(text_chunks)

            # create conversation chain
            st.session_state.conversation =
get_conversation_chain(vectorstore)

    if __name__ == '__main__':
        main()

    """
)

st.write(
    """
    The `main` function serves as the central control point for the AI
Attorney application. Let's break down its functionality:

```

The initial part of the function involves setting up the environment and configuring the Streamlit page. It loads environment variables using ``load_dotenv()`` and configures the Streamlit page with a specific title and icon representing the AI Attorney logo. The application's CSS styles are applied using ``st.write(css, unsafe_allow_html=True)``.

Session state variables are initialized to manage the conversation and chat history. If these variables are not already present, they are set to ``None``.

The main body of the function is dedicated to creating the user interface using Streamlit. It begins with a header displaying "AI Attorney" and provides a text input field for users to ask questions about their documents. A sidebar navigation menu offers options such as "Home," "About," and "Chat."

The application dynamically handles different sections based on the user's navigation choice. If the user selects "Home" or "About," corresponding pages are displayed using functions like ``home_page()`` and ``about_page()``. If the user chooses the "Chat" section and provides a question, the ``handle_userinput`` function is invoked to process the user's input and generate responses.

Outside the "Chat" condition, there is a file upload section in the sidebar where users can upload multiple PDF documents. Upon clicking the "Process" button, the application initiates processing. It extracts text from the uploaded PDFs using the ``get_pdf_text`` function, then splits the text into chunks using ``get_text_chunks``. These text chunks are used to create a vector store with ``get_vectorstore``, and a conversational chain is established with ``get_conversation_chain``. This chain is stored in the session state to maintain context across interactions.

In summary, the ``main`` function orchestrates the behavior of the AI Attorney application, handling user input, navigation, and document processing to create an interactive and responsive conversational experience.

```
"""
)

st.header(":mailbox: Get In Touch With Me!")

contact_form = """
<form action="https://formsubmit.co/YOUREMAIL@EMAIL.COM" method="POST">
  <input type="hidden" name="_captcha" value="false">
  <input type="text" name="name" placeholder="Your name" required>
  <input type="email" name="email" placeholder="Your email" required>
  <textarea name="message" placeholder="Your message here"></textarea>
  <button type="submit">Send</button>
</form>
"""

st.markdown(contact_form, unsafe_allow_html=True)

# Use Local CSS File
def local_css(file_name):
    with open(file_name) as f:
        st.markdown(f"<style>{f.read()}</style>", unsafe_allow_html=True)

local_css(r"..\ask-multiple-pdfs-main\style\style.css")
```

TODAYS NEWS CODE:

```
import streamlit as st
import requests
import pycountry

def news_page():
    st.title("News")
    url = f"https://newsapi.org/v2/top-headlines?country=in&apiKey=757271b3c2c5422388ffb027ce0f0f90"

    r = requests.get(url)
    r = r.json()
    articles = r['articles']
    for article in articles:
        st.header(article['title'])
        st.write(article['publishedAt'])
        if article['author']:
            st.write(article['author'])
        st.write(article['source']['name'])
        if article['urlToImage']:
            st.image(article["urlToImage"])
        st.write(article['description'])
        if article["url"]:
            st.write(article['url'])
```

STYLES.CSS CODE:

```
input[type=text], input[type=email], textarea {
    width: 100%; /* Full width */
    padding: 12px; /* Some padding */
    border: 1px solid #ccc; /* Gray border */
    border-radius: 4px; /* Rounded borders */
    box-sizing: border-box; /* Make sure that padding and width stays in place */
    margin-top: 6px; /* Add a top margin */
    margin-bottom: 16px; /* Bottom margin */
    resize: vertical /* Allow the user to vertically resize the textarea (not
horizontally) */
}

/* Style the submit button with a specific background color etc */
button[type=submit] {
    background-color: #04AA6D;
    color: white;
    padding: 12px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
```

```
/* When moving the mouse over the submit button, add a darker green color */  
button[type=submit]:hover {  
    background-color: #45a049;  
}
```

REFERENCES

- [1] Deltsova, N. (2020). Artificial Intelligence Technologies in the Field of Legal Services: Relevant Aspects. Current Achievements, Challenges and Digital Chances of Knowledge Based Economy. https://doi.org/10.1007/978-3-030-47458-4_47.
- [2] Eliot, L. (2020). An Ontological AI-and-Law Framework for the Autonomous Levels of AI Legal Reasoning. ArXiv, abs/2008.07328.
- [3] Verheij, B. (2020). Artificial intelligence as law. Artificial Intelligence and Law, 28, 181206. <https://doi.org/10.1007/s10506-020-09266-0>.
- [4] Vasiliev, A., & Pechatnova, Y. (2020). The Position of the Artificial Intelligence Among the Elements of the Legal Relationship. Digital Law Journal. <https://doi.org/10.38044/26869136-2020-1-4-74-83>.
- [5] Li, J., Fan, Q., & Zhang, K. (2007). Keyword extraction based on tf/idf for Chinese news document. Wuhan University Journal of Natural Sciences, 12, 917-921. <https://doi.org/10.1007/s11859-007-0038-4>.
- [6] Wang, X., & Ning, H. (2020). TF-IDF Keyword Extraction Method Combining Context and Semantic Classification. Proceedings of the 3rd International Conference on Data Science and Information Technology. <https://doi.org/10.1145/3414274.3414492>.

- [7] Pan, S., Li, Z., & DAI, J. (2019). An improved TextRank keywords extraction algorithm. Proceedings of the ACM Turing Celebration Conference - China. <https://doi.org/10.1145/3321408.3326659>.
- [8] Sarmiento, R., Cordeiro, M., Brazdil, P., & Gama, J. (2018). Incremental TextRank - Automatic Keyword Extraction for Text Streams., 363-370. <https://doi.org/10.5220/0006639703630370>.
- [9] Huang, H., Wang, X., & Wang, H. (2020). NER-RAKE: An improved rapid automatic keyword extraction method for scientific literatures based on named entity recognition. Proceedings of the Association for Information Science and Technology, 57. <https://doi.org/10.1002/pa2.374>.
- [10] Haque, M. (2018). Automatic Keyword Extraction from Bengali Text Using Improved RAKE Approach. 2018 21st International Conference of Computer and Information Technology (ICCIT), 1-6. <https://doi.org/10.1109/ICCITECHN.2018.8631917>.
- [11] Qingyun, Z., Yuansheng, F., Zhenlei, S., & Wanli, Z. (2020). Keyword Extraction Method for Complex Nodes Based on TextRank Algorithm. 2020 International Conference on Computer Engineering and Application (ICCEA), 359-363. <https://doi.org/10.1109/ICCEA50009.2020.00084>.