MUHAMMAD ABDULLAH
01-135211-100
FARRAH
01-135211-028

Search Engine for Legal documents and pleaded cases in Courts

Bachelor of Science in Information Technology

Supervisor: Dr. Muhammad Asfand-e-yar

Department of Computer Science
Bahria University, Islamabad

2024

Certificate

We accept the work contained in the report titled "Search Engine for Legal documents and pleaded cases in Courts", written by Mr. Muhammad Abdullah and Mrs. Farrah as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Information Technology.

Approved by . . . :

Supervisor: Dr. Muhammad Asfand-e-yar (..)

_____

Internal Examiner: Dr. Asim Ali (..)

_____

External Examiner: Dr. Muhammad Haroon (..)

_____

Project Coordinator: Dr. Kabeer Ahmed Bhatti (..)

_____

Head of the Department: Dr. M Khurram Ehsan (..)

_____

January 31st, 2024

Abstract

The increasing complexity of legal research and the demand for quick, efficient legal services, particularly in Intellectual Property Rights (IPR), require innovative solutions. This project proposes the design of an Artificial Intelligence (AI) and Natural Language Processing (NLP)-assisted specialized Legal Search Engine for the effective retrieval and classification of legal information, particularly Trademark and Copyright pleaded cases and ordinances. This initiative enhances and redefines legal research by incorporating deep learning for text classification, ontology models for relational mapping, and a user-friendly interface for accessibility. Using advanced models like the fine-tuned PAK-LEGAL-BERT, Naive Bayes, and XGBoost, this system achieves accurate case data extraction, ordinance mapping, and judgment forecasting. The system also integrates Flask for RESTful API development and React.js for an intuitive user interface. Additionally, an AI-powered chatbot, built using Naive Bayes and the Gemini API, enhances user engagement by delivering instant support and comprehensive case summaries.

Acknowledgments

"In the name of Allah, the Most Gracious and the Most Merciful". Alhamdulillah, all commendations to Allah for the qualities and His approval in finishing this undertaking. We are very grateful to our family who gave us the strength, courage and supported us financially and morally throughout the educational carrier. We are humbly thankful to our supervisor Dr. Arif Ur Rahman who made his efforts with us in making this project throughout the final year project. He puts his additional knowledge and efforts for our help and was always there for our guidance.

MUHAMMAD ABDULLAH AND FARRAH
Islamabad, Pakistan

June 2024

"Unity is strength... when there is teamwork and collaboration, wonderful things can be achieved."

Mattie Stepanek

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

GUI	Graphical User Interface
UC	Use Case
UI	User Interface
UML	Unified Modeling Language
AI	Artificial Intelligence
ANN	Artificial Neural Network
LSE	Legal Search Engine
DB	Database
API	Application Programming Interface
NLP	Natural Language Processing

Chapter 1

Introduction

In an age where legal research is becoming increasingly complex and time-consuming, the demand for efficient, accessible tools to navigate vast amounts of legal information has become a necessity. The project aims to conceptualize, design, and implement a novel AI-driven Legal Search Engine for practitioners and individuals dealing with legal matters. The platform proposes intelligent text classification, ordinance mapping, judgment forecasting, and other advanced features to tackle the fragmentation of legal research by focusing on Intellectual Property Rights (IPR), including trademark and copyright cases, using powerful Natural Language Processing (NLP) and deep learning algorithms. Users are provided with an interface that intuitively guides them through a comprehensive dataset of legal documents categorized for ease of access and accuracy. The system incorporates the use of advanced technologies, including PAK-LEGAL-BERT, for the extraction and categorization of legal text; Flask, which provides a robust API backend; and React.js for creating an intuitive frontend experience. An AI chatbot using Naive Bayes and the Gemini API provides instant support and case summaries, thereby redefining the way legal resources are accessed and used. Specific and actionable, the information provided by this platform empowers users to make informed decisions, thereby fast-tracking the process of legal research and setting new benchmarks in leveraging legal resources.

1.1 Problem Description

The web, by its very nature, is full of abundant information on legal matters, and much of it is scattered. Although many legal documents such as pleaded cases and ordinances, and judgments are available, they are not centralized well for any user to get relevant information easily. Most of them come in formats such as PDFs or HTML, which makes access impossible to specific sections or even details about cases without manual work, as it is the case with the documents regarding IPR issues such as Copyright and Trademark

laws.[1]

Practitioners and individuals seeking legal information face several challenges:

- Difficulty in accessing and searching integrated documents with such subjects like Copyright or Trademark cases and ordinances as information is usually not well organized.
- Inability to locate judgments related to particular articles or cases with ease, as case texts and judgments are not grouped according to the corresponding article numbers.
- Access to pleaded similar cases is limited, making it difficult to collect historical background or precedent without lengthy manual search.

These issues need an AI-driven application that can search, categorize, and interlink intelligently to enhance the research feature of the legal information. It would make access to integrated case data, judgment details, and other related legal documents easier; thus, it would help professionals and general users take more informed decisions in this complex legal world.

1.2 Project Objectives

Efficient and effective access to legal information is a crucial challenge in modern legal practice, particularly in the field of IPR in Pakistan. The core objective of this project is to design and deploy an AI-driven Legal Search Engine with the aim to ease legal research processes. The platform limits itself to only Trademark and Copyright cases, uses Natural Language Processing, and deep learning for intelligent similar case and ordinances search, ordinance search based on section number, judgment classification with a user-friendly interface for easy legal resource navigation. Advanced features will include the integration of tools such as PAK-LEGAL-BERT and an AI chatbot to revolutionize the accessibility of legal searches.

The objectives of the proposed system are as follows:

- Allow users, including practitioners and individuals, to search relevant IPR's legal resources effortlessly through an intuitive platform
- Empower users with AI-driven tools like judgment classification to classify judgement results.
- Enhance user engagement and support with an AI chatbot offering instant assistance and case summaries.

1.3 Project Scope

The idea of the project is to design, develop, and deploy an AI-powered legal search engine customized for IPR cases, particularly for Trademark and Copyright matters, with intelligent text classification, ordinance mapping, and judgment forecasting to enable users to efficiently search and retrieve relevant legal information of pleaded cases or ordinances. The project will include a user-friendly interface that will enable users to easily navigate through categorized legal documents for specific and accurate research. An AI chatbot will also be included to give summaries of cases instantly and provide support to users in order to enhance accessibility.

The back-end will be done using Flask to provide RESTful API services, while the front-end will make use of React.js for an intuitive and engaging user experience. The system will leverage advanced NLP models like PAK-LEGAL-BERT and machine learning algorithms to ensure accurate extraction and classification of legal information. While the project emphasizes technical reliability and efficiency, it will not provide legal advice or interpretations. This comprehensive platform aims to redefine legal research, offering practitioners and individuals a faster and more efficient way to access and utilize legal resources.

Chapter 2

Literature Review

Legal research has taken a big turn with the advent of advanced technologies, including AI, in making tools more efficient for legal professionals. There are several systems targeted at different needs in legal research, each with strengths and weaknesses. This review seeks to explore the main features and limitations of leading legal research systems, focusing on how they use AI in streamlining legal search processes.

2.1 Existing systems

The legal research landscape has been evolving with the advent of technology and AI, aiming to make legal resources more accessible and effective. This review examines the performance of existing models, integration of technologies, user experience, specialized domain focus, and beyond judgment prediction. It also highlights the current limitations of traditional platforms, providing a foundation for the development of an advanced Legal Search Engine.

Previous studies, such as Chalkidis et al. [2], have demonstrated the effectiveness of models like HIER-BERT in analyzing legal texts and predicting outcomes. These models excel at capturing the hierarchical structures and complex relationships inherent in legal documents, which is essential for making accurate predictions in cases involving multiple layers of legal references. Research has further highlighted the strong performance of machine learning models such as KD-BERT, SVM, and bi-LSTM, which have been applied to tasks like legal document classification, judgment prediction, and legal topic identification, achieving high accuracy rates [3, 4].

2.1.1 LexisNexis

LexisNexis is one of the most-used legal research platforms that provide access to a wide array of legal resources, including case law, statutes, regulations, and legal news. Among its key features is the Lex Machina, which predicts case outcomes based on historical data.

By using machine learning models, It Machina analyses millions of case data points for insights that range from win probabilities, judicial tendencies, and what the impacts of legal arguments are. The platform also hosts Lexis Analytics, which help in the identification of patterns and trends in legal data.[5]

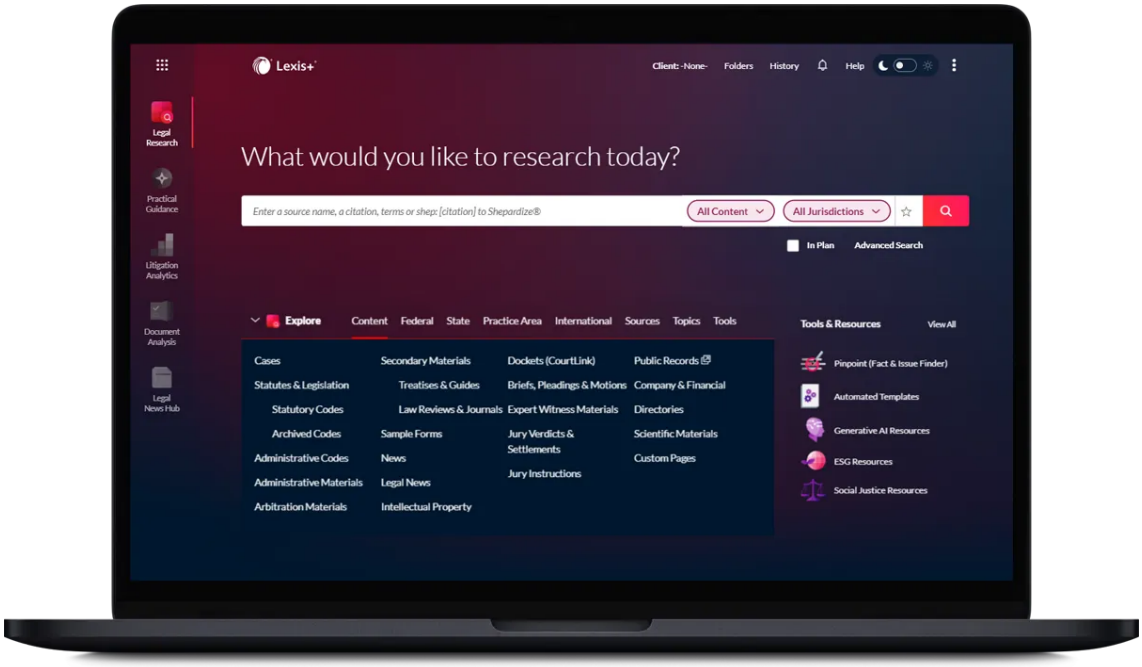

Figure 2.1: Lexis+ AI WebApp UI

2.1.2 Westlaw

Westlaw is another industry standard legal research tool that integrates AI and machine learning to streamline legal research and analyses. Westlaw has reported that their KeyCite system can predict case validity at an accuracy rate of 88%, greatly enhancing legal research by reducing the time spent in verifying the relevance and validity of cases. Besides KeyCite, Westlaw provides WestSearch Plus, an AI-driven search engine that understands natural language queries and returns highly relevant case results.[6]

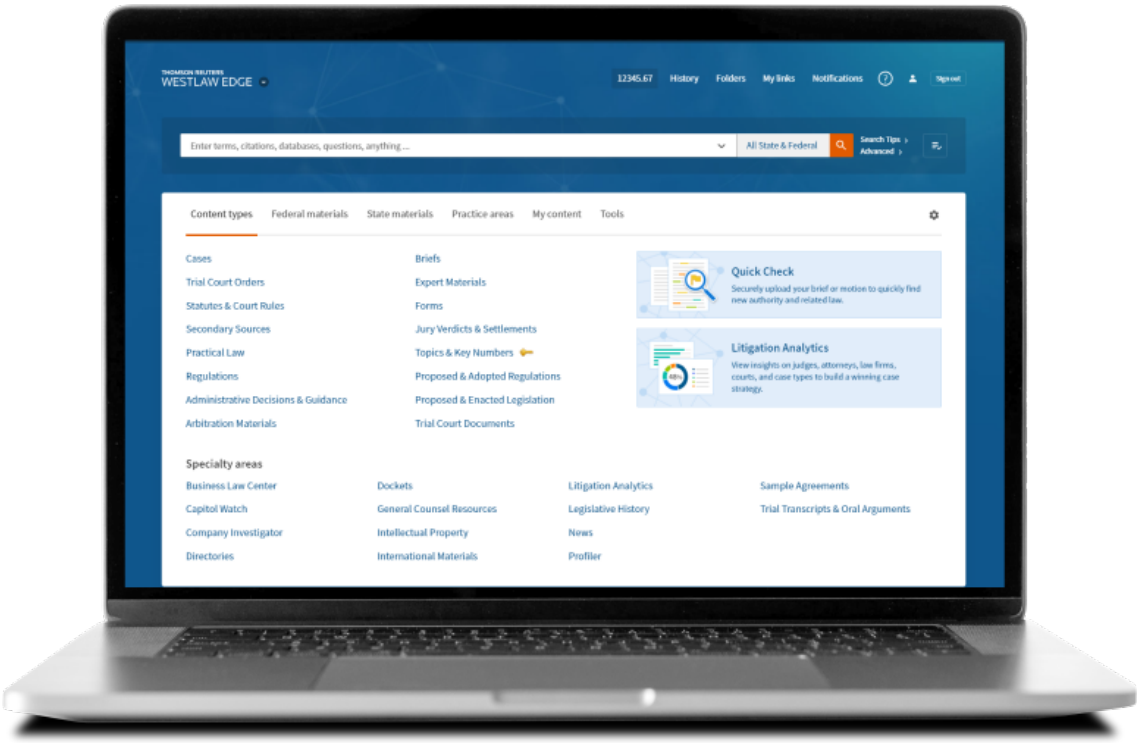

Figure 2.2: WestLaw Edge WebApp UI

2.1.3 Casetext

Casetext is a legal research platform that stands out for its innovative use of AI, particularly through its CoCounsel tool. CoCounsel, underpinned by two of the most technologically sophisticated AI models, GPT-3.5 and BERT, enhances legal research with its highly relevant case law results and AI-powered summaries. Another important feature is the use of Parallel Search to run related case law, statute, and precedent searches side by side for increased efficiency. Among the key differentiators setting Casetext apart, in particular from the more antiquated LexisNexis and Westlaw services, is a user-centric interface and AI-driven recommendations.[7]

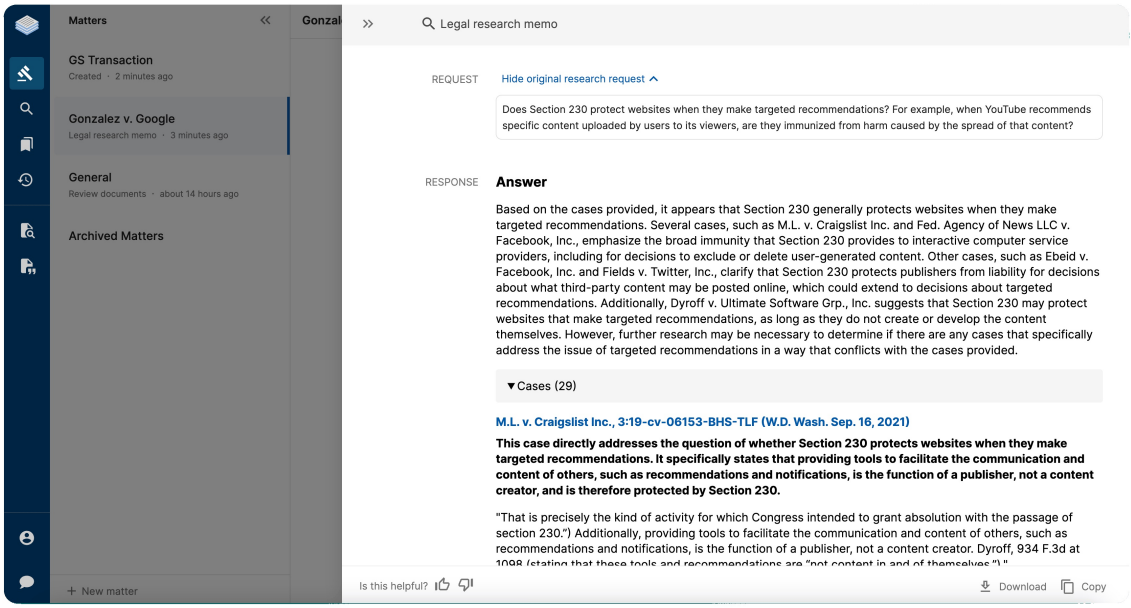

Figure 2.3: CaseText WebApp UI

2.2 Limitations

LexisNexis, Westlaw, and Casetext have limitations in user interface complexity and high pricing. LexisNexis and Westlaw offer customizable dashboards but are often overwhelming for users, with subscription costs ranging from $323 to $863 per month. Casetext is more affordable but still has a basic interface. My project differs by focusing on a simplified, user-friendly experience at a lower cost, making legal research more accessible and efficient for all users. The goal is to provide a cost-effective solution that reduces the complexity of legal research, making it accessible to a broader audience.

Table 2.1: Comparative Analysis of Legal Research Platforms

Aspect	**LexisNexis**	**Westlaw**	**Casetext**
Content	Case law, statutes, regulations, legal news, journals, treatises.	Extensive legal content across jurisdictions.	Judicial opinions and appellate court decisions.
Search	Advanced search options with filters.	Allows searches by jurisdiction, date, court, topic.	Database-specific legal case searches.
UI	Intuitive, customizable dashboards.	Customizable features and personalized alerts.	Simplified navigation for users.
Cost	$323/month (Essential); $863/month (Premium).	Pricing varies depending on the package.	Lower-cost subscriptions available.

Chapter 3

Software Requirements Specification

Software Requirements Specification is a very key factor in software projects, described in a way that it describes the interfaces, performance, reliability, and quality assurance of the application. The document explains in detail both functional and non-functional requirements. This step of requirements specification thus provides the backbone for the project.

3.1 Application Overview

This application would be oriented to meal services, making nutritional and personalized meals tailored for health awareness among elderly patients or patients due to specific health issues. Currently, food delivery services cannot offer options which fulfill the demand of a diet tailored for a person's needs; thus, users often face difficulty in finding meals according to their needs. It is here that our application fills the gap by connecting users with a network of well-trained home chefs who prepare customized meal plans, keeping in mind the individual health needs of each user.

3.2 Proposed System

The proposed system for the Legal Search will be a web-based application with Progressive Web Application(PWA) for mobile , structured into two key Modules:

- **Search Engine Module**: This feature allows users to search for legal cases similar to their queries. It operates like a standard search engine, where users can enter case descriptions or keywords to retrieve related cases. The system includes 4 models:
  - Search Model trained on generic cases dataset.
  - Search model trained on trademark cases dataset and can search by section No
  - Judgement Result Classification using SVM model.

- **Chatbot Module**: This component offers an alternative to the search engine. The chatbot first classifies the user's query to determine whether the general case model or the trademark case model should be used using Multinomial Naive Bayes text classification. Once the appropriate model is selected, the similar cases are searched and summarized using Gemini API, giving users a clear, concise overview of key points in the relevant cases.

Together, these modules create a robust tool for legal research, offering accurate case retrieval, smart query classification, and easy-to-understand case summaries.

3.3 Functional Requirements

Functional requirements discuss what the system must do or what its purpose is.They explain the features and core behaviour of the system. Functional requirements of our proposed system are:

- Users should be able to register by providing necessary details such as name, email, and password, and securely log in to their accounts after Registration to access and save history.
- User Should be able to search cases using three different Search Model.
  - Generic Cases dataset Search Model.
  - Trademark Cases dataset Search Model by text or by section number.
  - Judgement Classification Model.
- User's Search Engine's Search History should be saved automatically and user should be able to manage it.
- User should be able to use Chatbot for Cases search in summary format.

3.4 Non-Functional Requirements

Non-functional requirements discuss how the system performs its functions and the quality attributes it must have.Following are the non functional requirements of our proposed system:

3.4.1 Security

User's login,registration, profile information and history must be must be secured. Passwords saved in databased must be hashed for added security in cases database is breached. Use Cross-origin Resousrce sharing to only allow valid source to access Backend and Search Models API.

3.4.2 Usability

The system must provide a seamless and intuitive user experience across whole application, including user registration, login, case search, and chatbot interaction. Registration and login interfaces should be user-friendly responsive and validation exception should be displayed to users for their easy. Application should be easy to navigate, specially search Engine so user do not get confused in selecting Search Model. Web applications should have an interactive design for mobile devices too.

3.4.3 Flexibility

The system should be designed with flexibility in mind, allowing for modifications in the dataset, finetune of Models. The frontend, backend, and API tier should be decoupled, ensuring that changes in one layers, such as updating the dataset or fine tuning or modifying the search models, do not break the functionality of the others. This flexible design will allow for easier updates, integration of new features, and future scalability without significant problems.

3.4.4 Performance

The application should be responsive, able to provide responsive real-time search and quick responses to user interactions. Search models should be efficient in searching.

3.4.5 Scalability

The application should be scalable and be able to handle a rising number of users without compromising performance or reliability.

3.4.6 Compatibility

The application should be compatible with a wide range of web browsers. Support different screen sizes, resolutions, and hardware configurations.

3.4.7 Availability

The application should be available and maintain high availability with minimal downtime for maintenance or updates. It should Implement backup and recovery mechanisms to handle any failures, ensuring the application is operational 24/7.

3.4.8 Accuracy

The search engine must provide accurate and relevant results for all types of searches (generic cases, trademarks, judgments), ensuring users can find the information they need quickly and reliably.

3.5 System Requirements

The software and hardware requirements outline the necessary components, including operating systems, programming languages, libraries, and any specialized hardware, to ensure the successful implementation and deployment of the system.

3.5.1 Software Requirements

The following software tools and technologies have been selected for development of our system:

- **VsCode** is a hybrid IDE and code editor, popular due to its vast library of extensions, compatibility and clean UI.
- **ReactJs**: A front-end java-script framework is used for front-end development due to its component re-usability,popularity and state management capabilities.
- **NodeJs**: A back-end java-script framework is used for handling server-side logic and handling authentication,database and search models API.
- **Flask**: A micro web framework built on python, is used for development of RESTful API of search models.
- **MongoDB** A NoSQL database chosen for managing managing authentication and search engine's history. We opted for MongoDB because of the simplicity of data and cloud features of Mongo Atlas.

3.5.2 Hardware Requirments

The following hardware requirements are required to support the development and deployment of our system.

- **Personal Computer** for development,model training, testing and documentation.
- **Reliable Cloud Server or Home Machine** for Deployment of the Web Application's Search API, Backend and FrontEnd.
- **Mobile Device** for testing of WebApp Interacting Design.

3.6 Use Case Diagram

The use case diagram portrays the main functionalities of the legal search engine and describes how users will interact with the system. Key processes include user registration and login, searching through legal documents, saving history, and using the chat for query purposes. It captures the main system actions and interactions between the system and the user.[8]

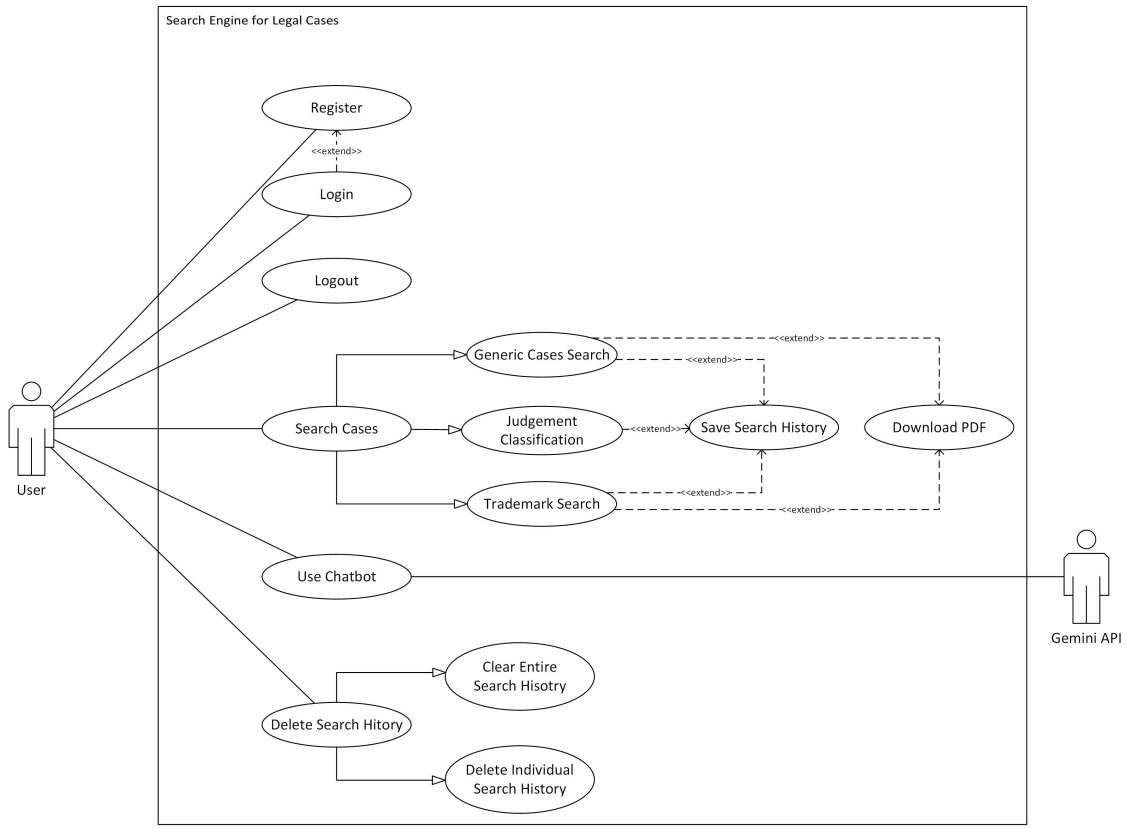

Figure 3.1: Usecase Diagram

3.6.1 Account Registration

The table "Account Registration Use Case" explains how a user creates a new account on the legal search engine. After putting the credentials it starts validating information, the system saves the details and sends back a token to keep user logged in in the browser. It also covers potential issues, like using an invalid email or a weak password.

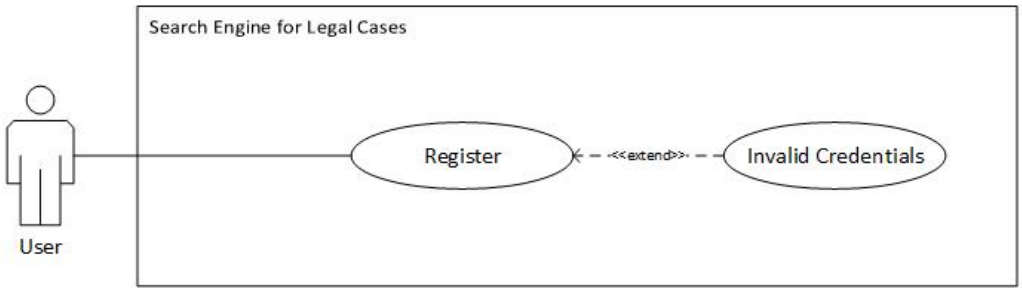

Figure 3.2: Account Registration Use Case Diagram

Table 3.1: Account Registration Use Case

Use Case ID	**LSE-Auth-1**
Name	**Register New Account**
Actor	User
Description	The User registers a new account in the legal search engine system if they want the Search Engine History to be saved.
Precondition	User is not already logged in and not registered.
Basic Flow	1. The User navigates to Signup page 2. The user fills and submit registration form
Post conditions	1. User Side drawer must be appeared in navbar 2. History of user must appear if exists.
Exception	1. Email is invalid or already registered. 2. Password is weak.

3.6.2 Account Login

The use case description "Account Login Use Case" demonstrate how a user can log in to their account on the web Application. The user navigates to the login page, enters email and password, and submits the form. The system validates the details, and if everything is correct, it creates a token for access. It also returns possible exceptions, such as if the email is not registered or the password is incorrect.

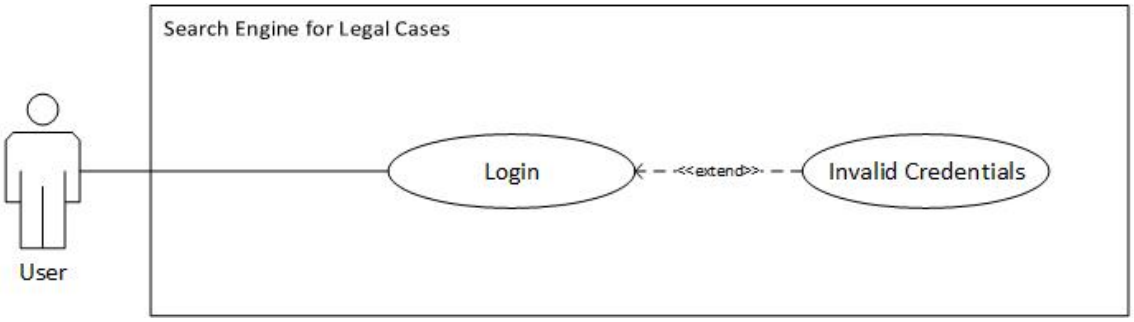

Figure 3.3: Account Login Use Case Diagram

Table 3.2: Account Login Use Case

Use Case ID	**LSE-Auth-2**
Name	**Login to Existing Account**
Actor	User
Description	The user login from his already registered account using email and password.
Precondition	User is not logged in and email is Registered.
Basic Flow	1. User navigates to login page. 2. Submits Credentials.
Exception	1. Email is not registered. 2. Password is wrong.

3.6.3 Account Logout

The above use case description comprises of the description of the Logout use case. To log out the account, the user opens the side drawer and clicks on the logout option. The system then deletes the JWT token from local storage, and automatically navigates them back to the home page.

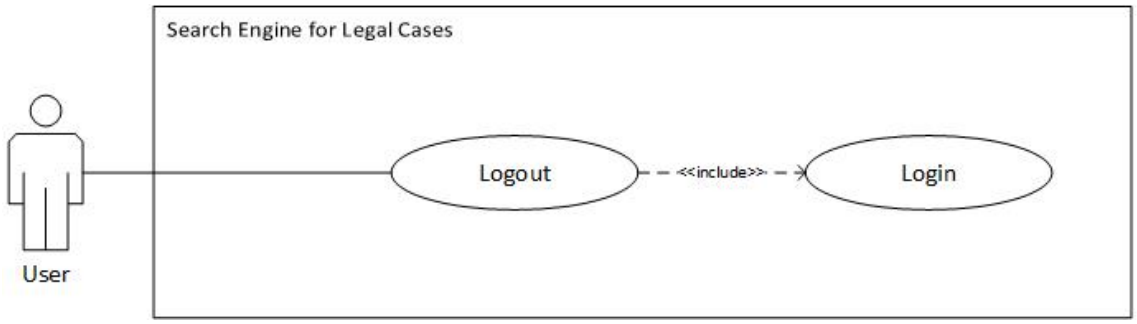

Figure 3.4: Account Logout Use Case Diagram

Table 3.3: Account Logout Use Case

Use Case ID	**LSE-Auth-3**
Name	**Logout Account**
Actor	User
Description	The user logs out from his logged in account.
Precondition	User must be logged in.
Basic Flow	• User open side drawer. • Click on Logout. • Auto navigates to Home page.
Exception	None

3.6.4 Search Cases

The use case description comprises of "Search Cases Use Case" the flow of how user will interactive with the search engine. The user navigates to the search engine page. They can then choose the type of search they want to perform, enter relevant details, and press the search button. If there are no matching cases or if the search API fails to respond, the user will see appropriate error messages.

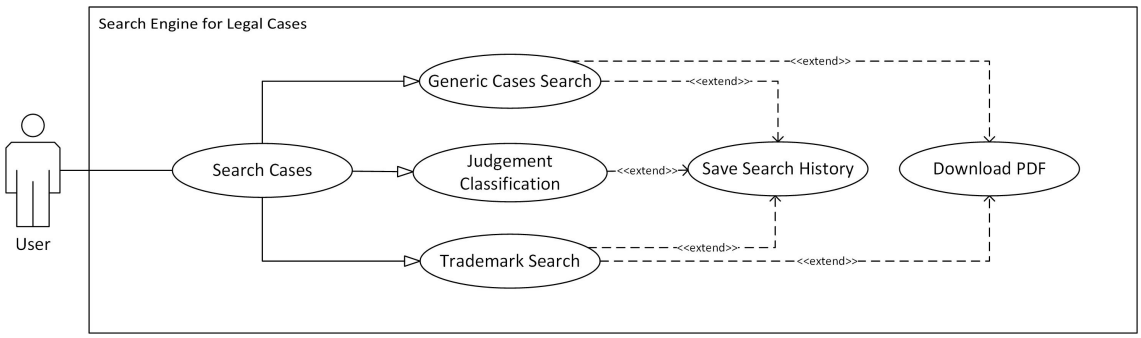

Figure 3.5: Search Cases Use Case Diagram

Table 3.4: Search Cases Use Case

Use Case ID	**LSE-Search**
Name	**Use of Search Engine**
Actor	User
Description	The User searches for cases using different search models
Precondition	None
Basic Flow	The User navigates to the Search Engine Page.
Generic Search Flow	From the Radio Button, select "Generic Search".Input relevant information in the search input and hit Search.
Trademark Search Flow	select "Trademark Search". Select Search by Text or Section No. using the select input. Input relevant information in the search input and hit Search.
Judgment Classification Flow	From the Radio Button, select "Judgment Classification". Input relevant information in the search input and hit Search.
Exception	No Case found to be similar or No Response from Search API.

3.6.5 Save Search History

The "Save Search History Use Case" demonstrate how a user's search activity is automatically saved in the system. When a logged-in user performs a search, their token is sent with the request. If the token is valid, the system saves the search history in the database. If the token is not valid, the process will not proceed, making sure that only authorized users have their search history saved.

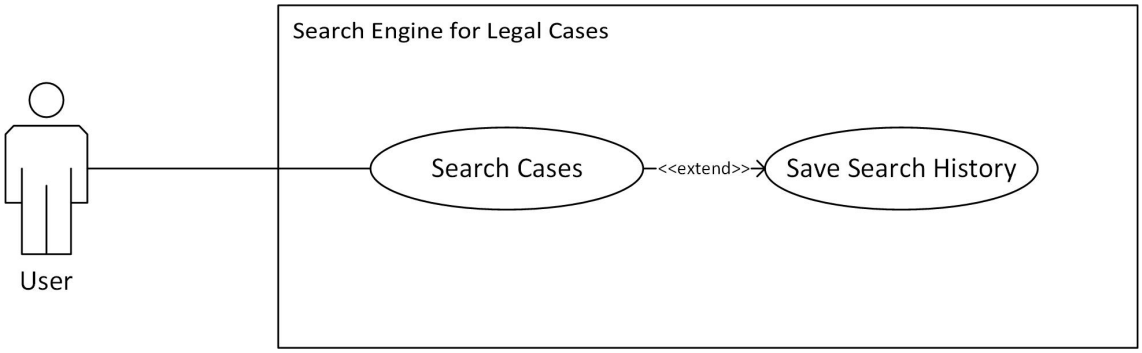

Figure 3.6: Save History Use Case Diagram

Table 3.5: Save Search History Use Case

Use Case ID	**LSE-History-1**
Name	Save Search History
Actor	User
Description	The User searches using Search Engine and History gets saved Automatically.
Precondition	User is Logged in
Basic Flow	The User Searches using Search Engine.
Exception	Token is not valid.

3.6.6 Delete Search History

The "Delete Search History" use case explains how a user completely clears its search activity or deletes an individual search history. When a logged-in user wants to clear its search history, they navigate to the side drawer and click on the "Clear History" button. When a user wants to delete a specific search history, they navigate to side drawer, find the search they want to remove front history, and click on delete icon of that search.

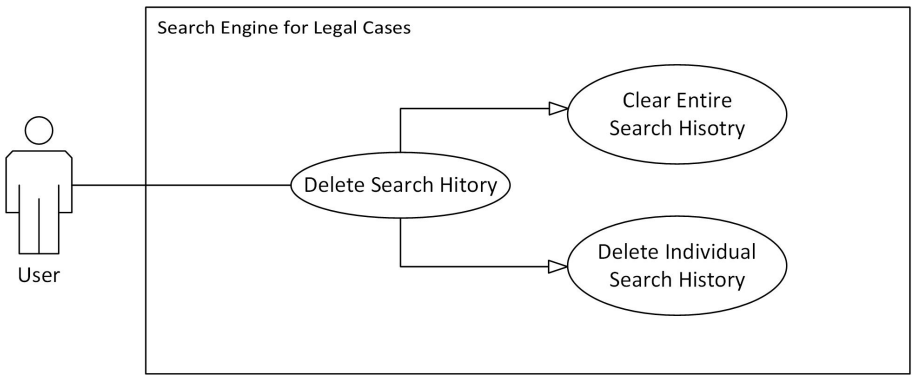

Figure 3.7: Delete History Use Case Diagram

Table 3.6: Delete Search History Use Case

Use Case ID	**LSE-History-2**
Name	**Delete Search History**
Actor	User
Description	The user deletes the specific history from the history list or clear the entire history.
Precondition	User is logged in and history is not empty
Basic Flow	The user open history and delete a specific history.
Alternate Flow	The user open history and clear the entire history.
Post Conditions	History is removed and is not visible to the user.
Exception	Token is not valid or the history ID is not valid.

3.6.7 Use Chatbot

The "Search Using Chat-Bot" Use Case explains how a user can search for case summaries using a chat-bot interface. The user opens the chat bot by clicking its button and enters the type of search along with specific input, such as "Search Section 13 for trademark search." After clicking the search button, the chat bot processes the request. If there are issues, like the Gemini API not responding or incorrect text classification, the user will face exceptions that disturbs the search process.

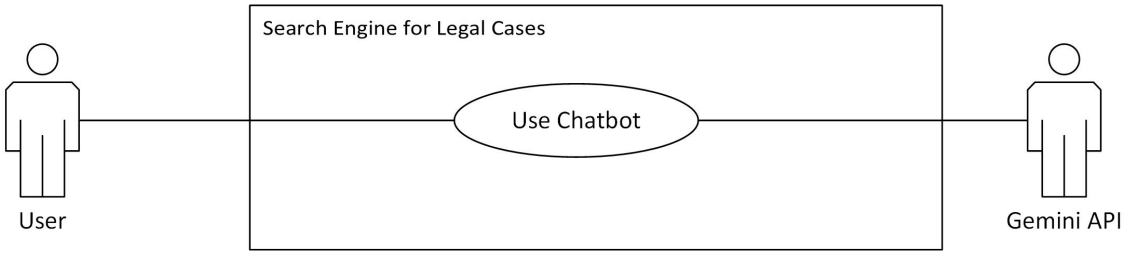

Figure 3.8: Chatbot Use Case Diagram

Table 3.7: Search using Chatbot Use Case

Use Case ID	**LSE-Chatbot-1**
Name	**Search using Chatbot**
Actor	User
Description	The user search for Case summary using chat-bot.
Precondition	None
Basic Flow	User Open chat-bot and inputs the Query.
Post Condition	Chat bot response is shown in interface.
Exception	Gemini API is not responding

3.6.8 Generate Result PDF

The "Generate Result PDF" Use Case explain how a user can download a case's details table PDF from search result but clicking on a "Save PDF" button.

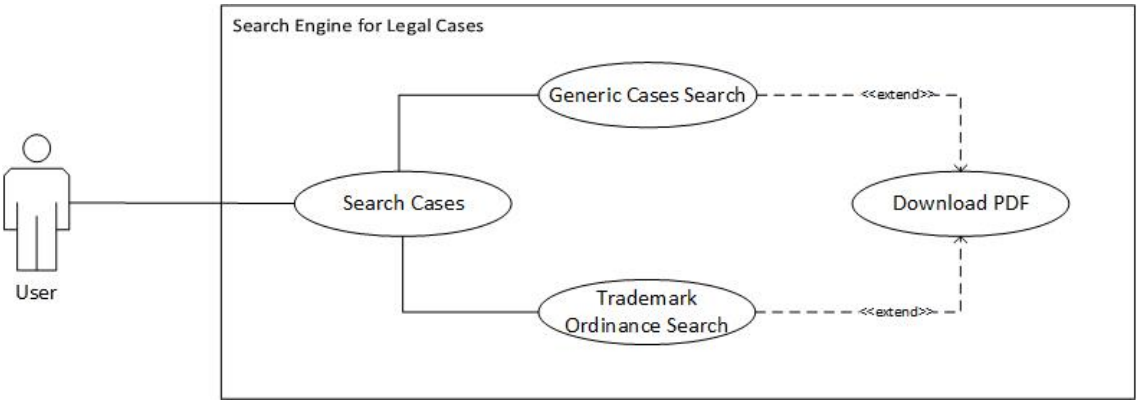

Figure 3.9: Generate PDF Use Case Diagram

Table 3.8: Generate Result PDF

Use Case ID	**LSE-Search-PDF**
Name	**Generate PDF Result**
Actor	User
Description	The User Download a case's details table PDF of Specific Search Result.
Precondition	Results are in the state
Basic Flow	User Click on Save PDF button along with each result.
Post Condition	File is Downloaded in the devices.
Exception	None

Chapter 4

Design

This section of the publication includes UML diagrams to help you understand the system. The diagrams concisely illustrate roles, actors, actions, and class perspectives.

4.1 System Architecture

Our legal search engine is a web-based application with a Progressive Web App for mobile devices. It can be explain using three-tier architecture framework , where models is divided into three separate layers, making the system easier to manage, maintain, and scale. The modification occurred in user interface, meaning changes done in presentation layer wont effect and wont require changes in application or data layer.

4.2 Workflow Diagram

The project workflow diagram visually represents the step-by-step process of our legal search engine development, illustrating how tasks are interconnected and flow from one stage to another.

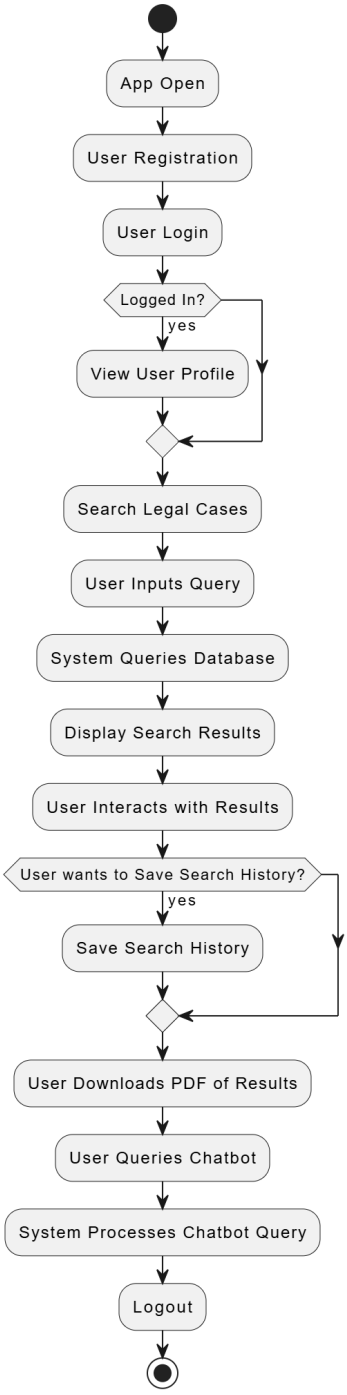

Figure 4.1: Workflow Diagram

4.3 Activity Diagram

This activity diagram gives an overview of the user interactions with the system. The user initiates actions such as entering a search query, logging in, registration,logging out, and reviewing search history, which are handled by the web browser interface. Based on the user's actions, the system interacts with the back-end and database to process queries, retrieve legal information, and display search results to the user. Additionally, the system requests Gemini API for summarization.[9]

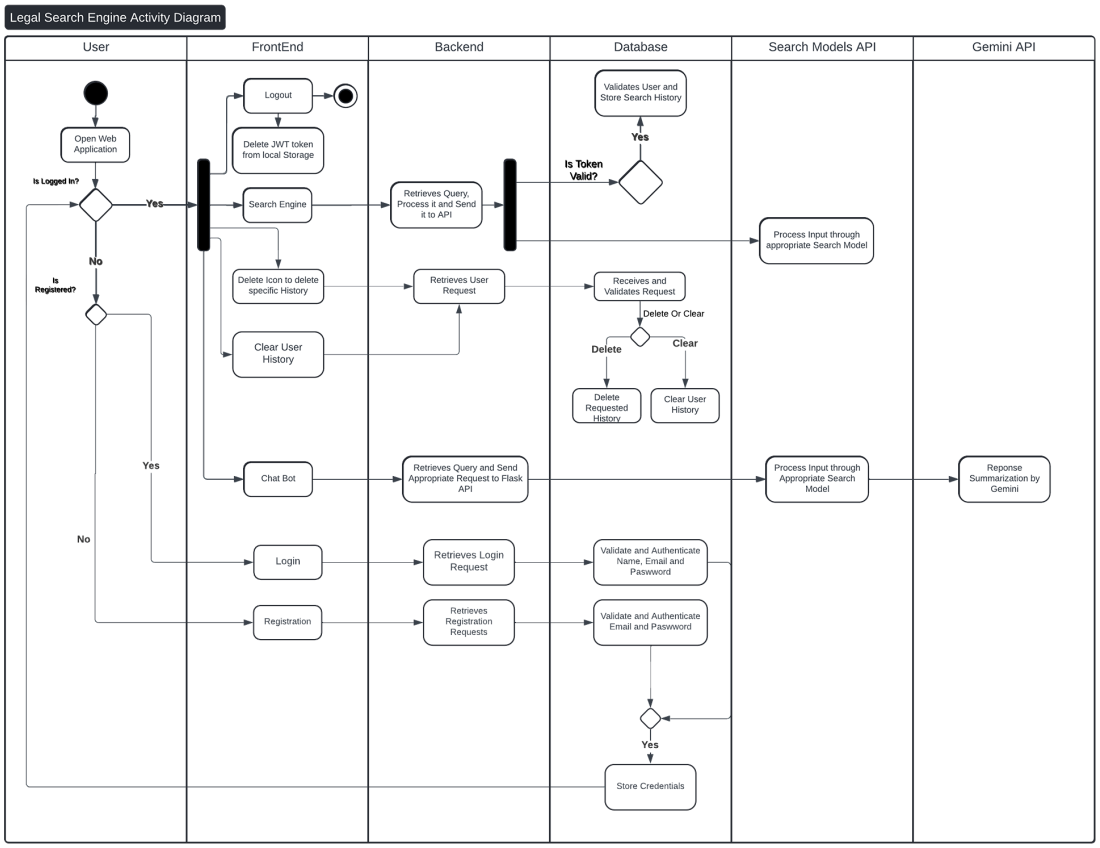

Figure 4.2: User activity diagram

4.3.1 System Architecture Diagram

A system architecture diagram visualize the structure and interactions within a system, illustrating its components,flow, and connections. The user interface features Authentication module, a legal search engine, and a chatbot, all interacting with the backend via authentication and search routes. The backend connects to a database for user authentication, JWT token creation, and history tracking. The Search Models API includes multiple search models (e.g., general search, trademark search, judgment classification, and Naive Bayes classification). The Gemini API further improves response of chatbot.

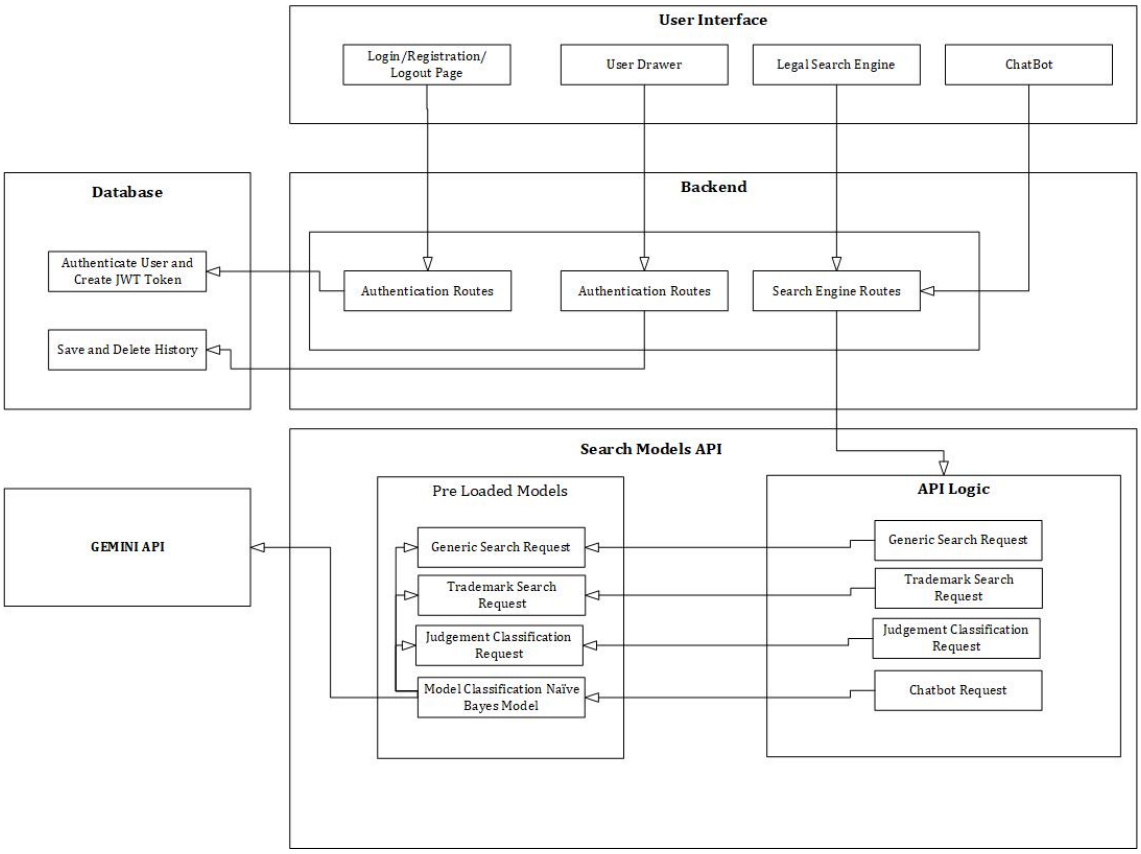

Figure 4.3: System Architecture Diagram

4.3.2 Presentation Tier

The presentation tier is the frontend of the system that provide interface to the user for interaction with the system and to access its functionalities. Frontend is developed using React, HTML and tailwind CSS.

4.3.3 Application Tier

The Application tier is the backend of the system that handles business logic layer. It processes routes and requests done by user through backend, allow presentation tier to communicate with database, provide authentication logic and manages overall functionalities of the system. NodeJs and ExpressJs is user to manage database,user request, query validation and authentication. Flash web framework is used to develop Search models API for models to be accessible through API routes.

4.3.4 Data Tier

Our Database layers involves storing and managing user's data, storing authentication token, manage validation and search history . Our Database contains two table, one for user's information like id,name,email and hashed password. Second Object contains search history id, user's id, search query details and timestamp.

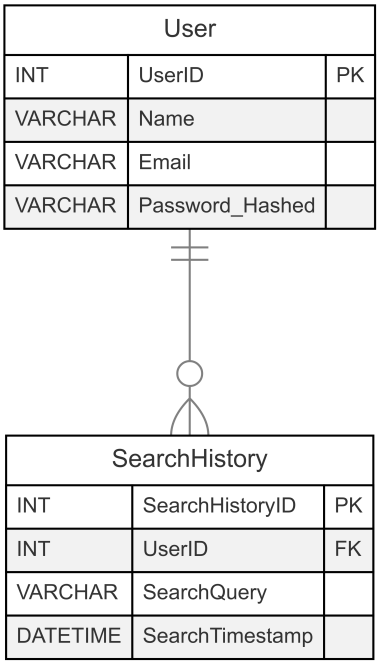

Figure 4.4: ERD Diagram

4.3.5 Design Methodology

The project follows the Agile Development Methodology to accommodate iterative development due to not having AI models before hand, agile was the most suited Methodology for our project . This approach allowed us to develop Interface and API for each Search model one by one in iterative approach with supervisor's feedback along it.

4.3.6 Design Constraints

4.3.6.1 Resource Limitations

The Web application is hosted on a cloud server with limited computational resources, affecting the performance of the AI models. embedding were computed before hand and saved in pkt file to eliminate the encoding time.

4.3.6.2 Data Constraints

The system relies on the availability of well-structured and accurate legal documents Dataset. It is important to maintain a comprehensive and up-to-date database of legal documents to ensure accurate results.

4.3.6.3 Security Constraints

Users data and search history must be stored securely. This constraint made it necessary to store hashed password instead of raw value.

4.4 Sequence Diagrams

4.4.1 Registration Sequence Diagram

The registration sequence diagram illustrates the interaction between the user and the system during the user registration process.

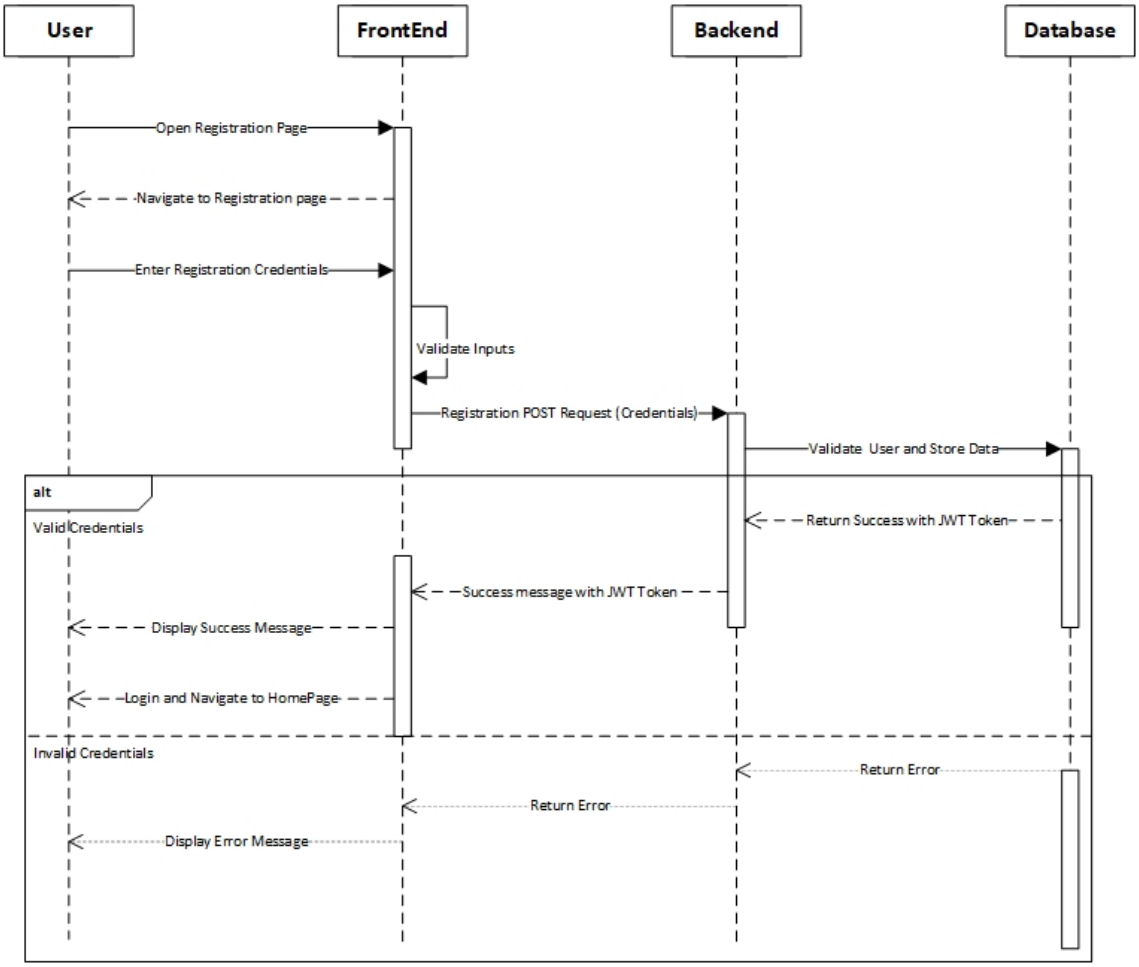

Figure 4.5: Registration Sequence Diagram

4.4.2 Login Sequence Diagram

The login sequence diagram outlines the process of user authentication within the system, beginning with the user accessing the login page and entering their credentials. Once the user submits their username and password, the login page communicates with the database to verify the provided information. Depending on the validation result, the user is either granted access to the application or prompted to retry, ensuring a secure and efficient login experience.

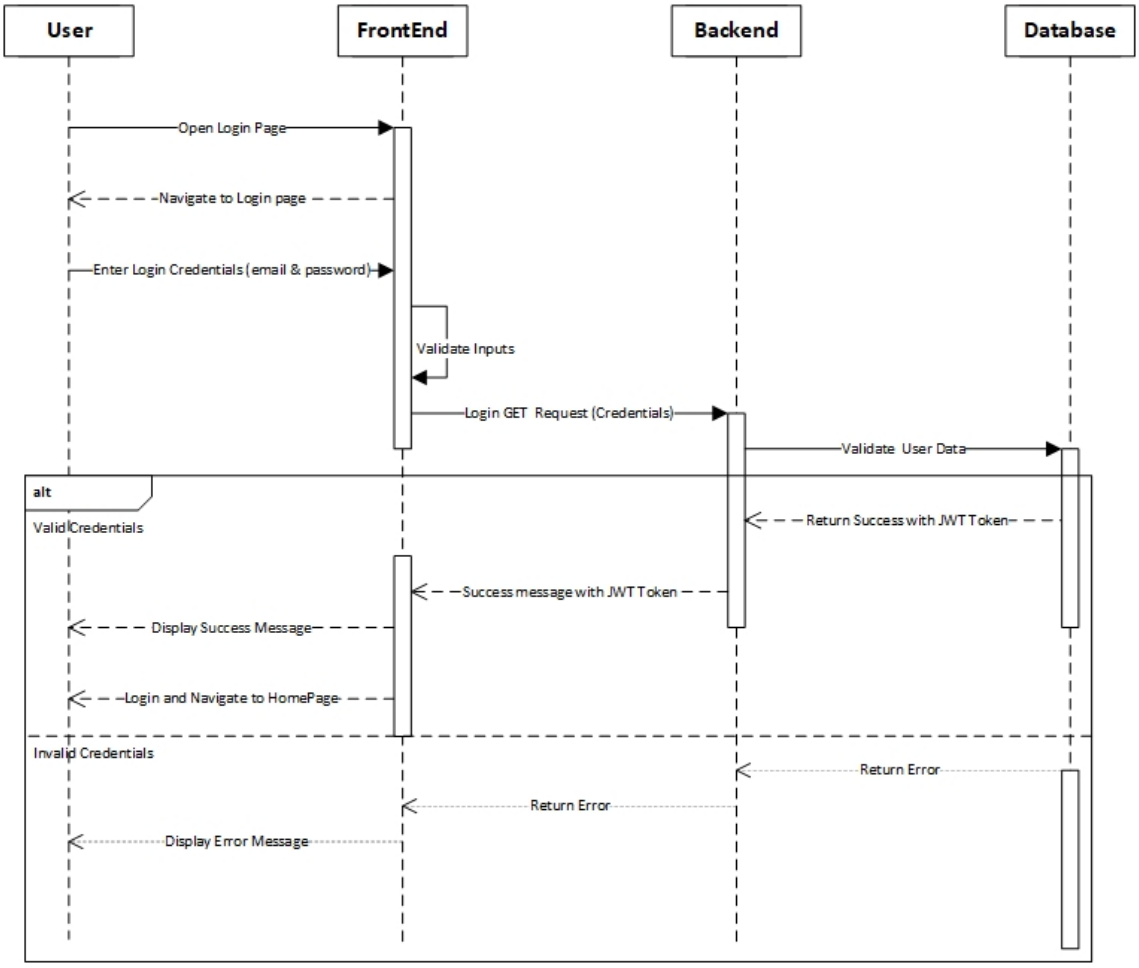

Figure 4.6: Login Sequence Diagram

4.4.3 Search Engine Usage Sequence Diagram

The search engine usage sequence diagram illustrates the interaction between the user and the search engine during a typical search operation. It begins with the user entering a query into the search bar and submitting it, which triggers a request to the search engine's backend. The system processes the query by retrieving relevant data from its indexed database and then returns the search results to the user, providing a seamless and efficient experience.

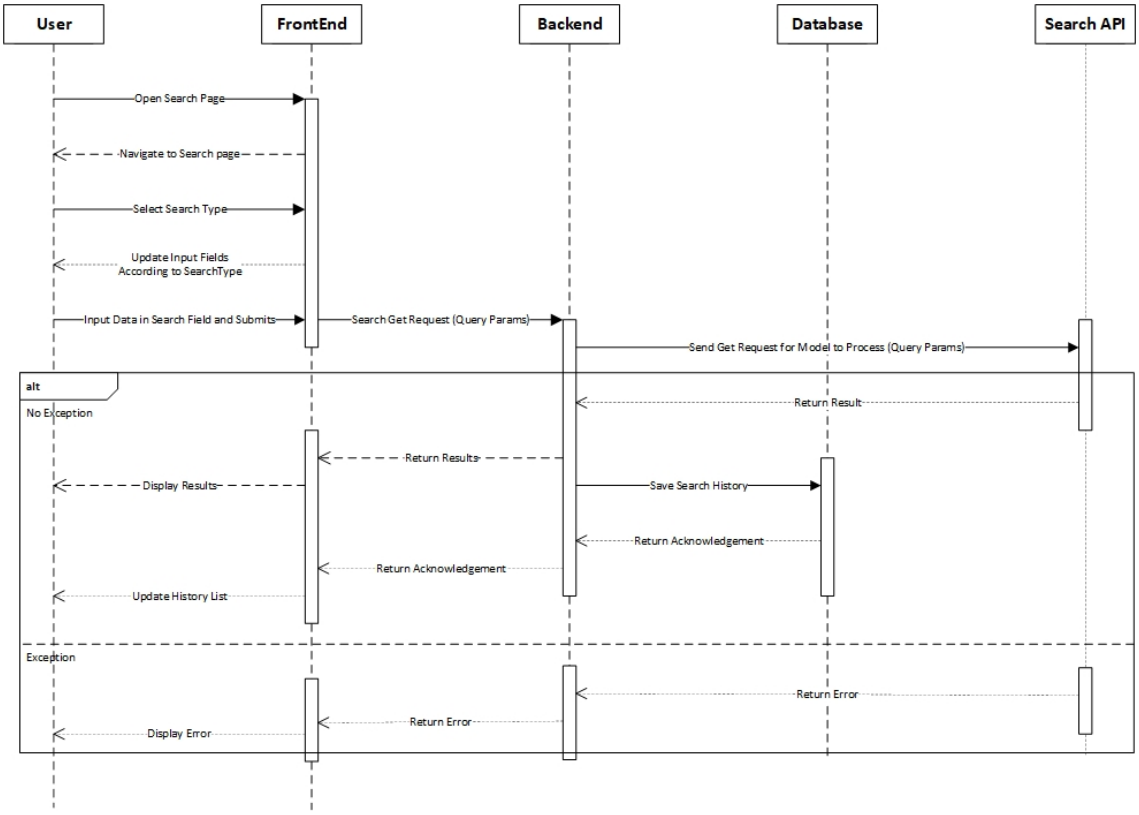

Figure 4.7: Search Engine Usage Sequence Diagram

4.4.4 Chatbot Usage Sequence Diagram

This diagram outlines the interaction process between the user and the chatbot during a conversation. It begins with the user sending a message to the chatbot, which then processes the input and generates an appropriate response.

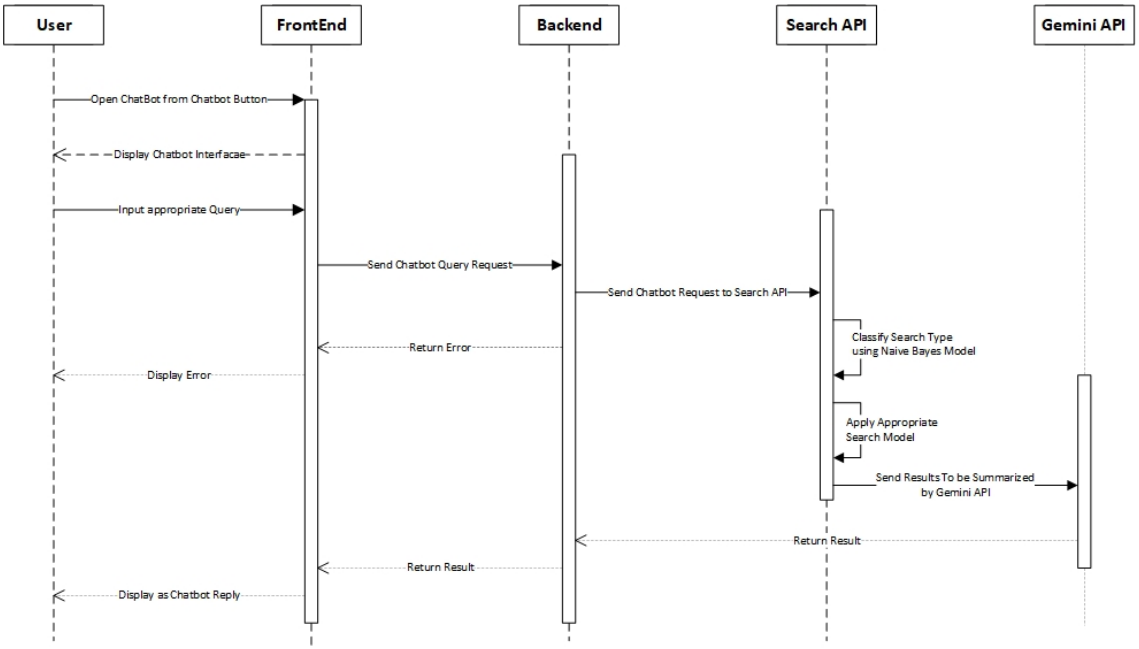

Figure 4.8: Chatbot Sequence Diagram

4.4.5 Access Search History Sequence Diagram

The access search history sequence diagram demonstrates the steps a user takes to retrieve their previous search queries from the system. It starts with the user requesting to view their search history, prompting the system to authenticate the user's identity and access the relevant data stored in the database. Once the system retrieves the search records, it displays them to the user, ensuring a smooth and informative experience while maintaining data security and privacy.

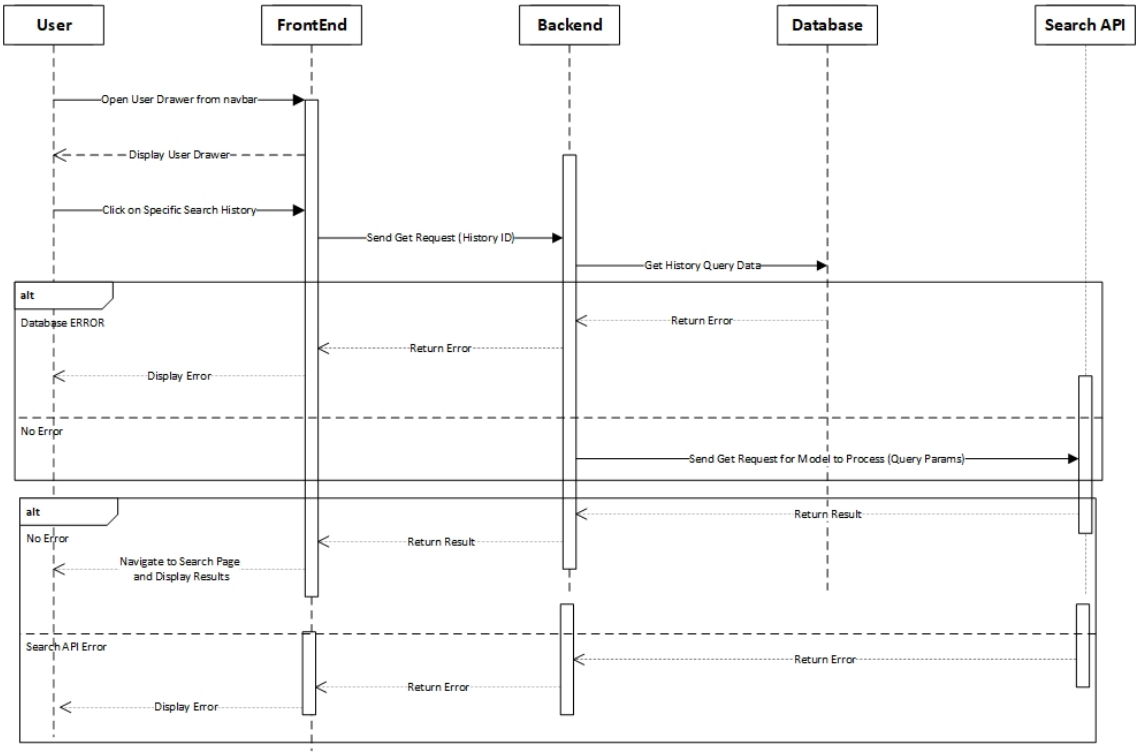

Figure 4.9: Access search history sequence diagram

4.4.6 Delete History Sequence Diagram

The delete search history sequence diagram outlines the process of a user removing their past search records from the system. It begins with the user initiating a request to delete their search history, which triggers the system to authenticate the user's identity and then proceed to permanently remove the specified search data from the database, ensuring the user's privacy and control over their personal information.

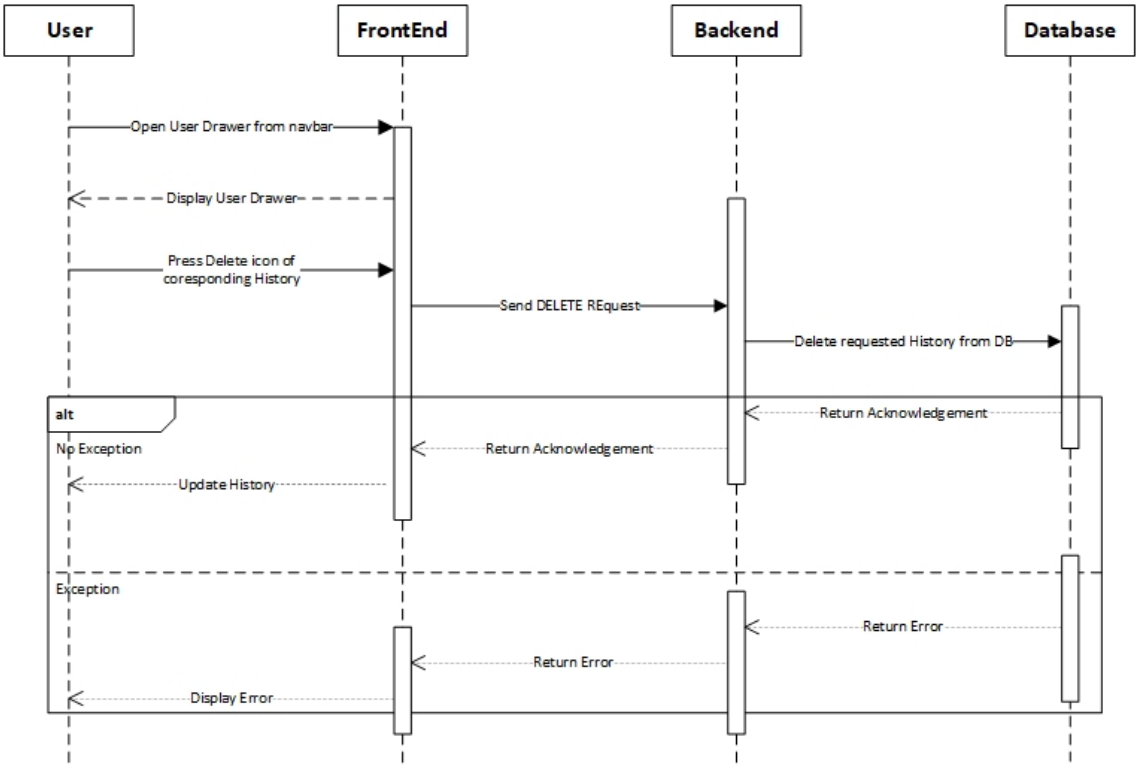

Figure 4.10: Delete History Sequence Diagram

4.4.7 Clear History Sequence Diagram

The clear search history sequence diagram illustrates the process a user follows to remove all their previous search records from the system.

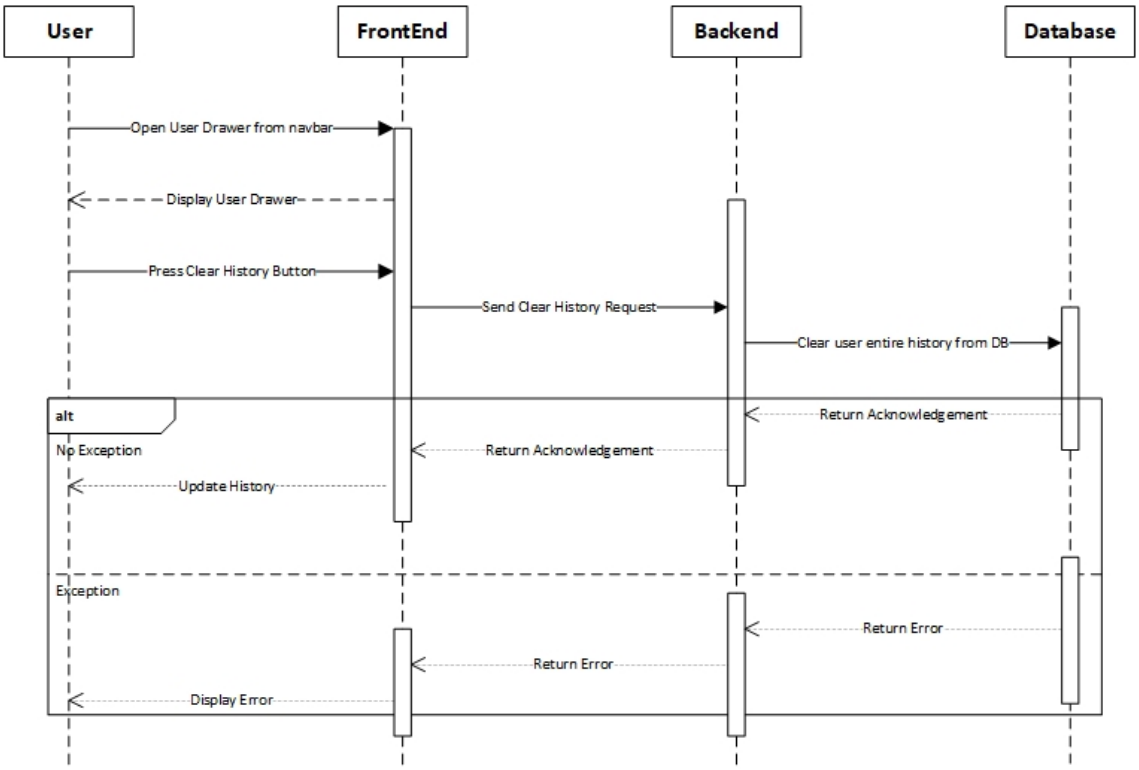

Figure 4.11: Clear History Sequence Diagram

4.4.8 Logout Sequence Diagram

The logout sequence diagram depicts the process initiated when a user requests to log out of the application. It begins with the user clicking the logout button, which triggers a request to the system to terminate the active session.

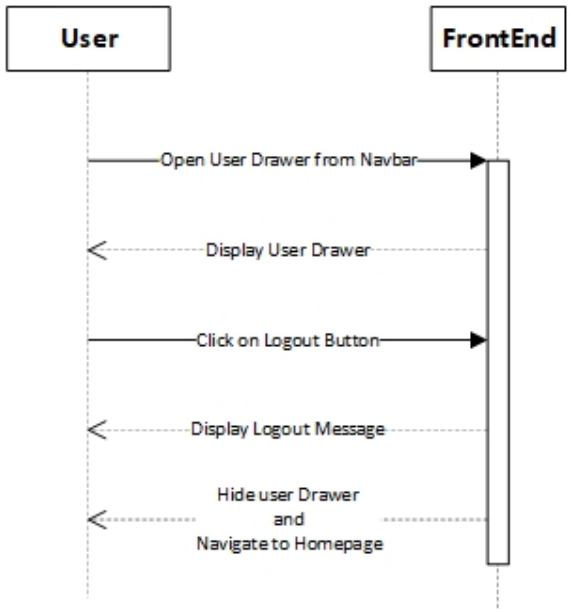

Figure 4.12: Logout Sequence Diagram

4.5 GUI Design

GUI (Graphical User Interface) design is the process of creating the visual components and interactions that users will encounter when interacting with a software application or digital product. The goal of GUI design is to provide an intuitive, efficient, and visually appealing interface that enhances the user experience.

4.5.1 Home Page UI

The home page UI is designed to engage users with user-friendly design that leaves a lasting impression.

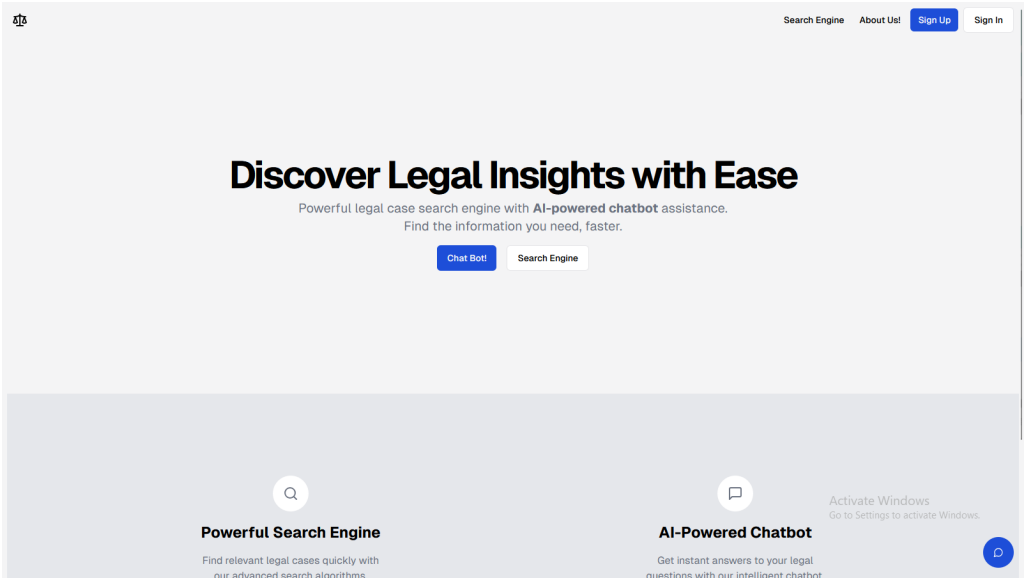

Figure 4.13: Home Page UI

4.5.2 Registration Form UI

The registration form UI is designed to be visually appealing and user-friendly, guiding users through the sign-up process seamlessly.

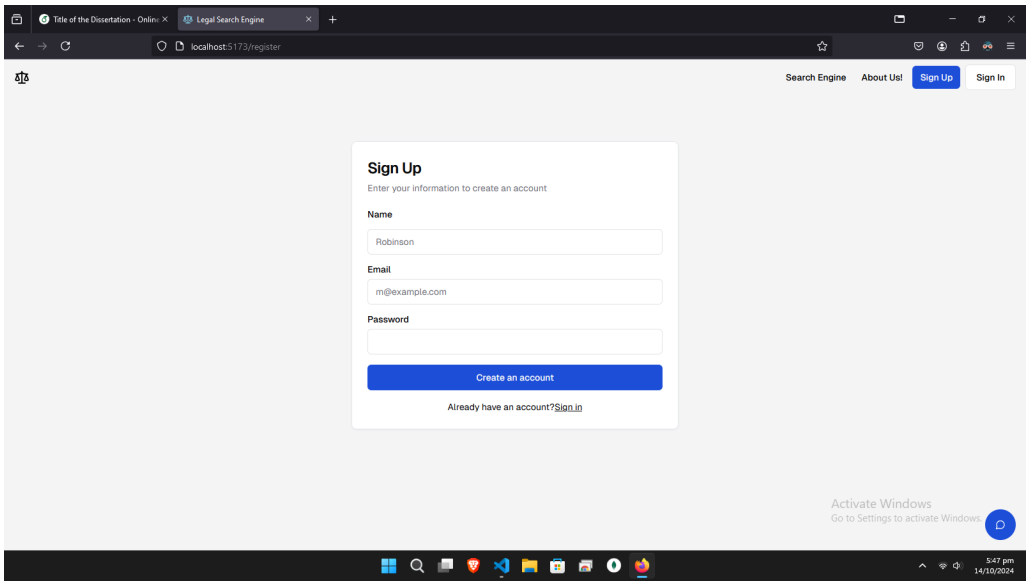

Figure 4.14: Registration Form UI

4.5.3 Login Page UI

The login page UI prioritizes ease of use, with a clean and simple design focused on the core functionality of secure account access.

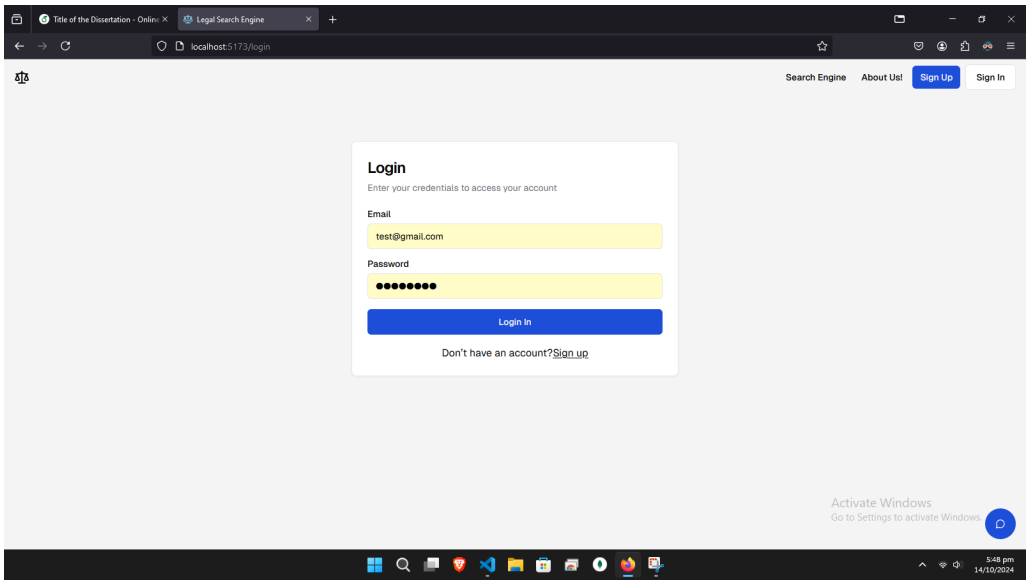

Figure 4.15: Login Page UI

4.5.4 Search Engine Page UI

The search engine page UI provides a visually engaging and efficient search experience, with a prominent search bar and clear presentation of results.

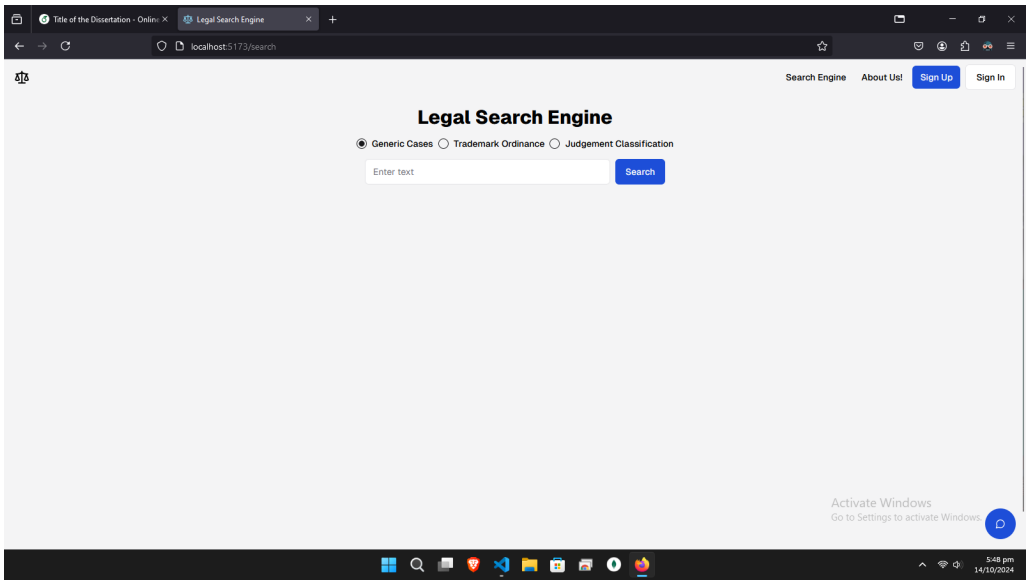

Figure 4.16: Search Engine Page UI

4.5.5 Generic Search Results UI

The generic search results UI is designed for readability and scanability, making it easy for users to quickly identify the most relevant information.

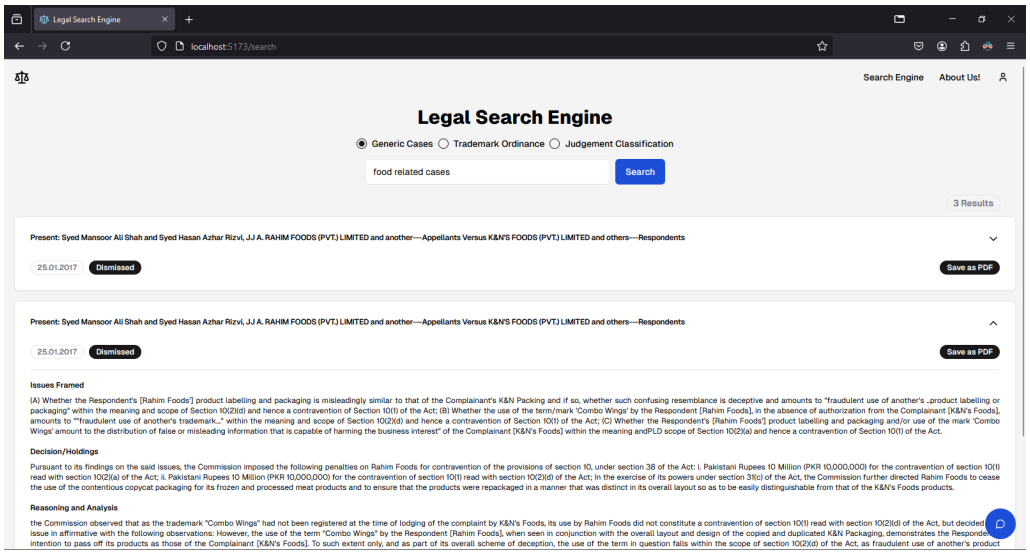

Figure 4.17: Generic Search Results UI

4.5.6 Trademark Search Results UI

The trademark search results UI is tailored to the specific needs of users searching for trademark information, with a visually appealing and intuitive layout.

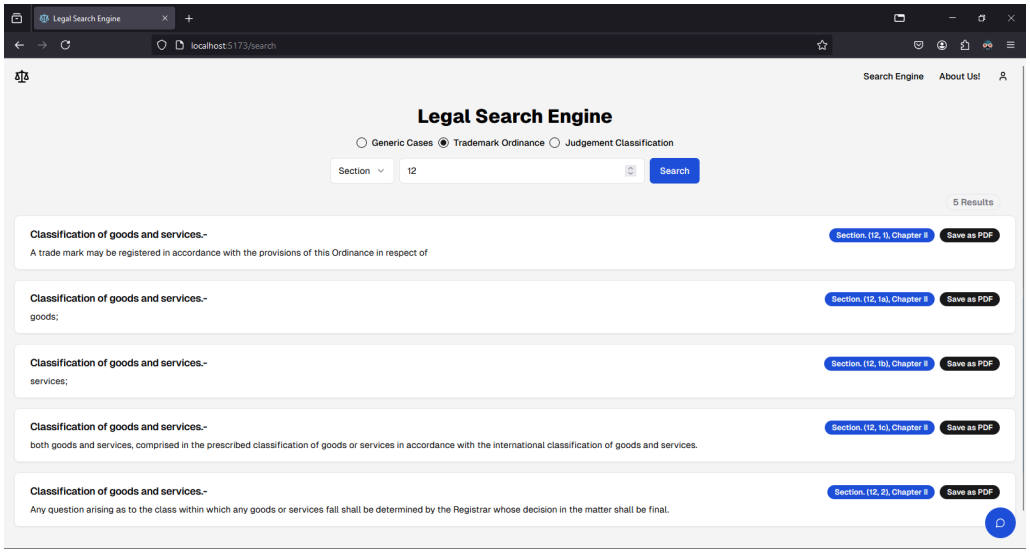

Figure 4.18: Trademark Search Results UI

4.5.7 User Profile Drawer UI

The user profile drawer UI maintains a consistent look and feel with the overall brand identity, providing a seamless and personalized experience for managing account information.

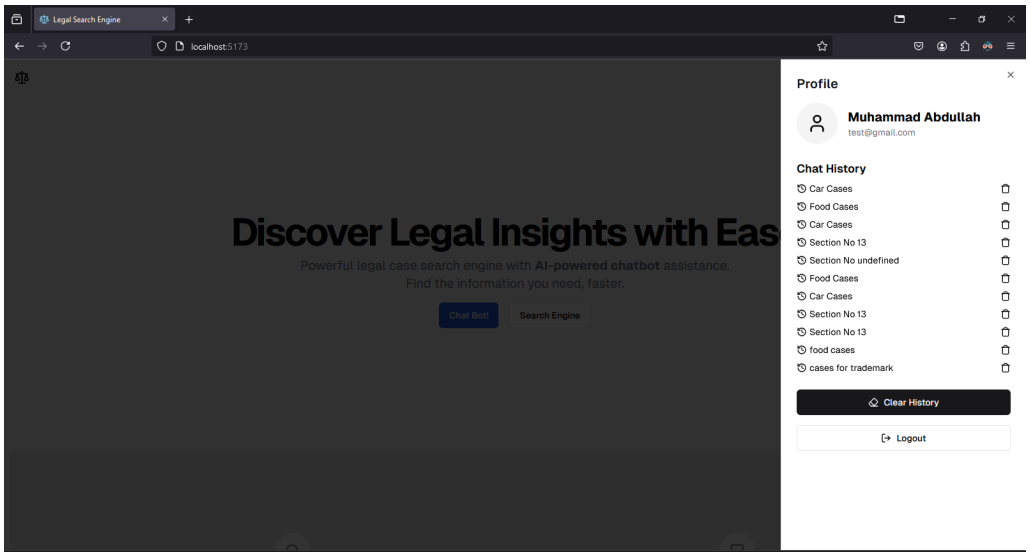

Figure 4.19: User Profile Drawer UI

4.5.8 Chatbot Component UI

The chatbot component UI is designed to be visually engaging and intuitive.

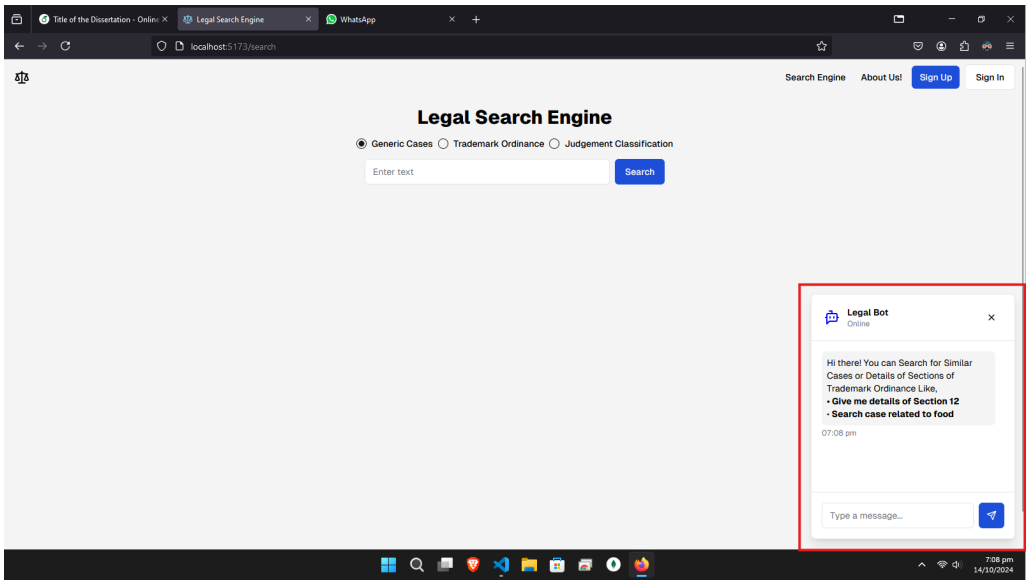

Figure 4.20: Chatbot Component UI

4.5.9 Search API Documentation UI

The search API documentation UI strikes a balance between visual appeal and informative content, making it easy for developers to understand and integrate the search API.

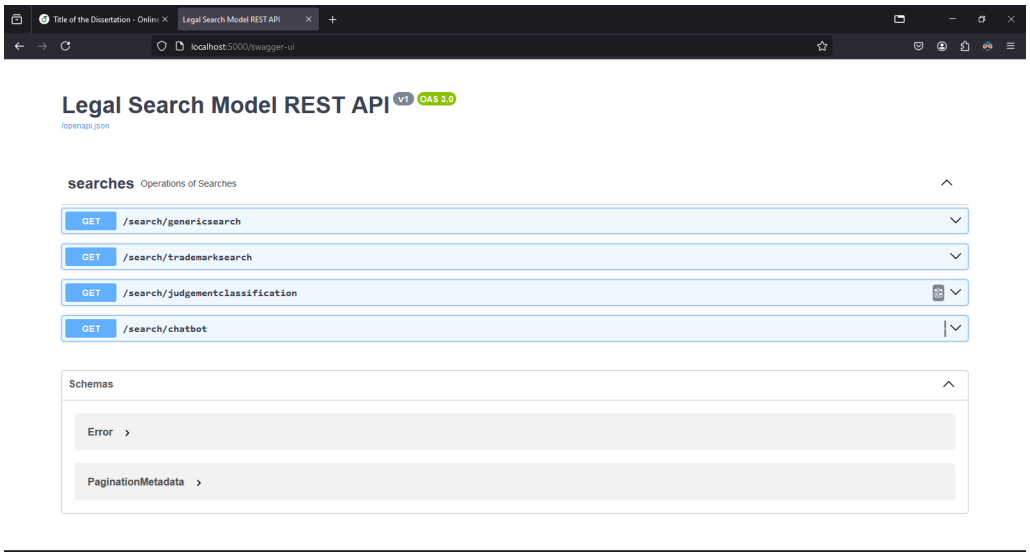

Figure 4.21: Search API Documentation UI

4.6 External Interfaces

External interfaces enable seamless communication between a system and third-party applications or services. They ensure smooth data exchange and integration, enhancing overall functionality.

4.6.1 External API

Google LLM model API "Gemini" is used for chatbots response functionality, Model is used to Summarize the array of result data in to simple chat response structure.

4.7 Physical View

Legal Search engine's all three components frontend, backend and search API is currently deployed on local host setup but its is designed to support cloud hosting for future deployment, which will increase its availability to public and improve performance.

Chapter 5

System Implementation

This chapter provides a brief description of the implementation strategies utilized to create this application. Implementation refers to the process of turning ideas, projects, and plans into action. Implementing a system is the actual execution of a technical specification or algorithm as a program, software component, or other computer system via programming and development. Additionally, this chapter contains information about the tools and procedures utilized to implement our web application.

5.1 Tools and Technologies

The following are tools and technologies being used in our system.

5.1.1 ReactJs Framework

The search engine's frontend is developed on the highly popular React.js framework that is widely used because of its component-based architecture, which supports reusability and modularity. Reactjs offers a fast and responsive user interface where load times are minimal. It is ideal for building modern web applications.

It takes help from several libraries:

- **Tailwind CSS**: A utility-first CSS framework that allows for quick styling with customizable predefined classes.
- **Zod**: A schema declaration and validation library for validating forms to ensure proper user input.
- **React Router Dom**: Manages the navigation of the application, facilitating smooth transitions between pages such as landing, login, registration, and search.

- **React Hot Toast**: A library for displaying error and success notifications throughout the application using React's Context Hook.
- **Axios**: A promise-based HTTP client for making requests to the backend, working seamlessly in both Node.js and React environments.
- **jsPDF**: A library for generating PDF files, used to create case detail reports on the client side.

5.1.2 NodeJs framework

The Node.js Backend of the application is an open-source JavaScript runtime that allows developers to run scalable server-side Web applications.

In handling requests and building the API, the following are some of the key libraries that the system uses:

- **Express** is a web application framework built on top of Nodejs for designing and building web applications quickly and easily.
- **Bcrypt** is a JavaScript library used for hashing passwords securely. and to decrypt password while authentication.
- **Js Validator** is a library used to validate common patterns are email validation, password's strength, data type etc. It is used for validating requesst query parameters incase someone send get request through API tool and bypass frontend validations.
- **mongoose**: Mongoose is an Object Data Modeling (ODM) library for MongoDB and Nodejs. It provides a schema-based solution to model your application data, provide simple way to define schemas, perform validation, and interact with the MongoDB database using JavaScript objects.

5.1.3 Flask framework

Flask is a lightweight web framework used to build the API to manage chatbot and Search Models. Reason of opting flask is because it is a microframework and ML models were also built on python, reducing the hassle of using child processes in Node Js to integrate Python models.

- **Flask Smorest** Flask Smorest is a library built on top of Flask to simplify the creation of REST APIs, along with the integration of Marshmallow that provides schema validation of query's request and response data.

- **Swagger UI** Swagger UI is an open-source tool that allows developers to visualize and make APIs interactive using a web-based interface. It creates a user-friendly documentation page based on the Open API specification and through schemas defined in your codebase.
- **Sklearn** Sklearn is an open-source versatile library used for machine learning and data analysis. We have used Sklearn's Multinomial Naive Bayes classifier and count vectorizer for our chatbot's text classification model.
- **Pickle** Pickle is an incredibly useful Python module for serializing and deserializing Python object structures.

5.1.4 Mongo Database

The database used for management of user authentication and search engine history is MongoDB. We chose MongoDB because of it user friendly cloud features and was suitable for handling small amount of data.

5.1.5 Gemini API

Gemini API is an end point to access google's AI generative models, used in system from summarization of search result in chatbot response.

5.1.6 Naive Bayes Model

Multinomial Naive Bayes (MNB) is a very popular and efficient machine learning algorithm that is based on Bayes' theorem. It is commonly used for text classification tasks where we need to deal with discrete data like word counts in documents. In this article, we will discuss MNB and implement it. The Legal Search Engine classification by the MN Naive Bayes model categorizes these into Case Search For pleaded legal case queries, Section Number For queries specifying ordinance's section no and ordinance Search For Ordinance queries.[10]

5.1.6.1 Dataset Format

The dataset forms the basis for the type of query determination whereby the system selects one out of three categories: Case Search, Ordinance Search, or Section Number. This is a very critical classification because it guides the system on which path to take in processing The dataset ensures that user queries are routed efficiently to the correct model for optimum functionality of the legal search engine.

Table 5.1: features of dataset used for training the Naive Bayes model.

Query	**Category**
Find similar cases to employment	Case Search
Find me section 34 of the law	Section Number
Search for legal cases involving intellectual property	Case Search
Show me ordinance laws regarding trademark dispute	Ordinance Search
What does the trademark ordinance say about intellectual property?	Ordinance Search
Look up what section 34 says	Section Number
Find the trademark rules related to patent	Ordinance Search
Give me information about the trademark ordinance related to industrial design	Ordinance Search

5.1.6.2 Performance Matrics

The performance matrix of the Naive Bayes model reflects its very high performance in all legal search tasks. For Case Search, precision, recall, and F1-score were all very high at 0.90, while the solid results of Section Number and Trademark Ordinance Search achieved an F1-score of 0.86. In sum, the overall accuracy for the model is 88%, with a consistency across all metrics in both the macro and weighted averages at 0.88, indicating a reliable and efficient model for legal case retrieval.

Table 5.2: Naive Bayes Performance Metrics

Category	**Precision**	**Recall**	**F1-Score**	**Support**
Case Search	0.90	0.90	0.90	71
Section Number	0.88	0.85	0.86	73
Ordinance Search	0.85	0.88	0.86	56
Accuracy	**0.88**			**200**
Macro Avg	0.88	0.88	0.88	200
Weighted Avg	0.88	0.88	0.88	200

Chapter 6

System Testing and Evaluation

The chapter for system testing and evaluation shall include all the test cases and test cases procedure of the project. The project demands software testing. Tests are conducted to assess the generated program and yield results. Software testing is carried out in a controlled environment considering all the normalities and anomalies in the system. In this project, we faced early agitations but after that received the expected results after running the tests. Otherwise all features worked as expected. The test cases we executed for this project are listed below, along with the results.

6.1 Testing Techniques

Testing methods involve procedures that need your intended program to go through numerous testing processes to establish whether the proposed functional or non-functional requirements work properly or not; this is a system defect specification before the system is published. Our system has been tested using a variety of methodologies, which are listed under the headings below. Some of them include acceptance testing, software performance testing, usability testing, load testing, compatibility testing, and security testing. Each phase has been thoroughly tested to ensure that any exceptions are handled properly.

6.2 Acceptance Testing

Acceptance testing was used to check that the Legal Search Engine fulfilled all the specified requirements, both functional and non-functional. It was targeted that it should satisfy that the system is actually ready for deployment; its user interaction, final implementation, and whether it met the project's aims. This testing helped prove that the system was, indeed, going to meet the expectations of stakeholders and could do what it was supposed to before its final release. Factors like functionality, performance, and user experience were considered in determining general acceptance.

6.3 Performance Testing

Performance testing was necessary to ensure that the Legal Search Engine is efficient, reliable, and fast in different respects. It checked the capability of the application to bear different user inputs and analyzed its load-bearing capacity. While testing, we ran various scenarios involving large queries and the simulation of concurrent users. Some of the key metrics that were monitored include response time, throughput, and error rate, which can be used to find potential bottlenecks. Results showed that the system was running optimally, consistently performing well to afford users a seamless experience, especially during periods of high demand.

6.4 Functional Testing

Functional testing aimed to verify that the Legal Search Engine performed as intended based on user expectations and system requirements. Several test cases were developed to ensure that features such as user registration, search functionality, and query handling worked properly.

6.5 Unit Testing

In unit testing, individual components of the Legal Search Engine were tested in isolation. For instance, we tested the search algorithm, user registration system, and database connections separately. This allowed us to identify and resolve any issues early in the development process, ensuring that each part of the system was reliable and without nay bug.

6.5.0.1 Integration Testing

Integration testing focused on ensuring that the system components worked cohesively. This included testing the interactions between the search engine, user authentication, and database systems. Any issues related to data flow between the components were addressed to ensure a seamless experience for users when performing searches, logging in, or signing up. Testing front-end, back-end, database and API integration testing was also done in this. Each API was tested separately using "Insomnia" an API testing tool.

6.6 Usability Testing

The main reason behind conducting usability testing is to make the Legal Search Engine more intuitive, efficient, and easy to use for its targeted users. Testing offers early insights

into potential user experience problems that developers and designers can iron out before the system's deployment. Testing finds the extent to which users are able to achieve stated tasks on the system, or whether the interface facilitates or makes such operations smooth and comfortable.

This included the intended user group comprised of legal professionals and regular users not being legal experts, on the usability of the Legal Search Engine. This will let us know how efficiently users can perform searches in the legal cases, ordinances, and sections, and provide us feedback regarding the user interface. In view of the findings, the system was made more accessible and easier to use by experienced and inexperienced users alike.

6.7 Compatibility Testing

Compatibility testing is a type of non-functional testing that ensures an application's compatibility with the hardware, operating system or system architecture required to run it. It is used to determine whether the software we developed is compatible with the devices it is intended to run on. Our application has been tested on several devices at the same time and is compatible with all browsers.

6.8 Context, Test Plan

1. **Context:**The development of a Legal Search Engine to search legal cases, ordinances, and classifications is considered within the project. Thus, accuracy, reliability, and efficiency while running the system are of prime importance.

2. **Plan:** The group set particular performance testing objectives, such as including a test for the accuracy, responsiveness, and efficiency of the system. Responding time, error rates, and resource utilization were some of the indicators adopted in measuring. A testing environment was established by the team in a way that it made conditions reflective of what happens in real life to ensure realistic outcomes from the test.

3. **Ideal:** The testing scenario required developing various test cases so that legal queries are depicted, like for example, searching the case laws, or referencing ordinances. Test data included legal documents and statutes; representing numerous real-world usage scenarios to ensure accuracy and completeness.

4. **Prepare:** The test team carried out all of the planned scenarios with the help of the data provided for preparation. There was close monitoring of performance indicators on points of bottleneck or potential failure. This forms the basis for detailed analysis

in results of the tests that may come in handy in subsequent optimization and improvements.

5. **Perform:** Test Plan Execution Execute developed test cases, record the output with proof that the system under test is in line with the accepted standards of performance. Performance Testing Ensure the system is responsive and efficient in processing legal queries. This approach is very helpful to better improve the overall reliability and performance of the Legal Search Engine.

6.8.1 Traceability matrix

A traceability matrix is a document that maps and traces user requirements with test cases, ensuring that all requirements are covered by tests. It helps in tracking the progress and completeness of a project, ensuring that no requirements are missed during the testing phase.[11]

Table 6.1: Traceability Matrix

Test Objective	Test Basis	Completion Criteria
Running the Application	The application should launch without errors when executed in a browser or as a web server.	Application launches without errors, and all functionalities are accessible within a reasonable time.
User Registration	New users should be able to register by providing a name, email, and password.	User account is created successfully, and the system displays confirmation of registration.
User Login	Registered users should log in using valid credentials (email and password).	Users are logged in successfully, and their profiles are displayed after login.
Search Engine	The system should allow users to search for legal documents and cases using various search filters.	Search results are displayed based on user inputs like text, section numbers, and classifications.
Log Out	Users should be able to log out of the system.	System successfully logs out the user and returns to the login page.
API Connection	The system should connect to the database via API to retrieve data for the search engine.	API retrieves data correctly from the database without connectivity issues.

6.9 Test Cases

6.9.1 Test of Application launch

This test case checks that the application launches correctly both from the local server and directly from a browser. The case ensures an error-free initialization of the web application

so that the user could access and start using it. It targets the verification of correct server start-up and proper opening of an application interface without any errors in loading or crashing. Successful execution of this test guarantees the accessibility of the system by users in any typical local environment.

Table 6.2: Test case of Running the Application

Test Case ID	TC-01
Description	Launching the application on a local server or browser.
Requirement	Application should launch when the web server is initiated.
Steps to be Taken	1. Start the server. 2. Access the application via the browser.
Expected Result	The application should launch without errors.
Actual Result	Application launched successfully with no visible errors.
Status	Success
Remarks	N/A

6.9.2 Test Case of Account Registration Page

This test case ensures the registration of a new user into the system by filling up the required areas, including name, email, and password. The validation will be done when all the conditions mentioned above are met, followed by generating a user account from the system. The idea is to ensure that the system processes the registration in a smooth manner by popping up the right confirmation messages and storing the user's information with efficiency.

Table 6.3: Test case of User Registration

Test Case ID	TC-02
Description	Registering a new user in the system.
Requirement	User must input name, email, and password for registration.
Steps to be Taken	1. Navigate to the registration page. 2. Fill in the registration form and submit.
Expected Result	A new user account is created, and confirmation is displayed.
Actual Result	The user account was successfully created.
Status	Success
Remarks	N/A

6.9.3 Test Case of Account Login

This test case covers the login functionality for already existing accounts. It's supposed to validate that the system correctly authenticates valid user credentials, which should be in the form of an email and password combination, while rejecting invalid ones. The

successful login should redirect to a profile or dashboard page that would actually prove that authentication and session management work as expected.

Table 6.4: Test case of User Login

Test Case ID	TC-03
Description	Logging in to an existing account.
Requirement	User credentials (email and password) must be valid.
Steps to be Taken	1. Navigate to the login page. 2. Enter valid email and password.
Expected Result	The user is successfully logged in and redirected to their profile page.
Actual Result	User was logged in successfully.
Status	Success
Remarks	N/A

6.9.4 Test Case of Search Engine

This test will validate the functionality and effectiveness of the search engine in pulling up legal cases or documents from user input. The test confirms that the system effectively handles text queries, section numbers, and any other input requirements to yield relevant and correct results.

Table 6.5: Test case of Search Engine

Test Case ID	TC-04
Description	Searching for legal cases/documents using the search engine.
Requirement	Search engine must retrieve relevant data based on input criteria.
Steps to be Taken	1. Navigate to the search page. 2. Enter text or section number and submit.
Expected Result	The system should display relevant search results based on the input.
Actual Result	Search results were displayed correctly.
Status	Success
Remarks	N/A

6.9.5 Test Case of API routes

This test ensures the API's dependability and functionality in managing requests and obtaining database data. To ensure smooth connection between the frontend, backend and search models APIs, Insomnia is used to test the API routes.

Table 6.6: Test case of API routes

Test Case ID	TC-05
Description	Testing of all routes of flask api and through postman.
Requirement	Application must connect to the database to retrieve search results.
Steps to be Taken	1. Perform a search query. 2. API retrieves data from the database.
Expected Result	Data is successfully retrieved and displayed without errors.
Actual Result	API connected to the database and retrieved the correct information.
Status	Success
Remarks	N/A

6.9.6 Test Case of Account Logout

The contribution of this test is directed to the logout functionality, verifying that the user successfully logged in will be brought back to the login page after going out, ensuring one session closes.

Table 6.7: Log Out

Test Case ID	TC-06
Description	Logging out of the application.
Requirement	User must be logged in to log out.
Steps to be Taken	1. Click on the "Log Out" button. 2. Verify that the user is logged out.
Expected Result	The user is successfully logged out, and the login page is displayed.
Actual Result	The user was logged out successfully.
Status	Success
Remarks	N/A

Chapter 7

Conclusions

The legal search engine developed in this project will represents an important milestone toward simplifying research in the legal world in these busy times. It offers users a simple, efficient means to search and retrieve legal cases and documents through an interactive interface that improves access to and reliability of critical legal information. With the combination of strong search capabilities and intuitive design, this platform enhances the experience for both the legal professional and the researcher. This is a technological bridge between complex legal data and everyday usability that empowers its users with reliable and swift access to these resources.

7.1 Future Consideration

Future developments will cover expanding the dataset to wider legal topics other than copyright and trademarks, addressing the limited availability of such cases in Pakistan. Moreover, developing chatbot by fine tuning using large language models like Gemini Pro or GPT-4 using legal datasets will significantly enhance its functionality and make it a very powerful assistant in exploring cases.

7.2 Recommendations

When planning and implementing this mobile application, platform dependencies should be removed so that it works on all platforms, not just Android-based devices. Additional recommendations should be made. User interfaces should be more interactive and use graphics to represent information. Application speed, functionality, dependability, efficiency, maintainability, portability, ease of use, content quality, load time, and, most importantly, usability should all be addressed. Furthermore, the mobile application should respond to learning objectives.

References

[1] J. Adams, "Copyright law and intellectual property rights: Issues in the digital age," *Journal of Legal Studies*, vol. 32, no. 4, pp. 102–115, 2022. Cited on p. 2.

[2] I. Chalkidis, I. Androutsopoulos, and N. Aletras, "Neural legal judgment prediction in english," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4317–4323, 2019. Cited on p. 4.

[3] J. Smith and A. Doe, "Analyzing the impact of ai on legal decision-making," *International Journal of Law and Technology*, vol. 12, no. 1, pp. 15–30, 2023. Cited on p. 4.

[4] M. Jones and R. Taylor, "Machine learning applications in legal analysis," *Journal of Law and Technology*, vol. 15, no. 2, pp. 45–60, 2021. Cited on p. 4.

[5] C. Hill, "Lexisnexis announces new capabilities for lexis ai, including rag enhancements," July 2023. Accessed: 2024-12-14. Cited on p. 5.

[6] Thomson Reuters, "Westlaw edge: Litigation analytics," 2024. Accessed: 2024-12-14. Cited on p. 5.

[7] S. Wilkins, "Gpt-4 is here: Casetext's cocounsel already brought it to legal," *Legal Tech News*, March 2023. Accessed: 2024-12-14. Cited on p. 6.

[8] Y. Waykar, "Role of use case diagram in software development," *International Journal of Management and Economics*, January 2015. Cited on p. 12.

[9] A. Bhattacharjee and R. Shyamasundar, "Activity diagrams: A formal framework to model business processes and code generation," *Journal of Object Technology*, vol. 8, pp. 189–220, January 2009. Cited on p. 23.

[10] S. Xu, Y. Li, and W. Zheng, "Bayesian multinomial naïve bayes classifier to text classification," in *Proceedings of the International Conference on Computational Intelligence and Security*, pp. 347–352, Springer, 2017. Cited on p. 42.

[11] Perforce, "Requirements traceability matrix," 2024. Accessed: 2024-12-14. Cited on p. 47.