**Contents**

[List of Figures 2](#_Toc162889447)

[Introduction 6](#_Toc162889448)

[Research Methodology 6](#_Toc162889449)

[Literature Search 6](#_Toc162889450)

[Selection Criteria 7](#_Toc162889451)

[Background Research 7](#_Toc162889452)

[History of NLP 9](#_Toc162889453)

[Fundamentals of NLP 10](#_Toc162889454)

[Syntax vs. Semantics 10](#_Toc162889455)

[Tokenization 11](#_Toc162889456)

[Part-of-Speech Tagging 12](#_Toc162889457)

[Name Entity Recognition 13](#_Toc162889458)

[Dependency Parsing 14](#_Toc162889459)

[Lemmatization and Stemming 14](#_Toc162889460)

[Lemmatization 14](#_Toc162889461)

[Stemming 15](#_Toc162889462)

[Comparative Analysis 15](#_Toc162889463)

[Machine learning in NLP 15](#_Toc162889464)

[The Role of NLP in Enhancing Access to Justice 16](#_Toc162889465)

[Identifying and Mitigating Bias 17](#_Toc162889466)

[Privacy and Security Concerns in Legal NLP Applications 17](#_Toc162889467)

[Current Trends and Future Directions of NLP in law 18](#_Toc162889468)

[Project Proposal 19](#_Toc162889469)

[Project Details: 19](#_Toc162889470)

[Project Specification: 19](#_Toc162889471)

[Project User Stories 20](#_Toc162889472)

[Traceability Matrix: 22](#_Toc162889473)

[High Level System Design: 22](#_Toc162889474)

[Requirement Analysis: 24](#_Toc162889475)

[MoSCoW Analysis 24](#_Toc162889476)

[Project System Requirement: 24](#_Toc162889477)

[Project Solution 26](#_Toc162889478)

[Assumptions, Constraints, Risks: 26](#_Toc162889479)

[Solution Description: 28](#_Toc162889480)

[Solution Delivery Approach: 28](#_Toc162889481)

[Project Management Approach: 28](#_Toc162889482)

[Design Phase: 32](#_Toc162889483)

[Wireframes: 32](#_Toc162889484)

[Database Design: 33](#_Toc162889485)

[Dataset Workflow/Design: 34](#_Toc162889486)

[LLD Design: 35](#_Toc162889487)

[Implementation: 37](#_Toc162889488)

[Database Implementation and Connection: 37](#_Toc162889489)

[Front-end Implmentation: 39](#_Toc162889490)

[Backend-Server Side: 42](#_Toc162889491)

[NLP Model Implementation: 49](#_Toc162889492)

[Evaluation: 59](#_Toc162889493)

[Unit Testing 59](#_Toc162889494)

[Designing the Unit Tests Using Pytest: 60](#_Toc162889495)

[Unit Tests Outcome: 63](#_Toc162889496)

[Unit Tests Outcome Discussion: 66](#_Toc162889497)

[Integration Testing Plan: 68](#_Toc162889498)

[Integration Testing Outcomes: 69](#_Toc162889499)

[Evaluation: 70](#_Toc162889500)

[Future Work: 70](#_Toc162889501)

[Conclusion: 70](#_Toc162889502)

[References 70](#_Toc162889503)

# List of Figures

[Figure 1 Showing the Hierarchy of NLP 8](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855417)

[Figure 2 Showing the Development of NLP overtime 10](#_Toc162855418)

[Figure 3 Showing the Overlap of NLP with ML and DL 15](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855419)

[Figure 4 Showing the NLP Pipeline 16](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855420)

[Figure 5 Showing User Registration Use Case 20](#_Toc162855421)

[Figure 6 Showing User Login Use Case 21](#_Toc162855422)

[Figure 7 Showing User Legal Advice Use Case 21](#_Toc162855423)

[Figure 8 Showing User Chat History Review Use Case 21](#_Toc162855424)

[Figure 9 Showing User Providing Feedback Use Case 21](#_Toc162855425)

[Figure 10 Showing User Logout Use Case 22](#_Toc162855426)

[Figure 11Showing General Overview of the Application 22](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855427)

[Figure 12 Showing HLD Design of Litigat8 23](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855428)

[Figure 13 Showing Different Layers of Litigat8 25](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855429)

[Figure 14 Showing Trello Board for litigat8- 2 29](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855430)

[Figure 15 Showing Project Timline Managment Using Gantt Chart for Litigat8 -1 29](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855431)

[Figure 16 Showing Trello Board for litigat8- 1 29](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855432)

[Figure 17 Showing Project Timeline Management Using Gantt Chart for Litigat8 -2 30](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855433)

[Figure 18 Showing Project Timeline Management Using Gantt Chart for Litigat8 - 3 30](file:///D:\WorkOpus\Litigate\Final_Litigate.docx#_Toc162855434)

[Figure 19 Showing Registration Wireframe 31](#_Toc162855435)

[Figure 20 Showing Login Wireframe 32](#_Toc162855436)

[Figure 21 Showing Main Chat Interface Wireframe 32](#_Toc162855437)

[Figure 22 Showing **User Model** 33](#_Toc162855438)

[Figure 23 Showing **ChatSession Model** 33](#_Toc162855439)

[Figure 24 Showing **Conversation Model** 33](#_Toc162855440)

[Figure 25 Showing the ERD Diagram for Litigat8 Database 33](#_Toc162855441)

[Figure 26 Showing the Framework for the Data Collection and Preparation for Litigat8 Training 34](#_Toc162855442)

[Figure 27 Showing the Low-Level Design (LLD) for Litigat8 35](#_Toc162855443)

[Figure 28 Showing the Setup of Database Config 36](#_Toc162855444)

[Figure 29 Setting up the Database 36](#_Toc162855445)

[Figure 30 Setting up the Models for Database 37](#_Toc162855446)

[Figure 31 Showing the constructed Table in the Litigate Database 37](#_Toc162855447)

[Figure 32 Showing the Front-end Implementation of Main-Page 38](#_Toc162855448)

[Figure 33 Showing the Front-end Implementation of Login Page of Litigate 39](#_Toc162855449)

[Figure 34 Showing the Front-end Implementation of Register Page of Litigate 40](#_Toc162855450)

[Figure 35 Showing the Front-end Implementation Chat Interface of Litigate 40](#_Toc162855451)

[Figure 36 Showing the implementation of Flask App with Configuration 41](#_Toc162855452)

[Figure 37 Showing the Implementation of Auth Blueprint 42](#_Toc162855453)

[Figure 38 Showing the Implementation of Chat Blueprint 42](#_Toc162855454)

[Figure 39 Showing the setting up of Login Function Server Point 42](#_Toc162855455)

[Figure 40 Showing the setting up of Register Function Server Point 42](#_Toc162855456)

[Figure 41Showing the setting up of Logout Function Server Point 43](#_Toc162855457)

[Figure 42 Showing the implementation of start\_chat\_session Server Point 43](#_Toc162855458)

[Figure 43 Showing the implementation of end\_chat\_session Server Point 43](#_Toc162855459)

[Figure 44 Showing the implementation of server handling of /submit route 44](#_Toc162855460)

[Figure 45 Showing the implementation of server handling of /save\_interaction route 44](#_Toc162855461)

[Figure 46 Showing the implementation of server handling of /get\_conversation/session\_id route 44](#_Toc162855462)

[Figure 47 Showing the implementation of server handling of /get\_chat\_sessions route 45](#_Toc162855463)

[Figure 48 Showing the implementation of server handling of /fetch\_chat\_histories route 45](#_Toc162855464)

[Figure 49 Showing the implementation of Fetch Requests of chat/fetch\_chat\_history 45](#_Toc162855465)

[Figure 50 Showing the implementation of Fetch Requests of chat/start\_chat\_session 46](#_Toc162855466)

[Figure 51 Showing the implementation of Fetch Requests of chat/submit 46](#_Toc162855467)

[Figure 52 Showing the implementation of Fetch Requests of chat/save\_interaction 47](#_Toc162855468)

[Figure 53 Showing the implementation of Fetch Requests of chat/get\_conversations/sessionID and Processing the of JSON load to front-end 47](#_Toc162855469)

[Figure 54 Showing the implementation of Fetch Requests of chat/start\_chat\_session 48](#_Toc162855470)

[Figure 55 Showing the text data structure before pre-processing steps 48](#_Toc162855471)

[Figure 56 Showing the preprocessing steps i.e., Tokenization, Lowercasing, Removing Stop Words, Stemming 49](#_Toc162855472)

[Figure 57 Data Structure after Data Pre-Processing 49](#_Toc162855473)

[Figure 58 Chunking/Spliting of the Text Data 50](#_Toc162855474)

[Figure 59 Data After Chunking/Splitting 50](#_Toc162855475)

[Figure 60 Showing the implementation of OpenAI Embedding Model 50](#_Toc162855476)

[Figure 61 Showing the implementation of Hugging Face Embedding Model all-Mini-LM-L6-v2 51](#_Toc162855477)

[Figure 62 Showing the Implementation of Pinecone Vector Database 51](#_Toc162855478)

[Figure 63 Showing the Setting Up of Pinecone Database on the Website 51](#_Toc162855479)

[Figure 64 Showing the Depreciation Warning of Pinecone 52](#_Toc162855480)

[Figure 65 Showing the creation of Vector Database 52](#_Toc162855481)

[Figure 66 Showing the Implementation of VectorDb Retriever 52](#_Toc162855482)

[Figure 67 Showing the implementation of Retrieval-based QA Chain Using OpenAI() LLM 53](#_Toc162855483)

[Figure 68 Showing the Response Generation with Source Citation 53](#_Toc162855484)

[Figure 69 Setting up the context for the LLM 53](#_Toc162855485)

[Figure 70 Showing the LLM answer to a Question out of Context 54](#_Toc162855486)

[Figure 71 Prompt Engineering Version 1 54](#_Toc162855487)

[Figure 72 Prompt Engineering Version 2 55](#_Toc162855488)

[Figure 73 showing the implementation of gpt-3.5-tubo , llama-2-7b Model , gpt-4.0 55](#_Toc162855489)

[Figure 74 Setting Up the Semantic Similarity func and Comparison func 56](#_Toc162855490)

[Figure 75 Setting up the Prompts and Correct Answers 56](#_Toc162855491)

[Figure 76 Showing the Comparison Result 56](#_Toc162855492)

[Figure 77 Showing the response generated LLMA-2 57](#_Toc162855493)

[Figure 78 Showing the response generated LLMA-2 57](#_Toc162855494)

[Figure 79 Showing the response Generated by gpt-4 57](#_Toc162855495)

[Figure 80 showing the response generated by gpt-3.5 57](#_Toc162855496)

[Figure 81 Showing the results of pytest for Unit Tests 65](#_Toc162855497)

[Figure 82 Showing the Unit Test for Interation with Litigat8 NLP Model 66](#_Toc162855498)

[Figure 83 Showing Warnings Generated by the Tests 66](#_Toc162855499)

[Figure 84 Showing the Fix of Login (Success) component warning 66](#_Toc162855500)

# Introduction

The way people get legal help has changed a lot over time. There was a time when getting legal advice was hard and expensive for most people. As the world changed, so did the need for legal help, which became more complicated. Technology has played a big role in changing how we get legal support today. Now, legal help is a key part of dealing with many problems in life, and it affects how fair and just people think the world is.

Thanks to technology, getting legal advice is much easier than before. Smartphones and apps have gone from just being ways to talk to each other to important tools for handling different parts of our lives, including legal issues. This technology lets people find legal information and help easily, giving them the power to handle their legal matters more actively. But, there's a problem: many good legal apps cost money every month, or people have to use many different apps to find what they need. This can make the whole process frustrating and confusing.

My project, Litigat8, is here to make things better. I've created Litigat8 to be a complete source of legal support, especially for issues between landlords and tenants. Whether it's understanding your rights as a tenant, knowing what a landlord can or cannot do, or getting advice on how to handle a dispute, Litigat8 is designed to help. My app offers easy-to-understand advice and resources for tenant and landlord laws.

Litigat8 is more than just information. It's a tool that helps users track their questions, organize their documents, and manage their legal needs easily. I made this app to be a helper and guide in the complex world of tenant and landlord laws. By making legal support simple, direct, and accessible, my goal is to make everyone feel confident and informed about their rights and responsibilities.

With Litigat8, I aim to change the way legal assistance is provided, making it not just easy to access but also supportive and empowering. I want to create a future where everyone, no matter their situation, can understand and use the law to their advantage. Litigat8 isn't just an app; it's a step towards a world where legal help is a tool for fairness and justice for everyone

Research Methodology

This section delineates the methodical approach employed for conducting a comprehensive literature search, selection, and subsequent analysis focusing on the role of Natural Language Processing (NLP) in improving access to justice.

## Literature Search

The literature search was meticulously performed across various academic databases to ensure a broad and thorough collection of relevant studies. The databases queried included Google Scholar, IEEE Xplore, ScienceDirect, SpringerLink, and the ACM Digital Library. To augment the scope of this research, additional sources were located through the references of seminal articles. The search strategy employed combinations and permutations of key terms such as "Natural Language Processing," "NLP in law," "access to justice," "legal informatics," "automated legal assistance," "bias in NLP applications," "privacy in legal NLP," and "future of NLP in law." Boolean operators (AND, OR) were employed to refine the searches, ensuring a focused retrieval of pertinent literature.

## Selection Criteria

**Inclusion Criteria:**

Peer-reviewed articles and conference papers written in English.

Studies that specifically explore the application of NLP technologies in the legal domain or for enhancing access to justice.

Publications that provide insights into the current trends, challenges, and future directions of NLP in legal applications.

**Exclusion Criteria:**

Non-peer-reviewed articles, grey literature, and opinion pieces.

Research that does not directly address NLP's impact on legal processes or access to justice.

Articles without empirical data, clear results on NLP's effectiveness, or a detailed methodology.

An initial screening based on titles and abstracts was conducted to evaluate each article's relevance to the research objectives. This was followed by a full-text review to ensure that the selected studies met the inclusion criteria. Duplicates were removed, and the selected literature was organized for in-depth analysis.

# Background Research

Natural Language Processing (NLP) serves as a critical bridge between human communication and computer understanding, positioning itself as a foundational pillar in both Artificial Intelligence (AI) and linguistics. Designed to streamline interactions between humans and computers, NLP aims to endow computers with the ability to understand human language with ease, bypassing the need for individuals to master complex computer languages. This discipline emerged from the aspiration to converse with computers using our natural vernacular, simplifying technological interactions. NLP divides into two main branches: Natural Language Understanding (NLU) and Natural Language Generation (NLG), focusing on comprehension and the generation of human-like text, respectively. Linguistics, or the scientific study of language, is crucial to NLP, encompassing everything from phonology (the sounds of language) to semantics (the study of meaning).

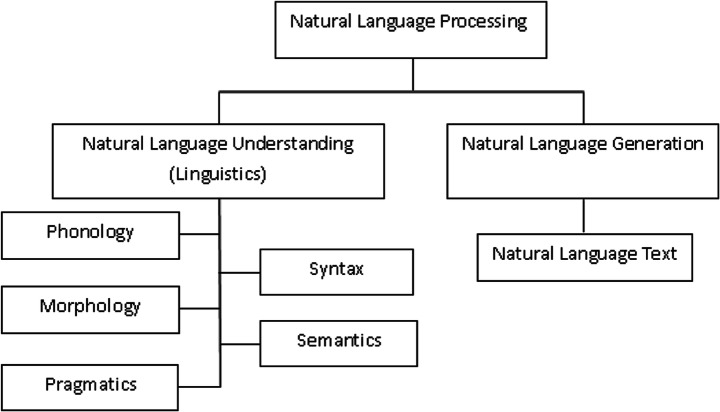


Figure 1 Showing the Hierarchy of NLP

The groundwork for NLP was laid in the mid-20th century with the development of syntactic theories by Noam Chomsky, revolutionizing syntax in theoretical linguistics and setting the stage for future advancements in both NLP and NLG (Chomsky, N., 1965). Historically, the field has been primarily driven by computer scientists, but it has also drawn attention from linguists, psychologists, and philosophers. NLP enhances our comprehension of human language by merging theories and techniques aimed at facilitating natural language communication with computers. It encompasses a variety of tasks, including automatic summarization, machine translation, and optical character recognition, all of which have direct real-world applications.

The evolution from rule-based to statistical methods and, more recently, to deep learning approaches, like those exemplified by BERT and GPT, has significantly advanced the capabilities of NLP systems. These methodologies demonstrate an enhanced level of understanding and text generation, making NLP a key player in modern technological discussions (Sutskever, I., Vinyals, O., and Le, Q.V., 2014).

Moreover, NLP's role extends to improving human-computer interfaces, automating customer service, delivering personalized content, and performing sentiment analysis. In sectors such as legal services, NLP can streamline document analysis, simplify legal research, and make expert advice more accessible to the general public, showcasing its vast potential to democratize specialized knowledge.

Despite these advancements, NLP faces challenges like language ambiguity, understanding context, and addressing the nuances of cultural language variations. Addressing these issues remains an active area of research, aiming to enhance the sophistication of AI in interpreting human language.

# History of NLP

The history and evolution of Natural Language Processing (NLP) can be traced back to the mid-20th century, marking a journey filled with significant milestones, technological advancements, and interdisciplinary contributions. This narrative begins in the late 1940s, a period when NLP, as a term, had yet to be coined, yet the foundational steps towards machine understanding of human language were being laid (Hutchins, W.J., 1986). The initial focus was on Machine Translation (MT), a task that sought to automatically translate text from one language to another, predominantly between Russian and English. This era was characterized by optimism and ambitious projects aimed at breaking down language barriers using computers.

However, the enthusiasm faced a substantial setback with the ALPAC report in 1966, which concluded that the prospects for MT were bleak and recommended a reduction in funding for such research (Hutchins, W.J., 1995). This report significantly dampened the momentum of NLP research, leading to a period of reassessment and recalibration of goals within the field. Despite this, some MT projects continued to operate, slowly refining their approaches and laying the groundwork for future successes.

The revival of NLP research in the 1980s was marked by a shift towards creating more sophisticated models of language understanding and generation. This period saw the emergence of computational linguistics as an active area of study, linking the understanding of language with logical structures and computational algorithms. Influential projects like SHRDLU, an early natural language understanding system, demonstrated the potential for computers to interpret and act upon human instructions within a controlled environment (Winograd, T., 1972). These developments highlighted the importance of syntax, semantics, and pragmatics in modeling language, leading to a deeper exploration of how linguistic principles could be encoded into computational systems.

The 1990s witnessed a paradigm shift with the introduction of statistical methods to NLP (Manning, C.D., and Schütze, H., 1999). The availability of large text corpora and advancements in computing power made it feasible to apply statistical models to language processing tasks. This approach marked a departure from rule-based systems, offering more flexibility and robustness in handling the complexities of human language. The era of statistical NLP laid the foundation for modern techniques, emphasizing the importance of data and probabilistic models in understanding language.



Figure 2 Showing the Development of NLP overtime

Today, NLP stands as a vibrant field at the intersection of AI, linguistics, and computer science, continuously evolving with the introduction of new models, datasets, and applications. Its history reflects a trajectory of overcoming challenges, embracing new methodologies, and expanding the boundaries of what machines can understand and generate in terms of human language. The future of NLP promises further integration with other disciplines, deeper understanding of nuanced linguistic phenomena, and broader applications that extend beyond current capabilities.

# Fundamentals of NLP

## Syntax vs. Semantics

Integrating discussions on syntax versus semantics within the context of Natural Language Processing (NLP) necessitates a nuanced understanding of both concepts, highlighting their distinctive roles and interconnectedness in language comprehension and generation. Drawing upon the foundational overview provided earlier and incorporating references in Harvard style, we can expand the narrative to encompass these critical linguistic components:

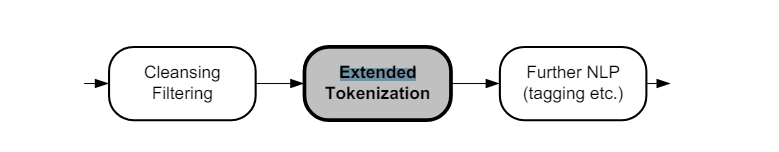
Natural Language Processing (NLP) stands at the crossroads of computer science, artificial intelligence (AI), and linguistics, aiming to bridge the gap between human communication and computational understanding. A pivotal aspect of NLP involves parsing the intricacies of syntax and semantics, each playing a unique yet complementary role in how machines interpret human language.

Syntax refers to the arrangement of words and phrases to create well-formed sentences in a language. It encompasses the rules that govern sentence structure, ensuring that the linguistic expressions are organized in a manner that adheres to the grammatical framework of the language. Syntax is crucial for NLP systems as it enables the parsing of sentences to understand their grammatical composition, which is foundational for further analysis (Chomsky, N., 1965). Without a robust syntactic analysis, machines would struggle to discern the roles of individual words and phrases within sentences, leading to potential misinterpretations of the intended messages.

Semantics, on the other hand, delves into the meaning behind words and sentences, exploring how we comprehend and attribute significance to linguistic expressions. It involves the interpretation of symbols, phrases, and sentences to understand their referential and conceptual content (Jackendoff, R., 2002). In NLP, semantic analysis allows computers to grasp the nuances of language, including idiomatic expressions, metaphors, and variations in context that influence meaning. By understanding semantics, NLP systems can go beyond mere word recognition to infer the intentions and sentiments expressed, facilitating more sophisticated interactions such as sentiment analysis, question-answering, and machine translation.

The interplay between syntax and semantics is fundamental to the NLP field. While syntax structures the form of language, semantics imbues it with meaning. Effective NLP systems leverage syntactic analysis to decode the grammatical structure of language, which, in turn, serves as a scaffold for semantic interpretation. This layered approach ensures that machines can not only recognize the formal aspects of language but also appreciate its content and context, paving the way for more nuanced and human-like understanding and generation of text (Manning, C.D., and Schütze, H., 1999).

## Tokenization

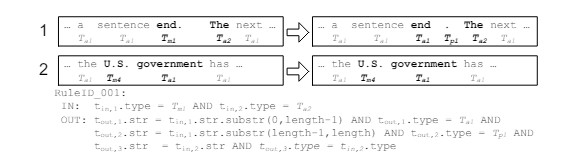
Tokenization is a fundamental process in Natural Language Processing (NLP) that involves breaking down text into smaller units, such as words, phrases, or symbols, known as tokens. This process is crucial for preparing raw text for further NLP tasks such as parsing, part of speech tagging, and semantic analysis. Tokenization simplifies

the complex structure of text, making it more manageable for algorithms to process and analyze.

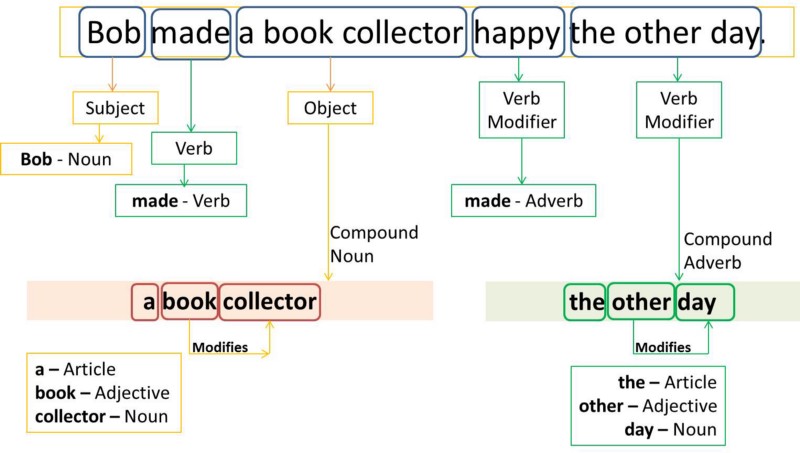
The significance of tokenization lies in its ability to delineate the boundaries between elements in a text, which are essential for understanding the grammatical and semantic

structure of the language (Jurafsky, D., and Martin, J.H., 2019). By segmenting text into tokens, NLP systems can more effectively apply linguistic rules and perform analyses that contribute to tasks like sentiment analysis, machine translation, and information retrieval.

There are several approaches to tokenization, including simple white-space tokenization, where tokens are defined by spaces, and more sophisticated methods that consider punctuation, special characters, and multi-word expressions. Advanced tokenization techniques may also involve linguistic analysis to correctly identify tokens in complex scenarios, such as distinguishing between the period in abbreviations versus the end of a sentence (Manning, C.D., and Schütze, H., 1999).

Tokenization serves as the first step in preprocessing text data, setting the foundation for subsequent NLP tasks. It is essential for ensuring that the input text is in a structured form that computational models can interpret and analyze efficiently. By breaking down text into manageable pieces, tokenization enhances the ability of NLP systems to perform a wide range of linguistic and semantic analyses, contributing to the overall effectiveness of language processing applications (Bird, S., Klein, E., and Loper, E., 2009).

## Part-of-Speech Tagging

Part-of-Speech (POS) Tagging is a critical preprocessing step in Natural Language Processing (NLP) that involves assigning parts of speech to each word in a given text, such as nouns, verbs, adjectives, etc., based on both its definition and its context. This process is fundamental for understanding the grammatical structure of sentences and is instrumental in various NLP tasks including parsing, named entity recognition, and even in machine translation and sentiment analysis.

The importance of POS tagging stems from its ability to contribute to the syntactic analysis of texts, thereby facilitating a deeper understanding of the meanings behind sentences. By categorizing words into their respective parts of speech, NLP systems can apply grammatical rules and perform more complex analyses, such as detecting relationships between words and understanding sentence structure (Jurafsky, D., and Martin, J.H., 2009).

Modern POS tagging utilizes machine learning algorithms, including hidden Markov models, maximum entropy models, and more recently, deep learning approaches that can take advantage of large annotated corpora for training. These models have significantly improved the accuracy of POS tagging, making it a reliable process in automated text analysis (Manning, C.D., 2011).

The application of POS tagging extends beyond simple grammatical analysis. It plays a crucial role in improving the performance of NLP tasks by providing additional contextual information. For instance, in sentiment analysis, knowing the parts of speech can help in accurately identifying opinion words. Similarly, in machine translation, understanding the grammatical role of words in sentences can enhance translation quality by ensuring that words are correctly interpreted and translated according to their usage (Smith, N.A., 2011).

## Name Entity Recognition

Named Entity Recognition (NER) is a crucial task in Natural Language Processing (NLP) that involves identifying and classifying named entities within a text into predefined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc. NER is fundamental for extracting structured information from unstructured text, making it essential for numerous applications like information retrieval, question answering systems, content classification, and knowledge graph construction.

The significance of NER lies in its ability to discern and categorize key elements in text, thereby enabling more effective organization and retrieval of information. This process aids in the understanding of the context and the relationships between entities, which is critical for semantic web applications, summarization, and enhancing search algorithms (Nadeau, D., and Sekine, S., 2007).

Advancements in NER have been driven by machine learning and deep learning techniques, utilizing both supervised and unsupervised learning models. The advent of recurrent neural networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, and more recently, transformer-based models like BERT (Bidirectional Encoder Representations from Transformers), have significantly improved the accuracy and efficiency of NER systems (Lample, G. et al., 2016; Devlin, J. et al., 2019).

These technological advancements have expanded the potential of NER in processing vast amounts of text data across various domains, from news articles and scientific papers to social media posts. By extracting meaningful entities, NER systems contribute to the automation of document analysis, enhancing information extraction processes and enabling more sophisticated data analysis and decision-making processes.

## Dependency Parsing

Dependency Parsing is a vital process in Natural Language Processing (NLP) that focuses on identifying the grammatical structure of sentences by establishing relationships between "head" words and words that modify those heads. This approach highlights how words in a sentence depend on each other for meaning, thereby mapping out the sentence's grammatical structure in terms of dependencies between words. Unlike traditional phrase structure parsing, which constructs a hierarchical tree structure of sentences, dependency parsing builds a tree that represents the dependencies between individual words, making it crucial for understanding the syntactic and semantic relationships within a sentence.

The importance of dependency parsing in NLP cannot be overstated. It enables the analysis of the syntactic structure of sentences, which is essential for numerous advanced NLP tasks, including machine translation, sentiment analysis, information extraction, and question answering. By understanding the grammatical relationships between words, NLP systems can interpret the context and meaning of sentences more accurately, leading to more effective processing of natural language texts (Kübler, S., McDonald, R., and Nivre, J., 2009).

Recent advances in dependency parsing have been driven by machine learning models, especially with the advent of deep learning techniques. These models, particularly those based on neural networks, have significantly improved the accuracy and speed of dependency parsing, enabling the processing of complex languages and intricate sentence structures (Chen, D., and Manning, C.D., 2014). Furthermore, the development of universal dependency treebanks and the use of transformer-based models like BERT (Devlin, J. et al., 2019) have contributed to the cross-linguistic applicability and performance of dependency parsers.

The application of dependency parsing extends beyond linguistic analysis to practical applications such as improving the efficiency of search engines, enhancing the relevance of search results through better understanding of query intent, and automating the summary generation of texts by identifying key syntactic elements.

## Lemmatization and Stemming

Lemmatization and stemming are two text normalization techniques used in Natural Language Processing (NLP) to prepare texts, documents, or words for further processing. Both methods are aimed at reducing the inflectional forms of words to their root form, but they operate differently and serve distinct purposes.

### Lemmatization

Lemmatization is the process of reducing a word to its base or dictionary form, known as the lemma. Unlike stemming, lemmatization considers the context and uses a full vocabulary of a language to apply morphological analysis to words. The lemma of a word is its canonical form or the form you would find in a dictionary. For instance, "running", "ran", and "runs" are all lemmatized to "run". Lemmatization is particularly useful in tasks that require high levels of accuracy and context understanding, such as semantic indexing, text analysis, and comprehensive information retrieval (Manning, C.D., Raghavan, P., and Schütze, H., 2008).

### Stemming

Stemming, on the other hand, is a heuristic process that strips the suffixes from words, aiming to achieve this reduction more aggressively than lemmatization. The objective is to reduce words to their word stem, base or root form—often a part of the word that is not itself a valid word in the language. For example, "fishing", "fished", "fisher" all reduce to the stem "fish". Stemming algorithms, such as the Porter stemmer, are simpler and faster than lemmatization, making them suitable for search queries and systems where the broad recall is more important than precise accuracy (Porter, M.F., 1980).

### Comparative Analysis

While both lemmatization and stemming aim to condense words to their root forms, lemmatization does so with an understanding of the morphological analysis of words, making it more accurate but computationally expensive compared to stemming. Stemming algorithms, due to their heuristic nature, are faster but can generate non-words and are less accurate in understanding the context of the word in a sentence.

## Machine learning in NLP

Machine Learning (ML) plays a pivotal role in Natural Language Processing (NLP), driving advancements that have transformed how machines understand, interpret, and generate human language. By leveraging patterns in data, machine learning algorithms enable computers to perform complex NLP tasks without explicit programming for each specific task. This integration of ML in NLP has facilitated the development of applications such as speech recognition, sentiment analysis, machine translation, and chatbots, among others.

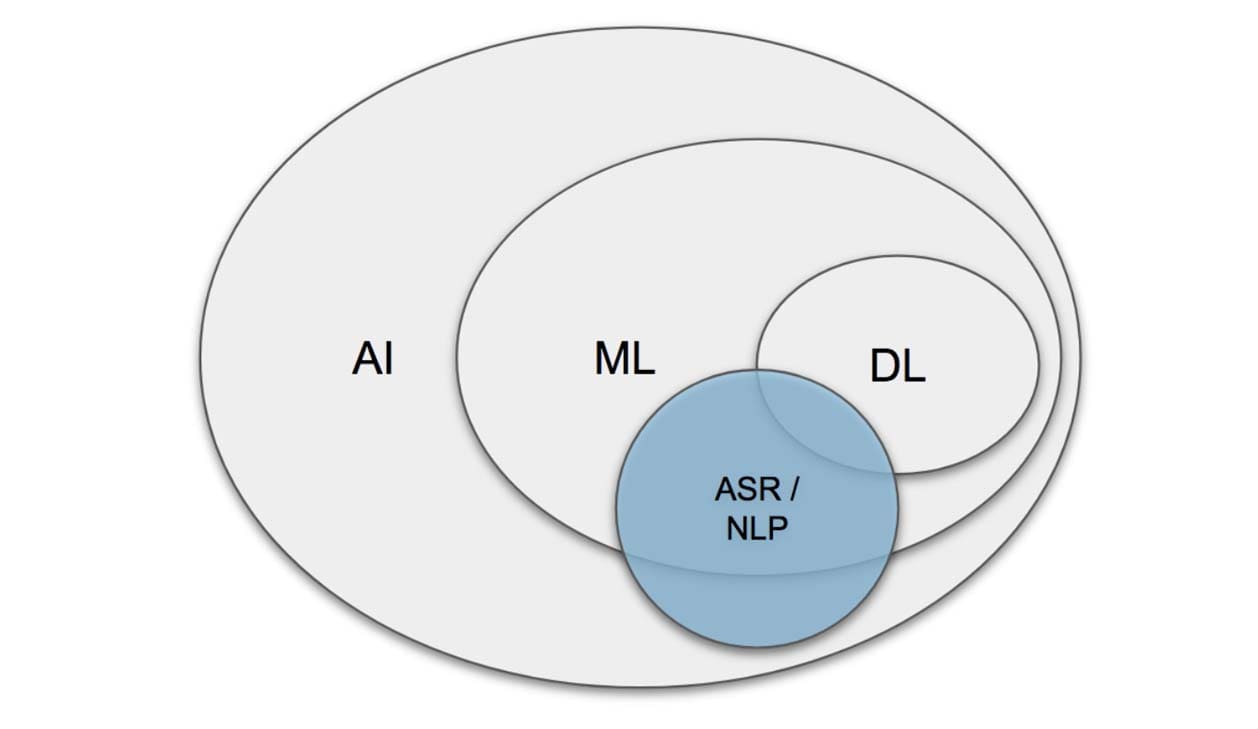
Machine learning models, ranging from traditional algorithms like Naive Bayes, decision trees, and support vector machines (SVMs) to advanced neural network architectures, have been instrumental in NLP. These models are trained on large datasets, learning to predict or classify text data based

Figure 3 Showing the Overlap of NLP with ML and DL

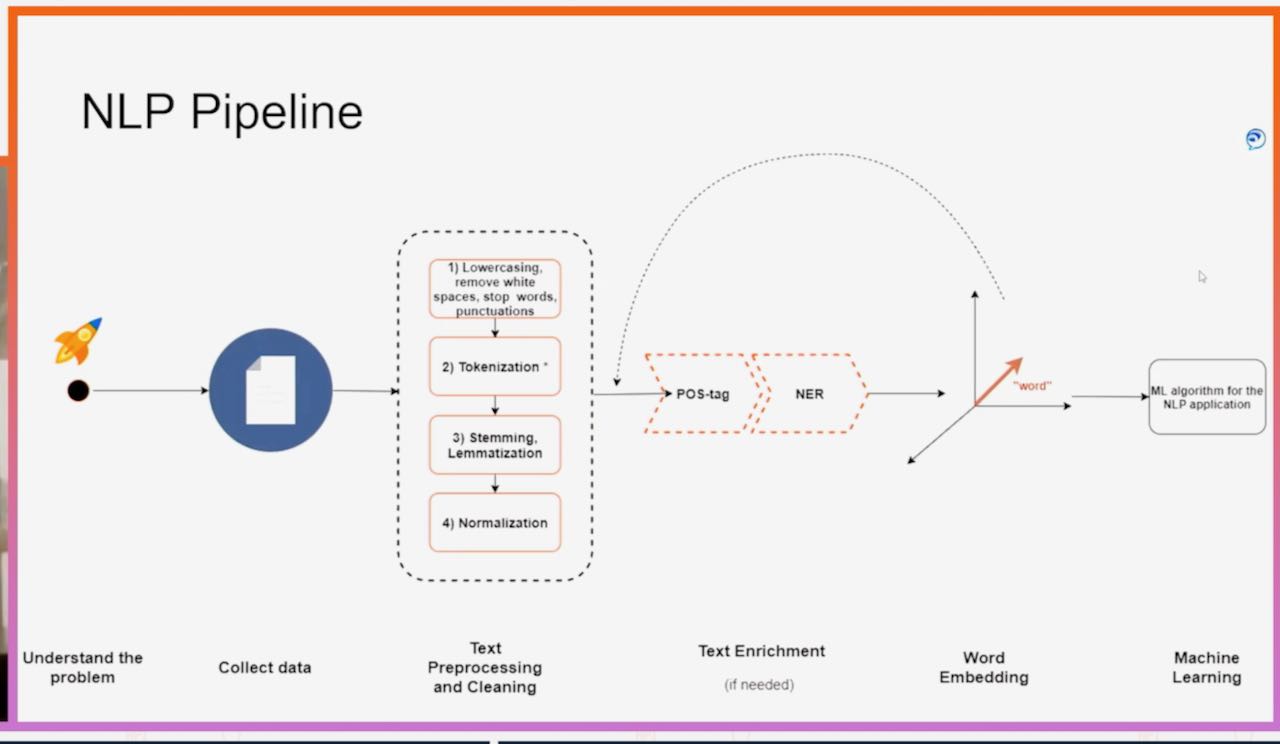
The advent of transformer models, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pretrained Transformer), represents a significant leap forward. These models have set new standards for a wide range of NLP tasks by effectively capturing deep contextual relationships within text (Vaswani, A. et al., 2017; Devlin, J. et al., 2019). Their ability to pre-train on vast amounts of text data and fine-tune for specific NLP tasks has led to unprecedented accuracy in language understanding and generation.

Figure 4 Showing the NLP Pipeline

Despite these advances, integrating machine learning into NLP poses challenges, including the need for large annotated datasets for training, the complexity of understanding language nuances, and the computational resources required for training sophisticated models. Additionally, there's an ongoing effort to improve model interpretability and reduce bias in machine learning-based NLP systems.

# The Role of NLP in Enhancing Access to Justice

The role of Natural Language Processing (NLP) in enhancing access to justice is increasingly significant, leveraging the power of AI to bridge the gap between legal services and those in need. By automating and streamlining legal processes, NLP technologies make legal information more accessible, understandable, and actionable for individuals and communities who may otherwise lack the resources for traditional legal representation.

**Demystifying Legal Language**

One of the primary barriers to accessing justice is the complexity of legal language. NLP tools can translate legal jargon into plain language, making legal documents, regulations, and procedures more understandable to the layperson. By providing summaries of legal texts and explaining legal concepts in simple terms, NLP aids individuals in navigating the legal system more confidently and making informed decisions (Katz, D.M., Bommarito II, M.J., and Blackman, J., 2017).

**Automated Legal Assistance**

Legal chatbots and virtual assistants, powered by NLP, offer preliminary legal advice and assist in document preparation, ranging from simple contracts to more complex legal filings. These applications can guide users through legal processes, ask relevant questions, and generate documents based on the user's responses, significantly lowering the barrier to initiating legal actions or responding to legal issues (Sourdin, T., 2018).

**Enhancing Legal Research**

NLP facilitates advanced legal research by enabling the efficient analysis of vast amounts of legal texts, case law, and legislation. Lawyers and legal researchers can use NLP-powered tools to quickly find relevant precedents, identify legal trends, and gather evidence to support their cases. This not only improves the quality of legal representation but also makes it more cost-effective, indirectly benefiting clients with limited resources (Alarie, B., Niblett, A., and Yoon, A.H., 2018).

**Accessible Dispute Resolution**

Online Dispute Resolution (ODR) platforms utilize NLP to offer accessible means for resolving disputes outside of traditional court settings. By automating parts of the mediation or arbitration process, ODR platforms can resolve conflicts more quickly and with less financial strain on the parties involved, making justice more accessible for all (Rule, C., 2017).

# Identifying and Mitigating Bias

## Privacy and Security Concerns in Legal NLP Applications

Privacy and security concerns are paramount in the deployment of Natural Language Processing (NLP) applications within the legal domain. Legal NLP applications, which handle sensitive and potentially confidential information, necessitate stringent measures to protect client data and ensure compliance with legal standards and regulations. The processing of legal documents, client communications, and other sensitive information by NLP systems raises substantial concerns about data protection, unauthorized access, and the potential misuse of information.

One of the primary challenges in legal NLP applications is ensuring the confidentiality and integrity of the data processed. Legal documents often contain privileged information, trade secrets, and personal data subject to various privacy laws, such as the General Data Protection Regulation (GDPR) in Europe (Voigt, P., and Von dem Bussche, A., 2017). As such, NLP systems used in the legal field must incorporate robust encryption methods for data storage and transmission, alongside secure access controls to prevent unauthorized access to sensitive information.

Furthermore, the use of NLP in legal applications involves ethical considerations regarding the fairness and transparency of automated systems. Bias in NLP models can lead to unfair outcomes or discrimination, undermining the trust in automated legal analyses (Barocas, S., Hardt, M., and Narayanan, A., 2019). Therefore, legal NLP applications must be designed with fairness in mind, incorporating methods to detect and mitigate bias in training data and model predictions.

Another significant concern is the potential for data breaches and the unauthorized disclosure of sensitive information. Legal NLP applications must comply with legal standards for data protection, implementing stringent security protocols and regularly auditing systems for vulnerabilities (Romanosky, S., 2016). M oreover, there is a need for clear guidelines and regulations governing the use of AI and NLP in legal contexts, ensuring that these technologies are used responsibly and ethically.

In summary, privacy and security concerns in legal NLP applications are critical issues that require careful consideration and proactive measures. Protecting sensitive legal information while ensuring the fairness and transparency of NLP systems is essential for maintaining client trust and compliance with legal and ethical standards.

## Current Trends and Future Directions of NLP in law

The integration of Natural Language Processing (NLP) within the legal domain has been transformative, reshaping how legal professionals interact with vast amounts of textual data and streamlining various aspects of legal research, document analysis, and client services. As NLP technologies continue to evolve, several current trends and future directions are emerging, promising to further revolutionize the practice of law.

**Current Trends**

**1**. ***Automated Legal Document Analysis:***

The use of NLP for automating the analysis of legal documents, including contracts, court opinions, and legislation, has become increasingly sophisticated. Tools powered by NLP algorithms can now identify, extract, and summarize relevant information from legal texts, significantly reducing the time and effort required for legal research and due diligence (Zhong, H., Guo, Z., Tu, C., Xiao, C., Liu, Z., and Sun, M., 2020).

**2**. ***Legal Chatbots and Virtual Assistants:***

Legal chatbots and virtual assistants, equipped with NLP capabilities, are becoming more prevalent. These technologies offer legal advice to the public, help in drafting simple legal documents, and provide support for customer service operations in law firms, making legal services more accessible (Kreutzer, R.T., and Sirrenberg, M., 2019).

**Future Directions**

**1. *Enhanced Legal Predictive Analytics:***

The future of NLP in law includes the development of more advanced predictive analytics tools. By analyzing historical legal data, NLP models could predict the outcomes of cases, helping lawyers make better-informed decisions about case strategies and likelihood of success (Ashley, K.D., 2017).

**2.** ***Ethical and Fair Use of NLP in Legal Applications:***

As NLP technologies become more embedded in legal processes, there will be an increased focus on ensuring the ethical use of these tools. This includes addressing concerns related to bias, transparency, and the explainability of NLP models to ensure fair and equitable legal outcomes (Branting, L.K., 2021).

**3. *Cross-lingual and Multijurisdictional Legal NLP Applications:***

Future NLP systems will likely become more adept at handling multiple languages and legal jurisdictions, facilitating cross-border legal research and global compliance tasks. This advancement could significantly benefit international law firms and organizations dealing with multinational legal issues (Tsarapatsanis, D., and Aletras, N., 2021).

# Project Proposal

## Project Details:

Litigate would be an online, hosted chat application accessible on mobile phones and computers. Users can find legal advice related to household and tenant law on the go. There will be a minimum age limit for accessing the platform, but people from all backgrounds will be able to use the application. An NLP model working in the backend, in relation to a database, will process user queries in real-time and produce appropriate responses using NLP techniques. The responses will include basic advice about the matter and, if available, a case law relevant to the issue from the database

## Project Specification:

**Project Scope and Objectives:**

The scope of the project has been defined by the identified said objectives that need to be accomplished throughout the development of the project. The primary objectives carry more weight as they would be allocated the most resources and time before the presentation of the project in April,2024. If the primary objectives are achieved only then the secondary objectives would be considered implementing into the overall workflow of the application.

**Primary Objectives:**

1. ***User Interface Development:*** Design a user-friendly interface for both mobile and desktop platforms that allows users to easily interact with the chat application.
2. ***Natural Language Processing Integration:*** Implement a robust NLP model to understand and process user queries accurately in real-time.
3. ***Legal Database Creation:*** Compile a comprehensive database that includes relevant case laws, statutes, and legal precedents pertaining to household and tenant law.
4. ***Real-time Response Generation:*** Develop a system capable of generating accurate legal advice and relevant case law references in response to user queries.

**Secondary Objectives:**

1. ***Accessibility and Inclusivity*:** Ensure the app is accessible to users from all backgrounds, with considerations for those with disabilities
2. ***Security and Privacy:*** Implement robust security measures to protect user data and ensure privacy, especially when handling sensitive legal queries.
3. ***User Authentication:*** Create a secure user authentication system requirement for accessing the platform.
4. ***Feedback Mechanism:*** Incorporate a feedback mechanism to collect user responses on the accuracy and helpfulness of the legal advice provided, facilitating continuous improvement.

## Project User Stories

Keeping the Objectives and the idea of the project in mind. I have brainstormed the potential use cases a user can have with my application. Which would further be broken down using other analysis techniques such as MoSCoW and depending on the resources and time available.

**Use Case 1:** User Registration

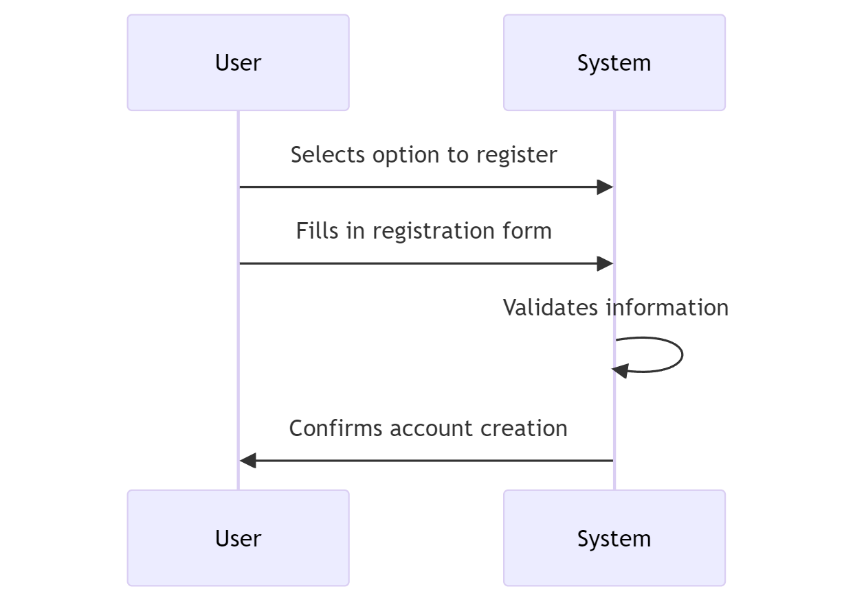


Figure 5 Showing User Registration Use Case

**Use Case 2:** User Login

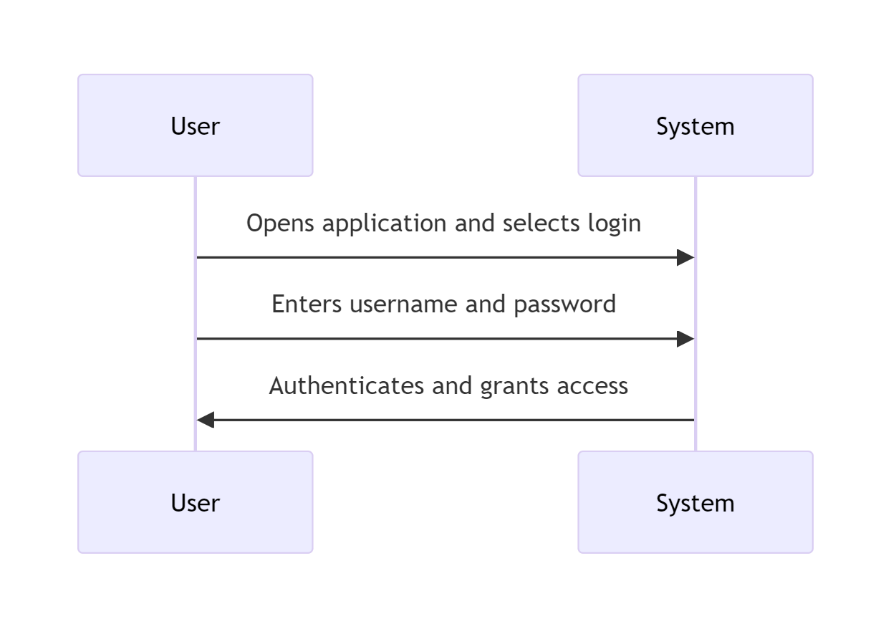


Figure 6 Showing User Login Use Case

**Use Case 3:** User Receives Legal Advice

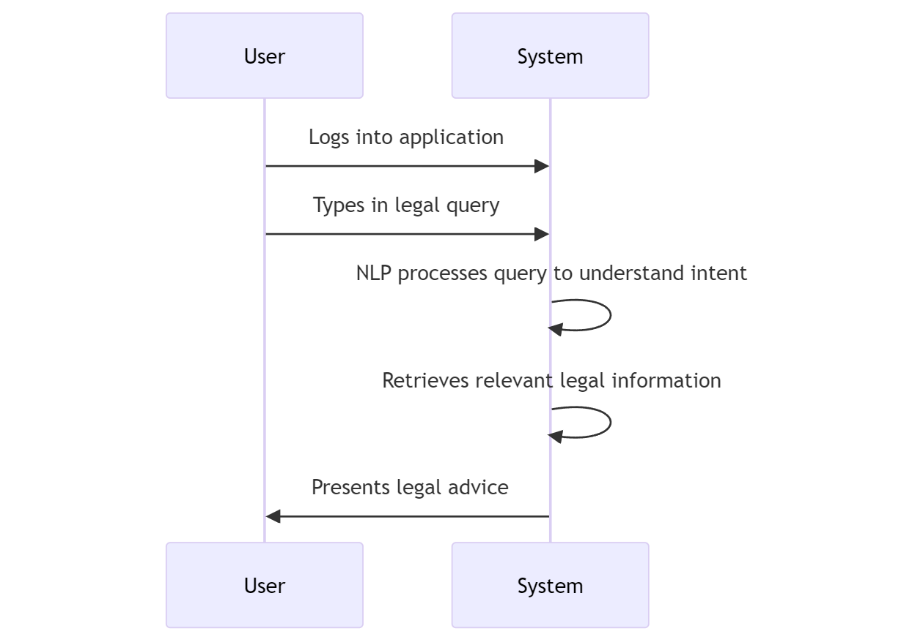


Figure 7 Showing User Legal Advice Use Case

**Use Case 4:** User Reviews Chat History

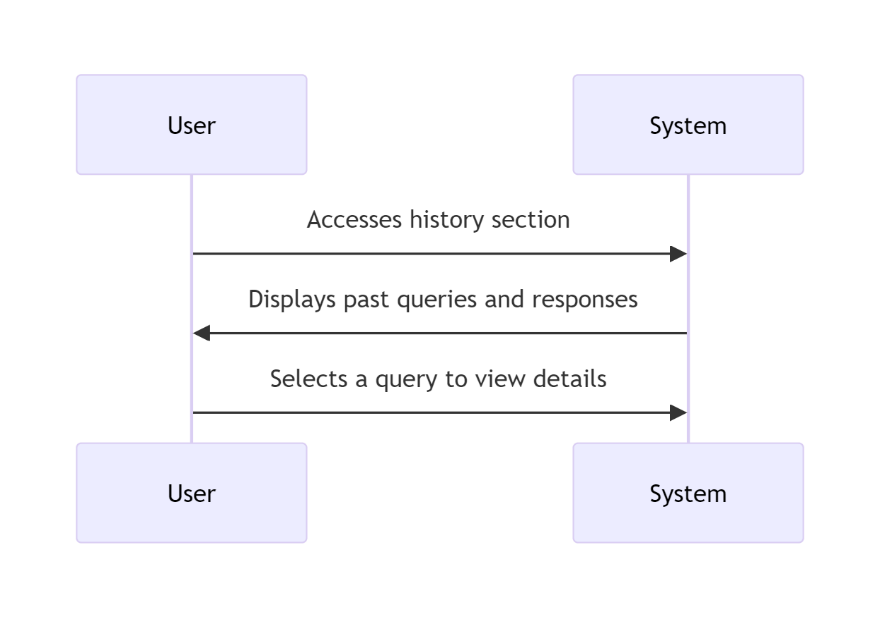


Figure 8 Showing User Chat History Review Use Case

**Use Case 5:** User Provides Feedback

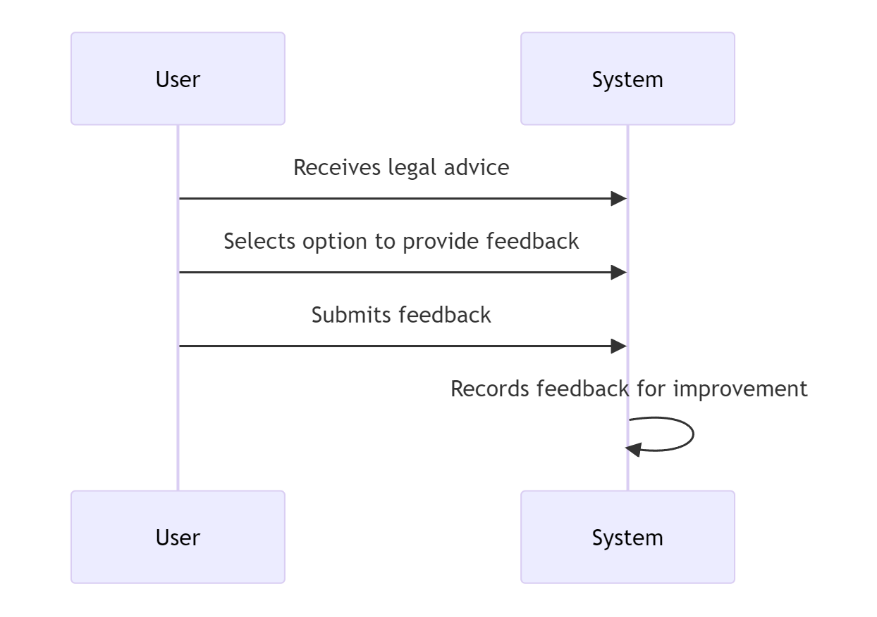


Figure 9 Showing User Providing Feedback Use Case

**Use Case 6:** User Logs Out

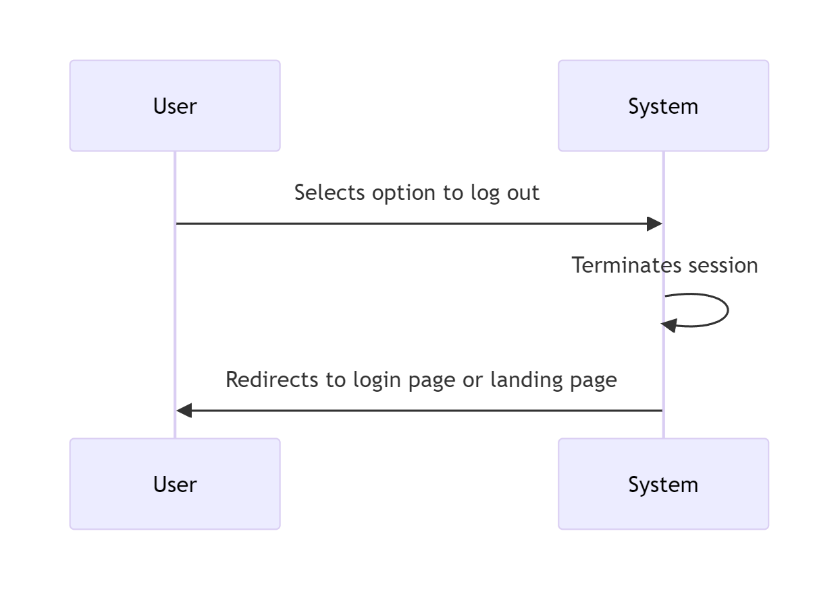
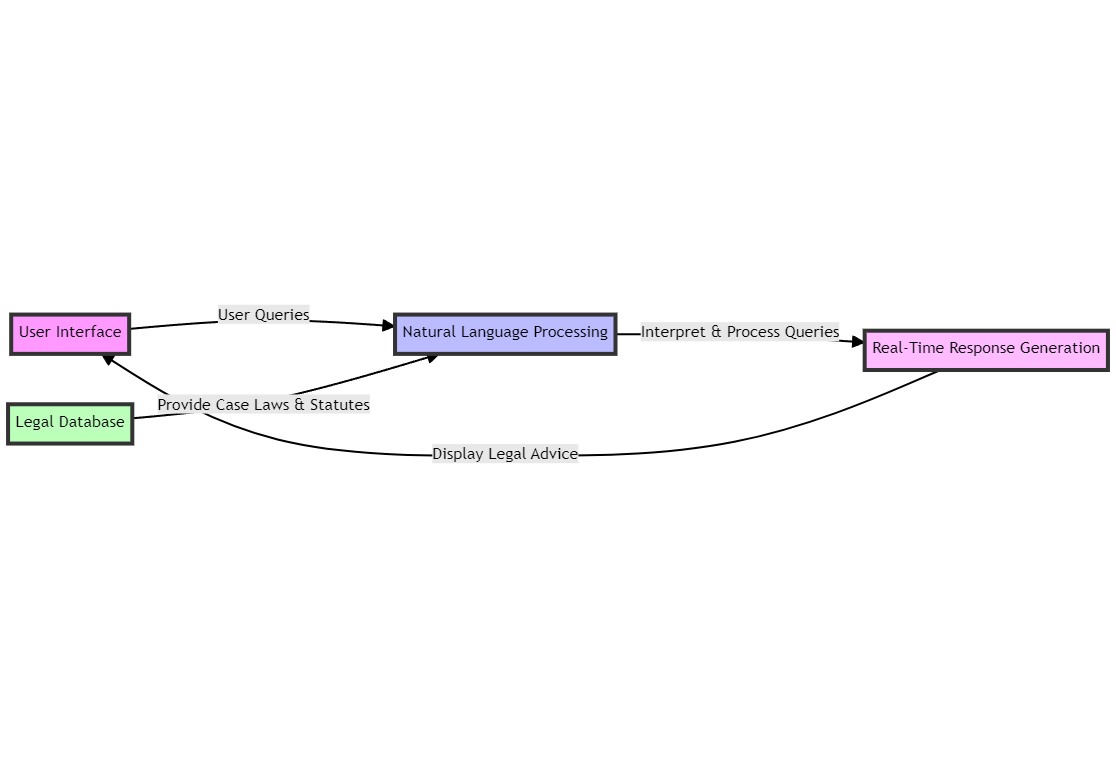


Figure 10 Showing User Logout Use Case

## Traceability Matrix:

# High Level System Design:

In line with the user stories I've decided to focus on, sketching out a high-level design (HLD) helps us map out the system's architecture in a nutshell. This overview helps pinpoint the necessary hardware and software interactions, data exchanges, and communication flows across the system. At the heart of the architecture is a user interface designed to retrieve and update information from database tables. One of the key advantages of our approach is its scalability. As new data needs arise in the future, our system can easily expand to accommodate these additional data sources.

Figure 11Showing General Overview of the Application

This diagram shows a high-level flow of data between different software and hardware components.

One of the disadvantages of using the architectural design and solution, is the current technological skillset that is needed to achieve this implementation of the project. The Mitigation of the potential risks will be taken into account by discussing the it with the supervisor every week.

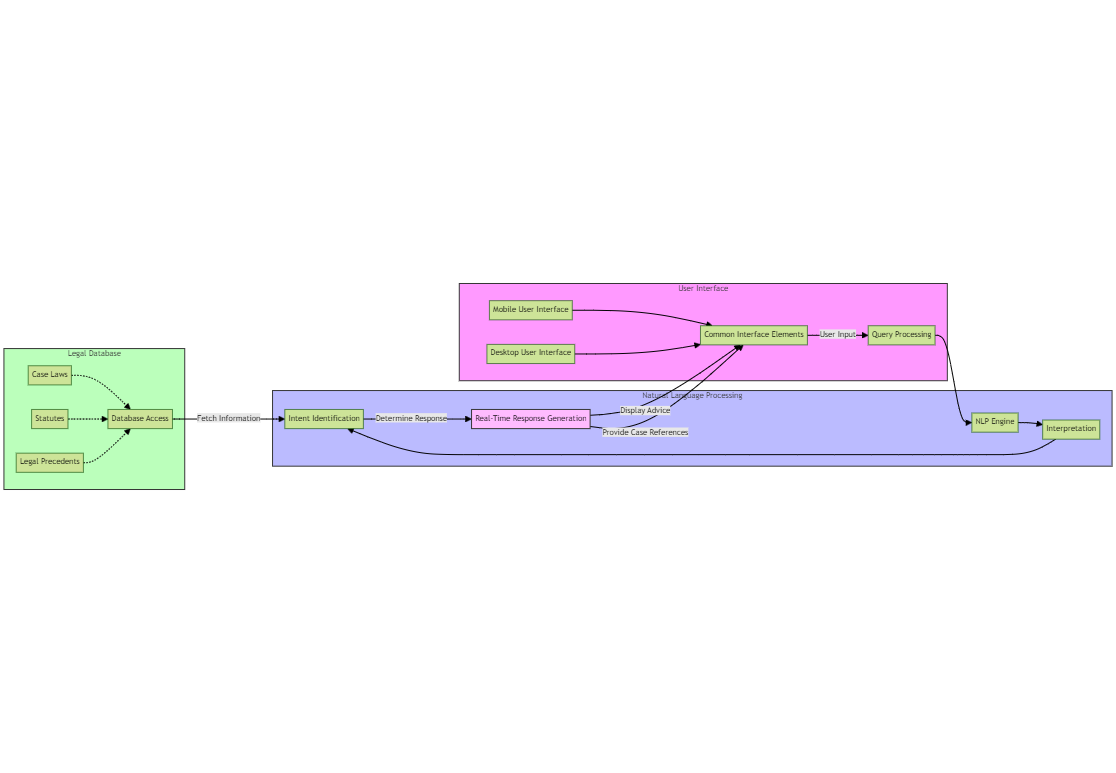


Figure 12 Showing HLD Design of Litigat8

The diagram presented offers a detailed view of our proposed high-level design, showcasing the different elements, connections, and data movements.

# Requirement Analysis:

The requirements would be defined using the MoSCoW Analysis and according to the objectives defined before.

## MoSCoW Analysis

Based on the objectives mentioned in the Project Scope and objective I was able to shortlist the requirements for the project using the Moscow Method. Where the Must have requirements take the highest priority followed by should have and could have requirements respectively. And I thought of the requirements that could be implemented as a part of future development of the project

**Must Have**

|  |  |  |
| --- | --- | --- |
| **ID** | **System Component** | **Requirement Description** |
| M1 | User Account Management | User registration with comprehensive data validation. |
| M2 | User Account Management | Secure user authentication system implementation. |
| M3 | User Account Management | Effective user session management and secure logout. |
| M4 | User Account Management | End-to-end encryption of user data for security. |
| M5 | Core System Functionality | Integration of NLP engine for natural language processing. |
| M6 | Core System Functionality | Real-time legal advice generation based on NLP analysis. |
| M7 | Core System Functionality | Legal database connectivity for dynamic access. |
| M8 | Core System Functionality | Automated retrieval and presentation of case laws and statutes. |

**Should Have:**

|  |  |  |
| --- | --- | --- |
| **ID** | **System Component** | **Requirement Description** |
| S1 | User Interface | Responsive design for both mobile and desktop platforms. |
| S2 | User Experience | Feature for saving and displaying user query history. |

**Could Have:**

|  |  |  |
| --- | --- | --- |
| **ID** | **System Component** | **Requirement Description** |
| C1 | Extended Functionality | Multilingual support to cater to a diverse user base. |
| C2 | Extended Functionality | Notification system to alert users about updates or responses. |

**Won’t Have:**

|  |  |  |
| --- | --- | --- |
| **ID** | **System Component** | **Requirement Description** |
| W1 | Future Considerations | Hosting on a real domain for processing real-time queries. |
| W2 | Future Considerations | Advanced document generation capabilities. |

## Project System Requirement:

Based on the requirement analysis done for the project. I have decided on the software and hardware requirements of the project. As the requirements of the project are modest and it would only be hosted on the localhost for the MVP. I have kept the requirements at minimum for the functioning of the project.

**Development Languages and Tools**:

1. ***Backend Programming Language:*** Python, chosen for its extensive library support, particularly for natural language processing (NLP) tasks. I decided to choose this language for the access of the existing information for python in compared to other languages such as C++. And another factor that contributed to its selection was my own proficiency in the language given the duration of the project. It would have been very highly unlikely to complete the project if I had decided to proceed with the language, I didn’t have command on.
2. ***Frontend Web Technologies:*** HTML, CSS, and JavaScript, for creating an interactive and user-friendly web interface. The User interface is decided to kept simple as most of the resources and time would be allocated to developing the NLP model. Hence, no frontend framework is being used at the moment but if time is left. I’m planning to implement React as the frontend framework for the application.
3. ***NLP Library:*** NLTK, utilized for implementing the chat application's NLP capabilities to process and understand user queries. For developing and training the model I’m going to be using Torch library. Which would be used in the definition of neural networks and processing the input from the user. From the NLTK libraries, I’m going to be using PortStemmer to implement functions such as tokenization, stemming and bag of words.
4. ***Web Framework:*** Flask, selected for its simplicity and efficiency in setting up lightweight web applications, suitable for a project focused on functionality demonstration. Option of Django was considered as well for the development of the project but given the simple nature of the MVP I decided to settle down for flask. And I had some previous experience using this framework so it was a natural choice for me.
5. ***Database System:*** SQLite or Firebase, due to its ease of integration with Python applications and sufficiency for the storage needs of a development-stage project.
6. **Integrated Development Environment(IDE):** VsCode, recommended for its support for Python and web technologies, alongside integrated version control.

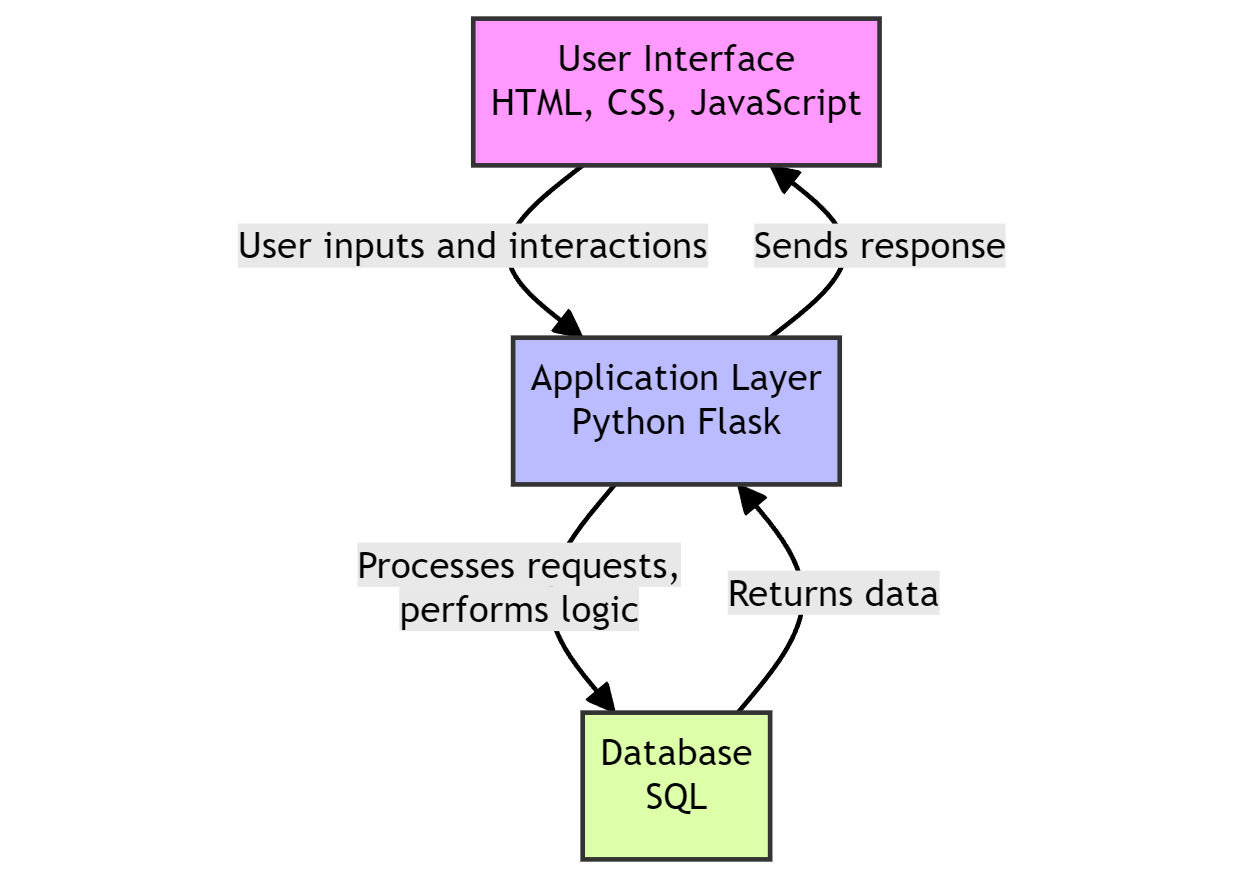
****

Figure 13 Showing Different Layers of Litigat8

**Hardware Requirements:**

Given the localized hosting and demonstration focus of the project, the hardware requirements are kept simple:

1. ***Processor***: A minimum of a Dual-core processor is required to ensure smooth running of the development server and NLP processes.
2. ***Memory:*** At least 4 GB RAM to support the simultaneous execution of the development environment, web server, and any ancillary tools.
3. ***Storage:*** A minimum of 5 GB of available disk space to accommodate the application codebase, SQLite database, and dependencies.

# Project Solution

## Assumptions, Constraints, Risks:

**Assumptions:**

***Technical Capabilities:***

It's assumed that the chosen technologies and frameworks for NLP(NLTK), machine learning (Torch), and other functionalities will effectively support the development of Litigat8.As this is a realm still to be explored and considering the limited time and resources available full-scale deployment of the project can be somewhat troublesome. But the given requirements for a Minimum Viable Product (MVP). It is assumed that all the technical requirements are satisfactory.

***Data Availability:***

Adequate and accurate tenant-landlord law documents, statutes, and regulations are available for populating the database. A I’m going to be preparing the dataset myself. It is assumed that the dataset would have adequate information for the model to produce appropriate responses for the user. And all the information would contain in the dataset is correct. Even though there can be discrepancies in the dataset

***User Feedback:***

Users will provide feedback for system enhancement and that the feedback will be representative and constructive. It is assumed that the feedback provided is constructive and explainable but still there can be instances where the feedback isn’t constructive or comprehensible. And User Feedback is one of the secondary objectives hence it would take less resources during the production of software.

***Compliance and Legal Clearance:***

All necessary legal clearances and compliance requirements related to providing legal advice and handling user data will be obtained. It is assumed that the model has been approved of giving legal advices in the matters of household and tenant law. As law and AI is a very tricky domain where you have to be careful about the data that is produced and dispersed to the user. As this is a demonstration of its capabilities and the project is being made for ONLY this purpose. Hence, it is assumed it has all the legal clearance

***Budget and Time:***

The project will be completed within the allocated budget and time frame. According to the defined user stories and analysis of requirements using methods like MoSCoW. It is assumed that all the set requirements would be completed within Budget and Time. But as this can vary depending on various factors. But for the sake of convivence an assumption is put into place.

**Constraints:**

***Single Stakeholder:***

As the sole stakeholder, your availability and decision-making will be critical for the project's progress and direction. As I’m going to be the only one that would be directing the progress of the project along with the supervisor which can put a bit of constraint on the project in a sense that I would have to make all the critical decision instead of having a shared responsibility of the decision-making progress.

***Budget Limitations:***

Budget constraints may limit the scope of development, particularly concerning the scalability and future plans for web and mobile interface accessibility. The options of cloud hosting and own professional domain will be looked in the future if there are enough resources allocated to the project if there are investors involved in the project but for this development cycle it is constraint to be hosted locally without a domain.

***Technological Limitations:***

The performance of Litigat8 may be constrained by the capabilities of the chosen technologies and frameworks. Some of the technologies chosen for the project for instance python in itself may not be the best choice when it comes to making heavy load models which are capable of processing natural language. But for the sake of demonstration and to have an MVP to show for by the end of April. Some of the technologies were given more weight.

***Geographical and Jurisdictional Limitations:***

Legal advice and statutes provided will be limited to specific geographical regions or jurisdictions. As this model is being developed just for the jurisdiction of UK law hence it would only be able to provide support and advice for Tenant and Household law in UK.

**Risks:**

Some of the foreseeable risk have been identified that could potentially happen in the duration of the project. The mitigation technigques for these risks would be discussed with the supervisor in the weekly meetings. And sufficient alternative pathways would be setup for the project if I encounter any unforeseen circumstances which could lead to failure of any component of the project.

***Data Privacy and Security:***

Risks associated with user data privacy and security breaches. As all the data shared by the user would be of confidential nature because of it being associated with their private matters regarding household and tenant law. Hence, there are risk of data breach in a sense if a third-party gets access to the history of the user chats and may have a malignant intent. Thus for the mitigation of this risk sufficient steps would be taken to ensure the data is kept safe.

***Technology Failure:***

Risks of technology failure impacting the accuracy and reliability of legal advice provided. As there are going to be lots of functioning elements in the project from the interface to the database lots of different factors would be involved. Hence, there is a risk of technological failure if one of the components crashes or fails due to unforeseen circumstances. But sufficient steps would be taken in order to prevent that from happening. Ensuring good coding practices and proper documentation of the project to ensure any unforeseen circumstances should be delt with using proper protocols put in place.

***Dependency on External Systems:***

Future dependency on external cloud hosting platforms and cloud-based database like firebase. I have yet to decided if I want to go for a inbuild database like SQLite or Firebase. If I decided on settling for firebase then there is a potential risk that the application might not work because of being depended on an external firebase database. But steps would be taken and would be discussed with the supervisor to mitigate the risk as much as possible.

## Solution Description:

Based on the extensive information that was collected during the research and the requirement analysis of the project, sufficient evidence is provided for the case that a web application would be suffice to resolve the said user stories and objectives that have been identified for the project. A web application would enable to design a proper interface for the user for ease of access along with a functioning backend layer with an NLP model which in itself is very intensive and alongside that would the database for the whole system which would allow to create, update and delete user data along with the chat data. However, due to number of limitations, it is limited in its ability to create complete system. Therefore, the proposed system would deliver a proof of concept, which would only show the potential of the system in case if it’s implemented as a complete system.

The poof-of-concept application (litigat8) will showcase the process to input the query as a string and then the system would be able to process the information to come with appropriate responses along with case laws and statues if they exist for that specific case

The application would showcase how the user would be able to go back to his/her queries if he or she wants to look over his/her queries again.

# Solution Delivery Approach:

## Project Management Approach:

**Agile Methodology**

In line with modern software development practices, the project adopted the Agile methodology starting from October 15, with a projected end date of April 15. This approach, characterized by its flexibility and iterative nature, emphasizes adaptability and responsiveness to evolving requirements. Agile methodology, as defined by Beck et al. (2001), offers a dynamic alternative to the rigid, linear progression of the Waterfall model. It is conducive to environments like ours, where user needs and system functionalities may not be fully delineated from the onset.

**Project Management Using Scrum Framework**

Scrum, a subset of Agile, was chosen as the operational framework for its fit with the project’s objectives. It provided a structured yet adaptable environment to create a user-centered platform. The Scrum methodology facilitated constant supervisor engagement, enabling me to incorporate feedback iteratively and ensure that the platform remains aligned with the project objectives.

Weekly sprints were scheduled, wherein I focused on delivering specific, prioritized features from the product backlog. Regular sprint reviews and retrospectives ensured that the project adapted to feedback and improved upon the processes continuously.

**Project Management Tools**

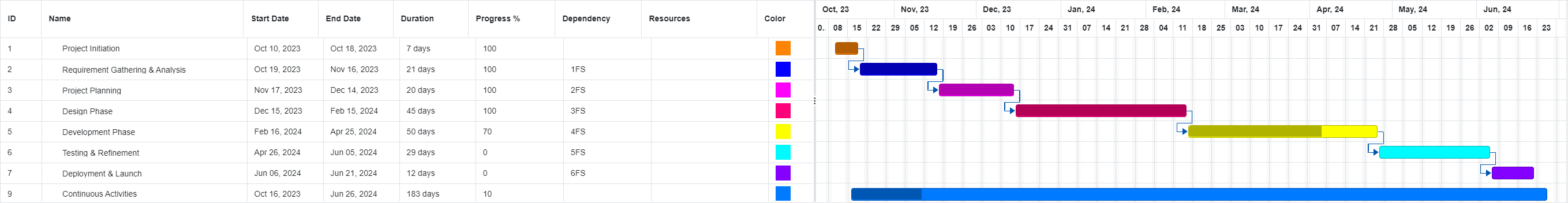
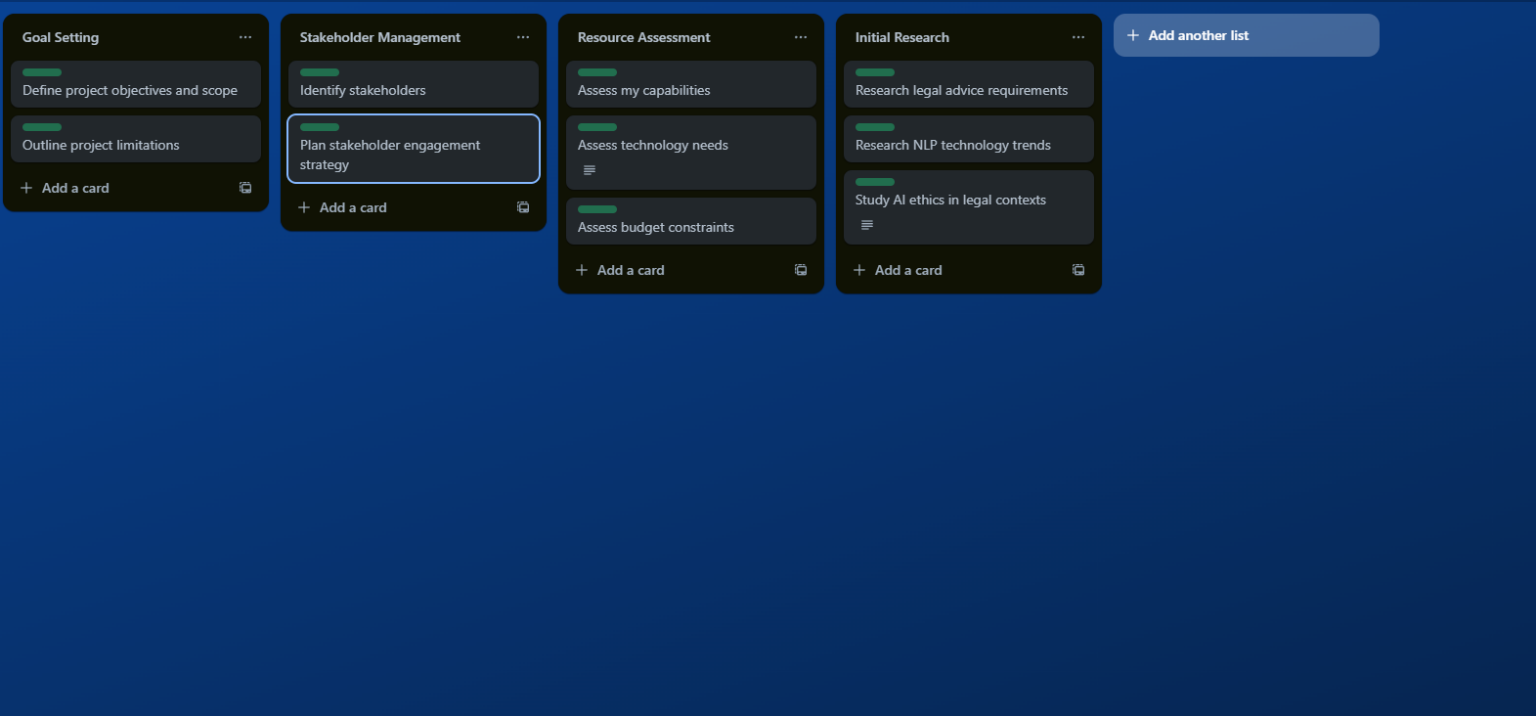
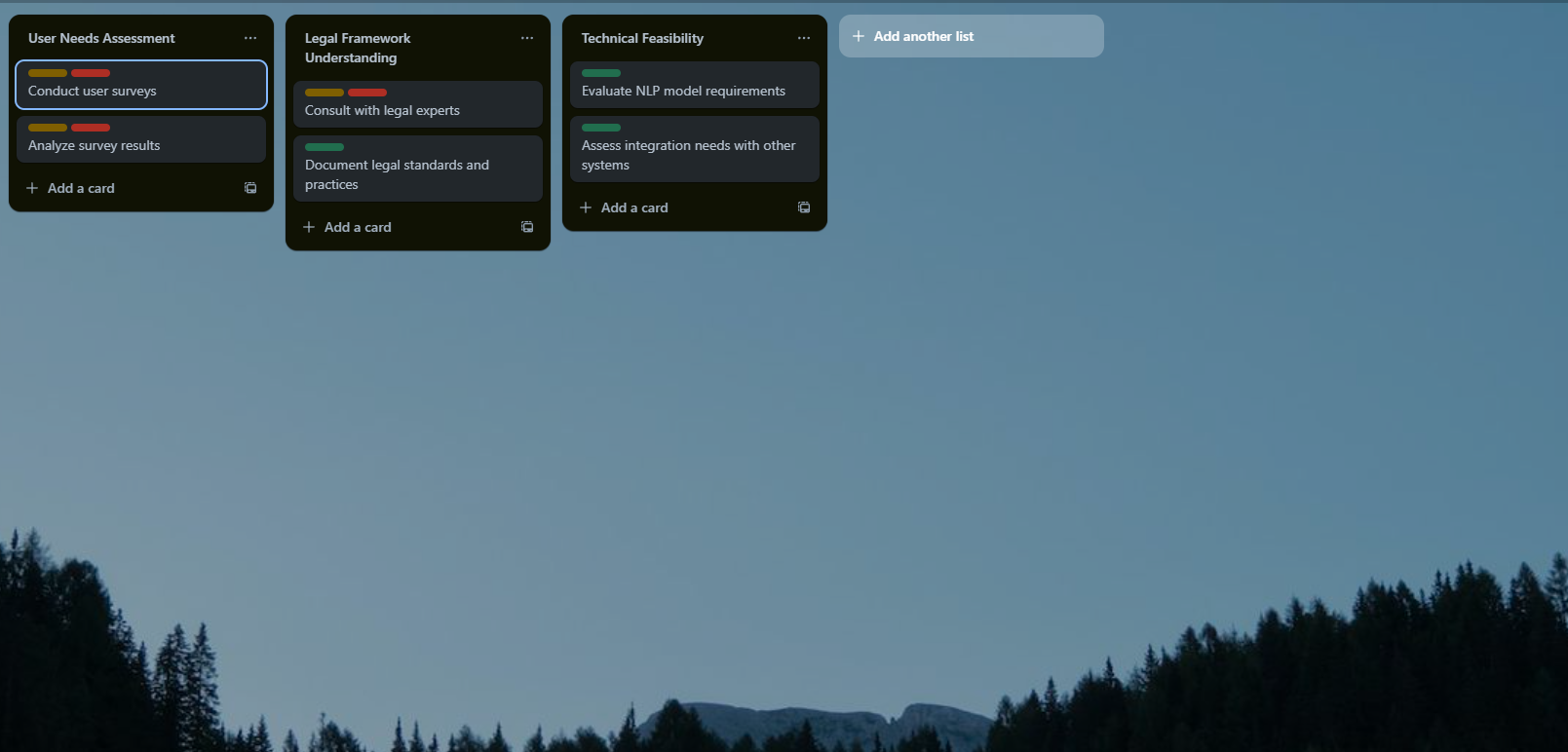
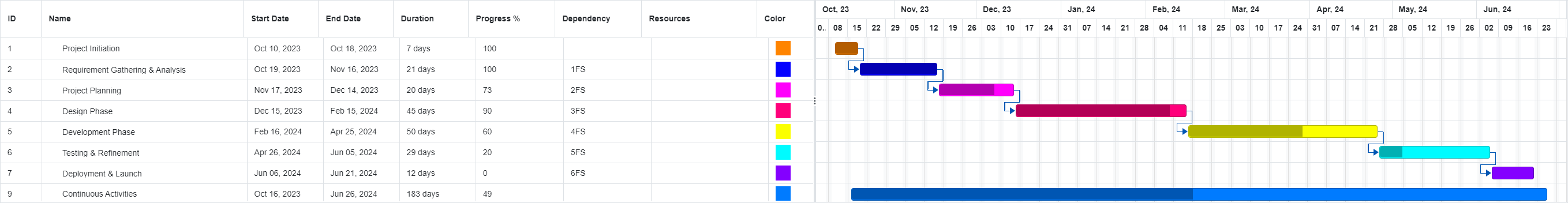
Trello was instrumental in organizing the workflow. It enabled the categorization of tasks into boards representing different Scrum artifacts (product backlog, sprint backlog, in-progress, and done). Trello's visual interface and ease of reorganization catered to the dynamic nature of Agile project management, providing transparency and a high-level overview of the project's status at any given time.

Figure 14 Showing Trello Board for litigat8- 2

Figure 15 Showing Project Timline Managment Using Gantt Chart for Litigat8 -1

Figure 16 Showing Trello Board for litigat8- 1

Figure 17 Showing Project Timeline Management Using Gantt Chart for Litigat8 -2

**Gantt Chart**

Despite Agile's emphasis on flexibility, the project's overarching timeline was charted

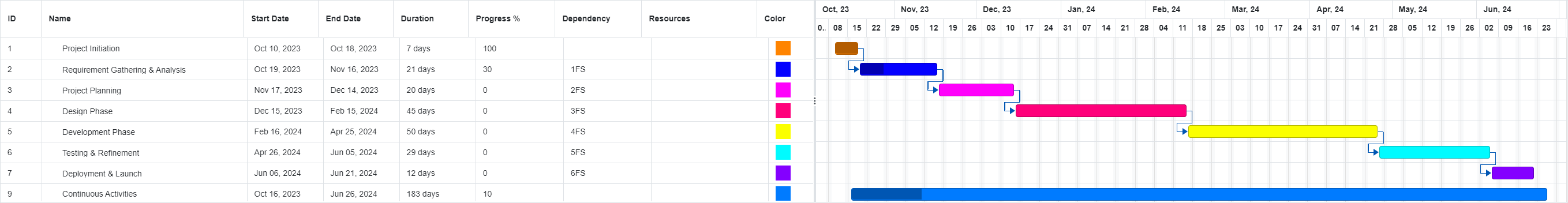


Figure 18 Showing Project Timeline Management Using Gantt Chart for Litigat8 - 3

out using a Gantt chart. This served as a visual planning tool to set key milestones and task durations. It offered a macroscopic view of the project's lifecycle, ensuring adherence to the overarching deadline, and facilitated the management of dependencies between tasks.

**Version Control with GitHub**

GitHub was selected for version control, considering its robust platform for collaborative coding and its prevalence in the software development community. GitHub’s branching mechanism allowed for the isolated development of features, with

subsequent integration into the main codebase after thorough review. This practice not only ensured the integrity and continuity of the codebase but also encouraged experimental development without risking the stability of the system.

The project’s GitHub repository acted as a single source of truth for code changes, providing a comprehensive audit trail for contributions, discussions, and modifications.

# Design Phase:

This stage marks the design phase of the application where I would be designing the required systems that would be operating within the

## Wireframes:

Wireframes were used to form a design the front-end for the application keeping in mind accessibility and functionality. A great consideration was taken to keep in my mind the requirements as well for the system so that it performs as such as it is planned. The wireframes were designed in Figma using the tools provided by the application. The aim was to make the fron-end as intuitive as much as possible so the users don’t have any problem navigating through the application.

**Registration:**

The registration page was designed to accomplish the first objective of having a user authentication system. The frontend was kept as simple as possible with three input placeholders for name/username, email and password. This design would help the users with the ease of registration.

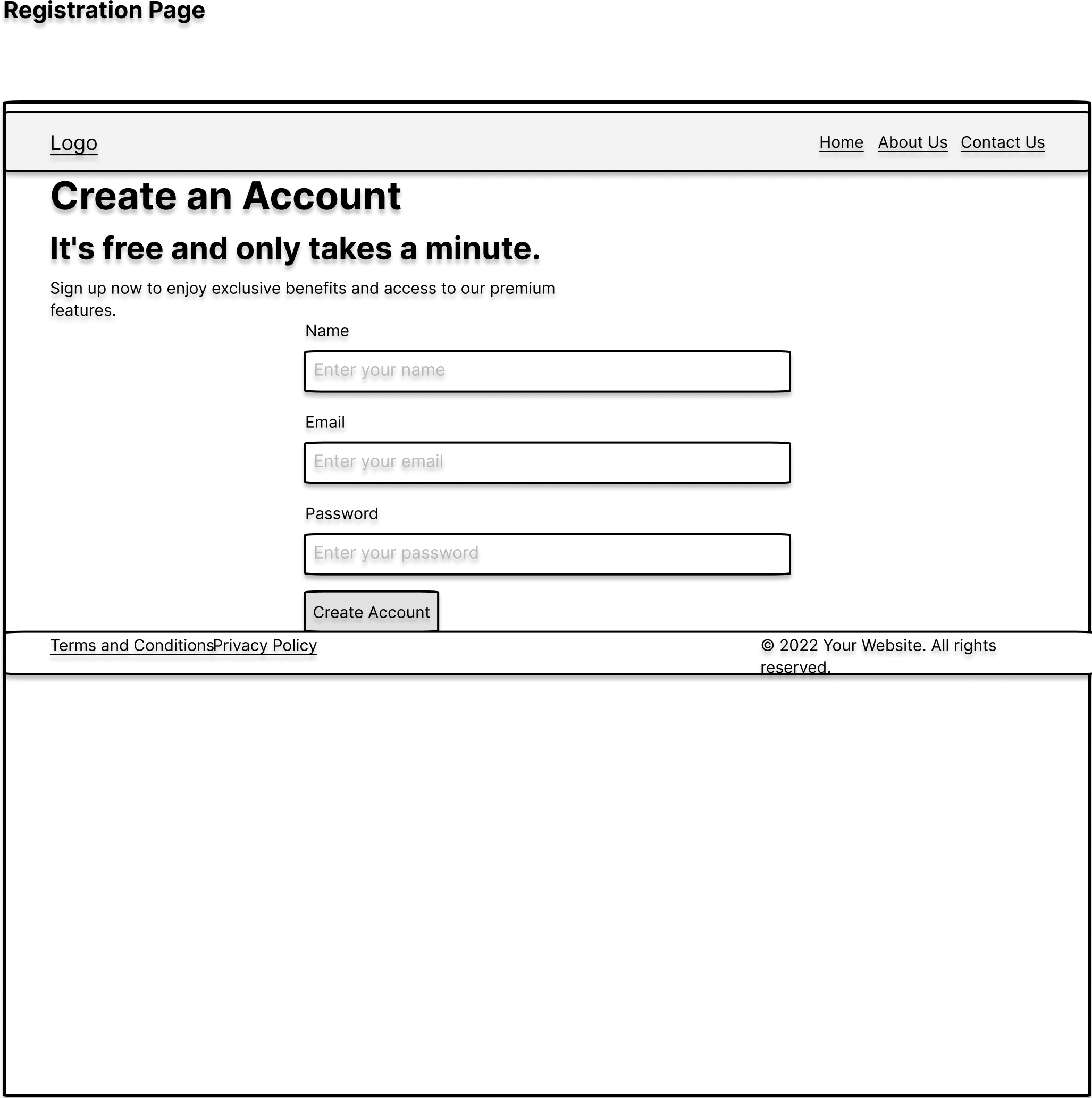


Figure 19 Showing Registration Wireframe

**Login:**

Extending on the authentication system keeping the same objective in mind to keep it simple. The front-end was designed with two input placeholders for email/username and password. Thus making the logging in process as simple as possible for the User.

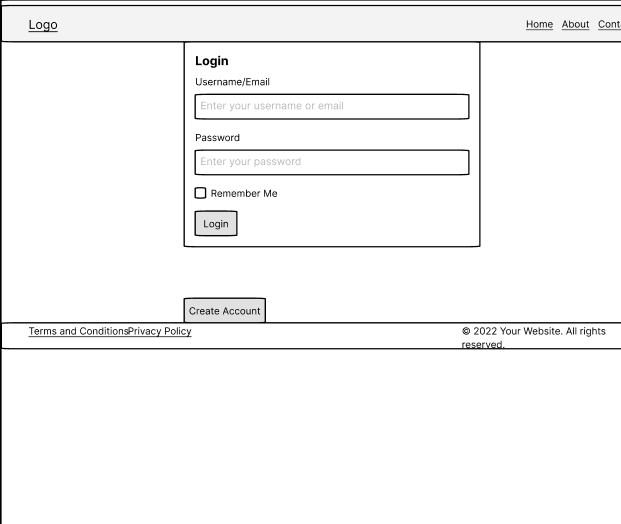


Figure 20 Showing Login Wireframe

**Chat Main Page:**

The main page was designed following the same intuition of accessibility and functionality. The design was kept simple with the input prompt at the bottom and a simple send button to send queries to the NLP model to be processed at the backend. On the Left side is the chat history if the user wants to refer to it at anytime in the future or to go back to reflect on it.

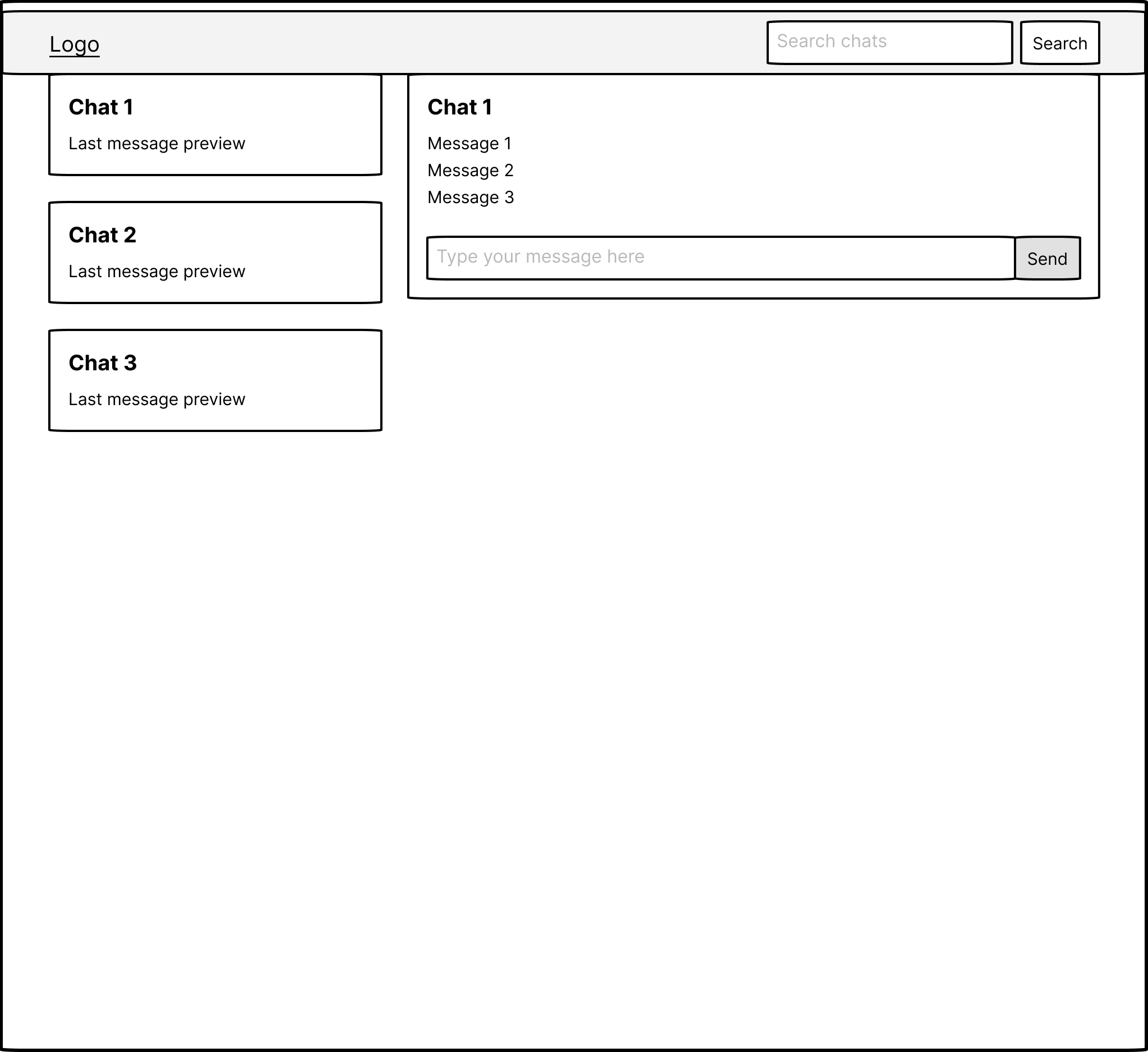


Figure 21 Showing Main Chat Interface Wireframe

The sections concludes the front-end design of the application of the application. As most of the utility would be designing the backend/NLP model. But ample consideration was give to the frontend to improve the user experience as much as possible.

## Database Design:

This schema defines a simple yet effective structure for managing users, chat sessions, and conversations in a chat application. The reasoning behind the design choices is as follows:

* **User Model**: Represents the individuals using the chat application. Each user has a unique username and a password. Storing the password implies that it should be securely hashed before storage to ensure security. The uniqueness of the username ensures that each user can be distinctly identified.

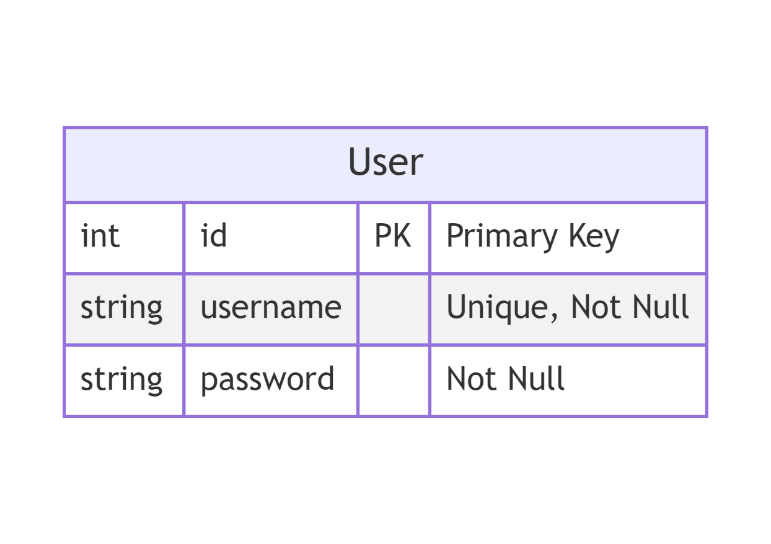


Figure 22 Showing **User Model**

* **ChatSession Model**: Each session represents a period during which a user is actively engaged in conversation. It is linked to the User model via a foreign key (**user\_id**), establishing a one-to-many relationship between users and chat sessions. This means a single user can have multiple chat sessions over time. The **start\_time** field automatically records when each chat session begins.

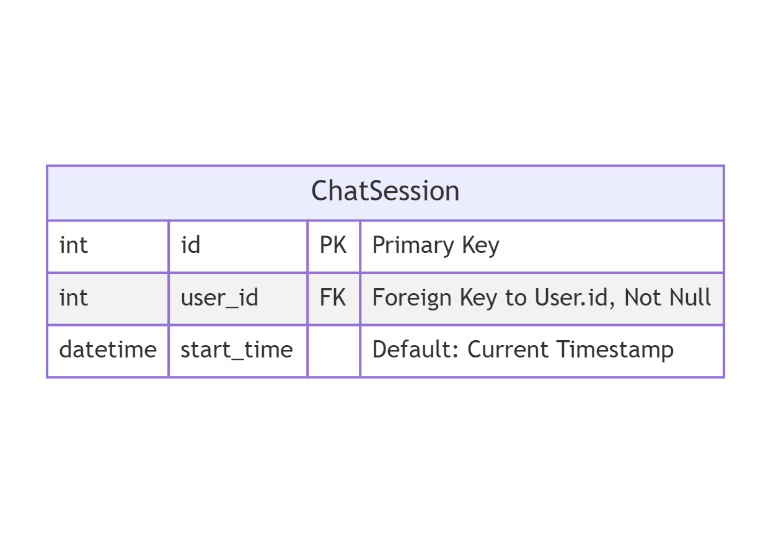


Figure 23 Showing **ChatSession Model**

* **Conversation Model**: The conversation model Captures individual messages from within a chat session. It includes a foreign key to **ChatSession**, establishing a one-to-many relationship between a chat session. This allows the application to organize messages under their respective chat sessions. Each message has a **type** to distinguish between user and AI messages, supporting the interactive nature of the chat. The **timestamp** field captures the exact time each message was sent, which is crucial for displaying messages in the correct order and context.

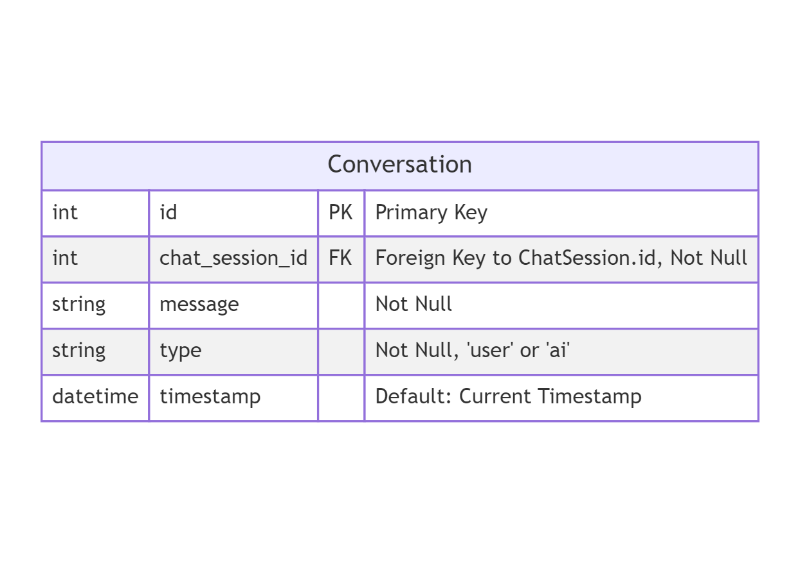


Figure 24 Showing **Conversation Model**

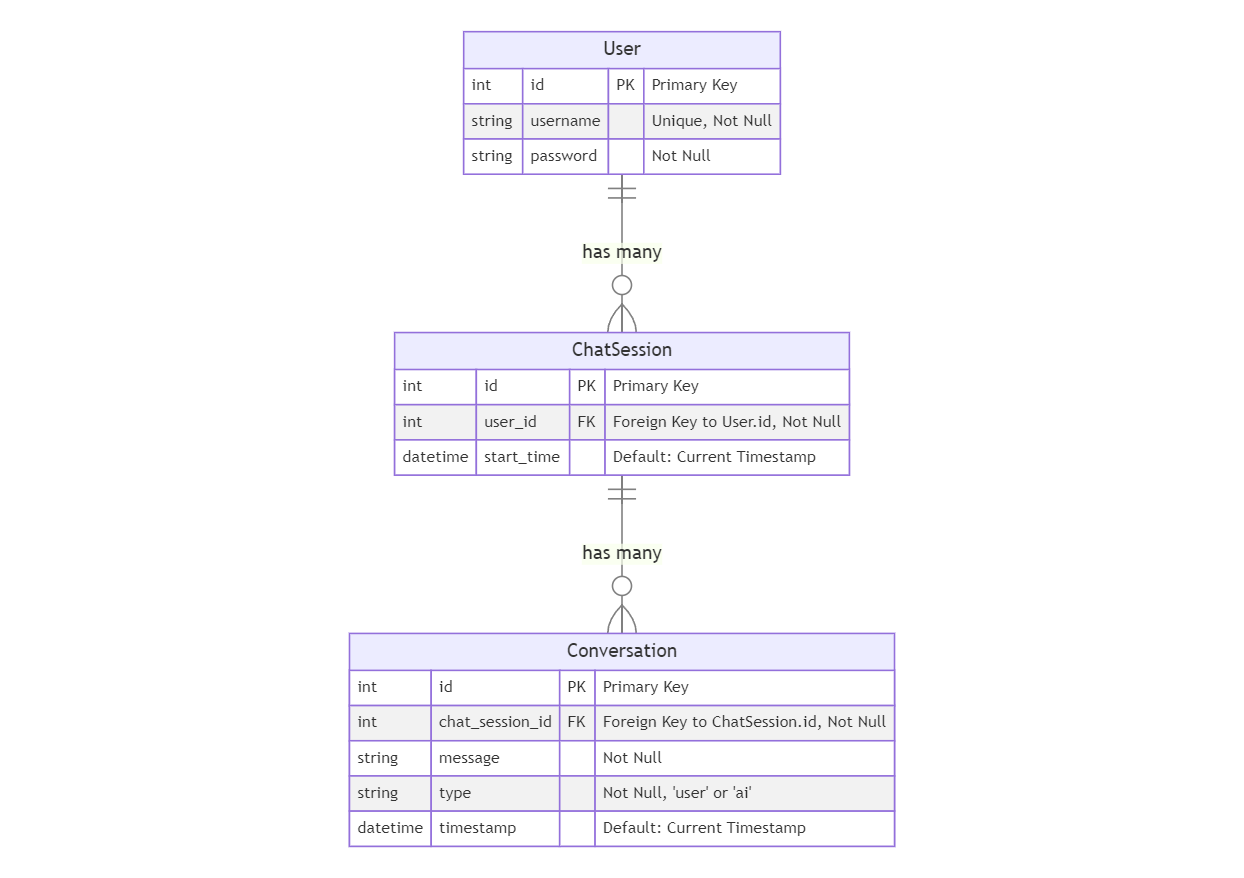


Figure 25 Showing the ERD Diagram for Litigat8 Database

## Dataset Workflow/Design:

The proposed dataset will be constructed Pdfs/Text documents available publicly for household and tenant law which include public articles or research papers etc. All of the entities are publicly available to refer and the model would just analyze the text in these to reference later. And they would be put through various data preprocessing steps such as Text Lowercasing, punctuation removal and Tokenization etc. To ensure the efficiency of the dataset and to remove the noise from the dataset. This would increase the overall efficiency of the NLP model to process data.

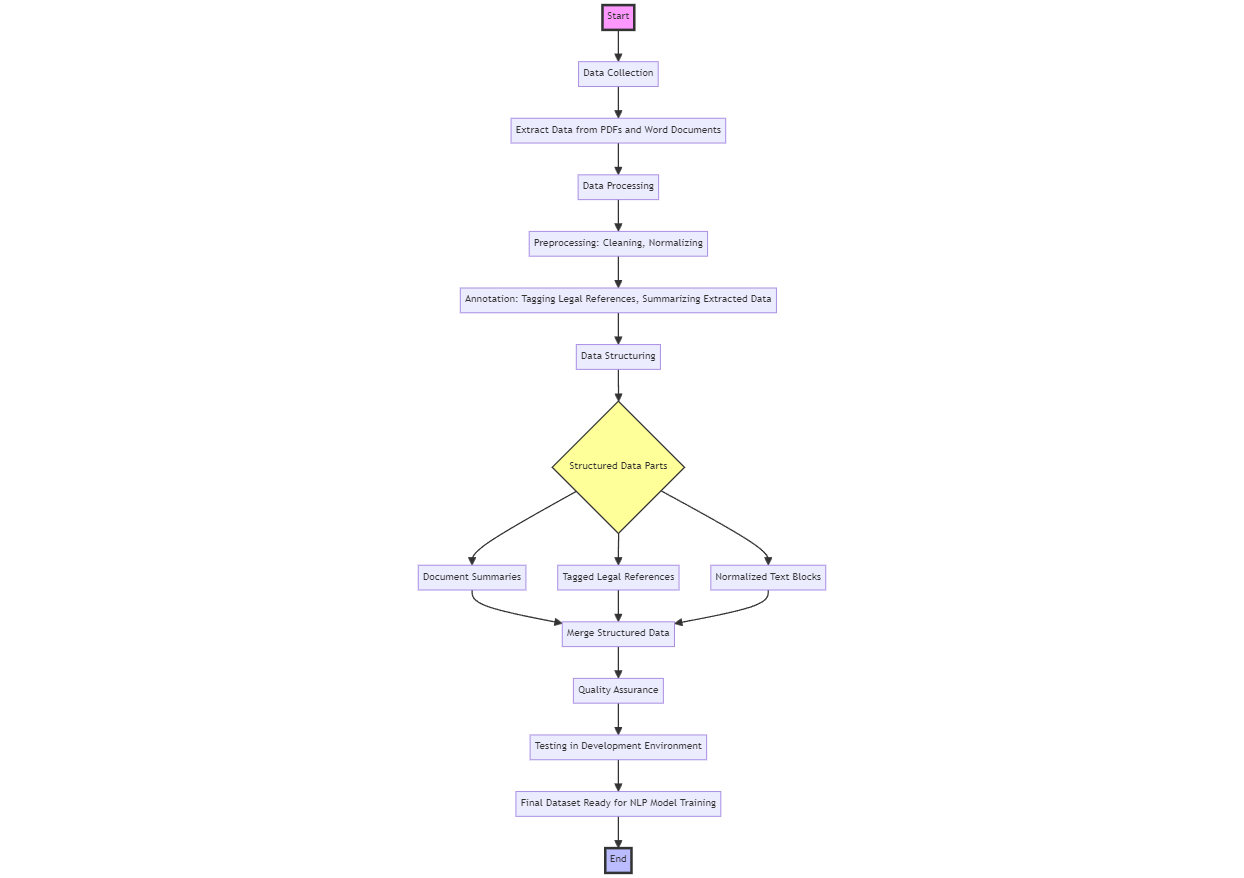


Figure 26 Showing the Framework for the Data Collection and Preparation for Litigat8 Training

The data that would be used would be Analysed before hand to make sure it is from a valid source and it is publicly available. And all the documents/ articles that would be used will be reference appropriately.

## LLD Design:

The proposed LLD Design shows a comprehensive technical view of the system which was modelled after doing the requirement analysis of the project.

The LLD Diagram for the project solution was implemented for detailing the implementation scheme of the application as LLD provide a blueprint of how every feature and component within the system would be implemented. The displayed LLD shows the exact programming constructs, algorithm, interface design and data models. There are few reasons why the LLD was proposed the most important one being that LLD ensures good code quality. As it helped me during the development to adhere to the agreed set of standards and maintain consistency. It also enabled me to reduce risks and uncertainty in the project as everything was modelled before the implementation of the program which helped during the project to mitigate risks.



Figure 27 Showing the Low-Level Design (LLD) for Litigat8

With the LLD it the main functionality of the applicaition can be identified with ease. The front-end would handle all the requests from the User in a structured way making use of JavaScript and AJAX. Ensuring that the information is fed properly from the front-end to the backend on specific routes. The backend is designed to handle all the POST and GET requests that are initialized. The authentication will be handled by programming structure labeled as **auth.py** in the diagram which would communicate with the MySQL database to validate the user details and redirect to the main dashboard page. The **Models.py** would be responsible for making the database tables of Users, ChatSession and Conversation models. The programming structure defined as **chat\_handler.py** would be responsible for handing the requests and functionality related to the chat with the NLP model i.e., saving the chat session, generating responses from the **nlp\_engine.py** file and handling all the POST and GET requests generated in the meanwhile. The **nlp\_engine.py** would handle all the logic related to the model training and preparation and would be responsible for generating the responses based on the user queries.

# Implementation:

## Database Implementation and Connection:

The database was set up using SQLalchemy which is a library inside of python. This enabled me to make the dataset dynamically by just making the models of each entity.

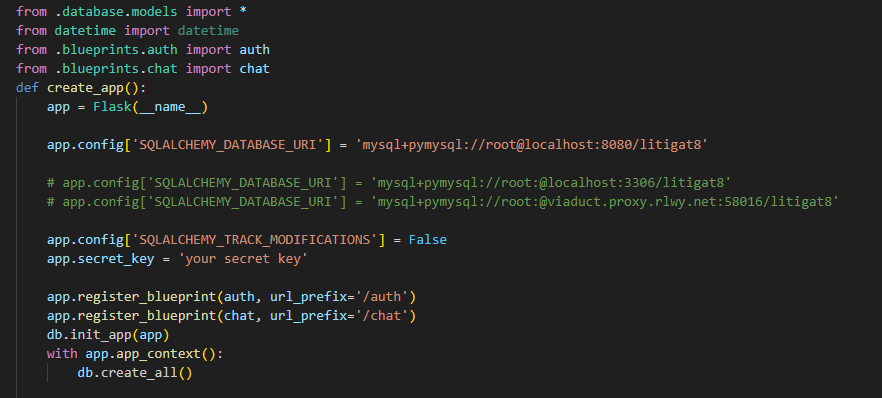


Figure 28 Showing the Setup of Database Config

With the database initiation its assigned to the flask application by setting up the correct URL, username and password for the database inside of the SQL database. The variable **db** can be used to access any of the model i.e User, ChatSession and Conversation. Making the overall access of the database really easy instead of using SQL statements running directly on MySQL.

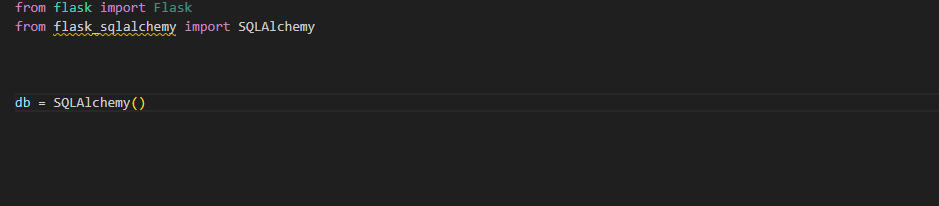


Figure 29 Setting up the Database

The models set up are made on accordance with the database schema which is displayed in the Figure 25. All of them have a primary key but ChatSession and Conversation have foreign keys and one-to-many relationship with the User\_ID.



Figure 30 Setting up the Models for Database

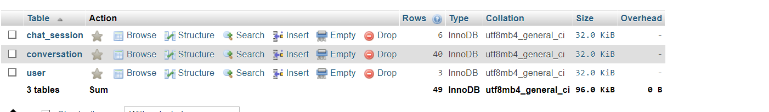


Figure 31 Showing the constructed Table in the Litigate Database

The chapter concludes the setting up of the database for Litigat8 application.

## Front-end Implementation:

**Main Home Page:**

The main page is designed based on the initial approach of keeping the user interface as intuitive as possible while keeping the functionality of the page. The initial design was designed using wireframes to make sure the it has all the essential components are present in the page.

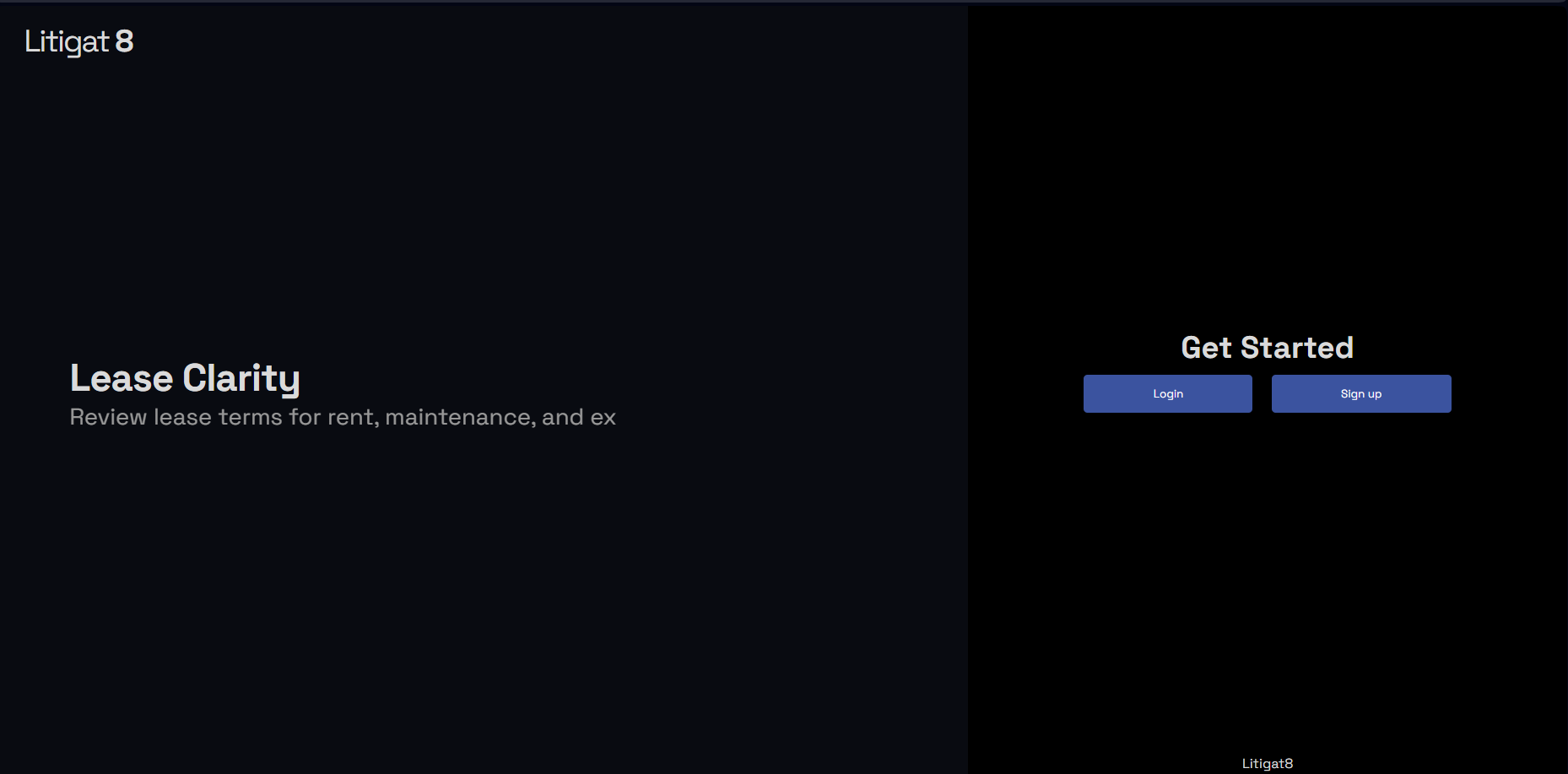


Figure 32 Showing the Front-end Implementation of Main-Page

An additional feature of typewriting was implemented as well as well to give it a bit more dynamic look.



**Login And Sign-Up Page:**

Following on the intuitive and approachable design of the pages, the design was based on the wireframes that was designed earlier. With two input field in login page and three input field in resistration page asking for the desired input from the users.

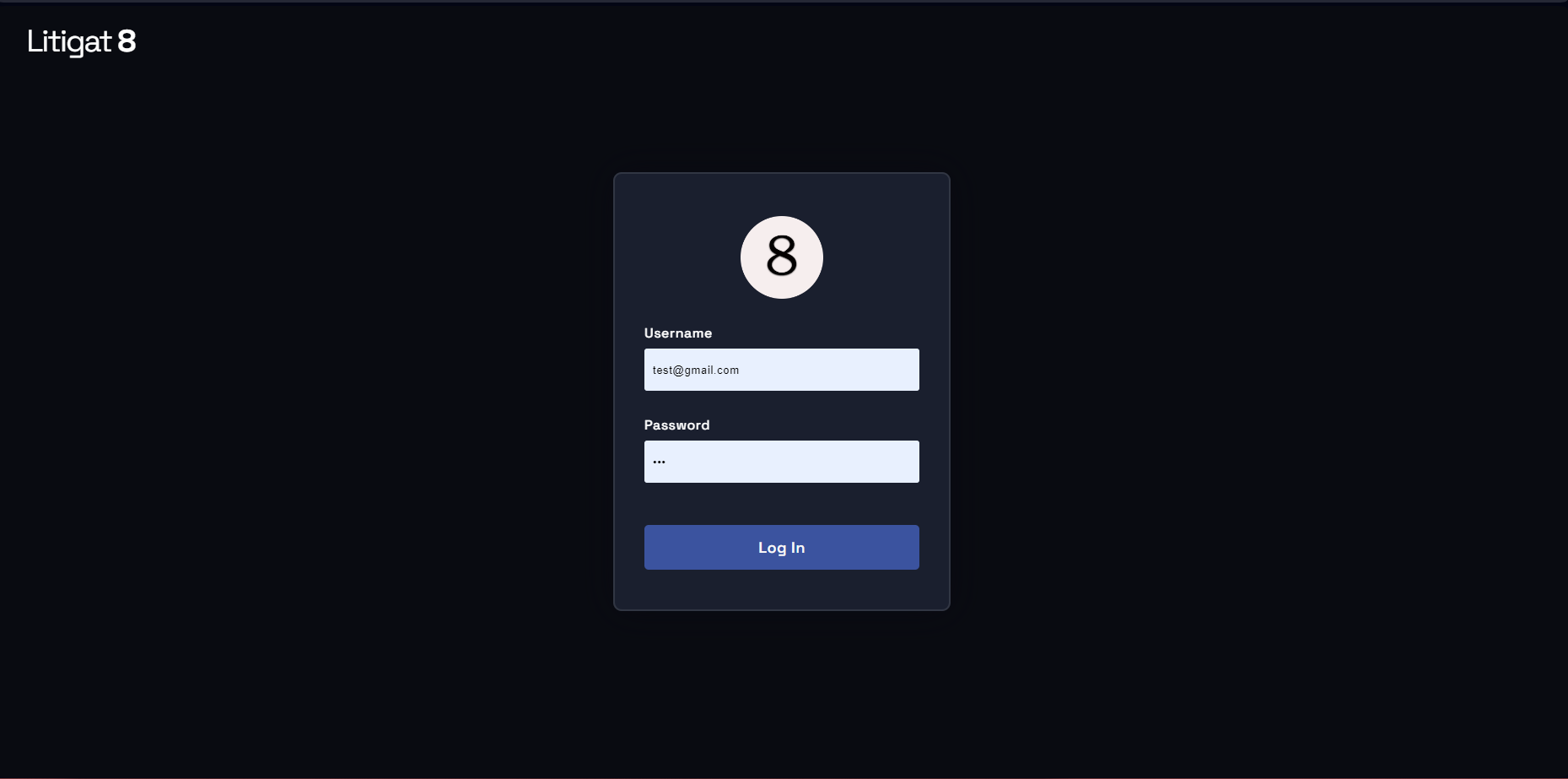


Figure 33 Showing the Front-end Implementation of Login Page of Litigate



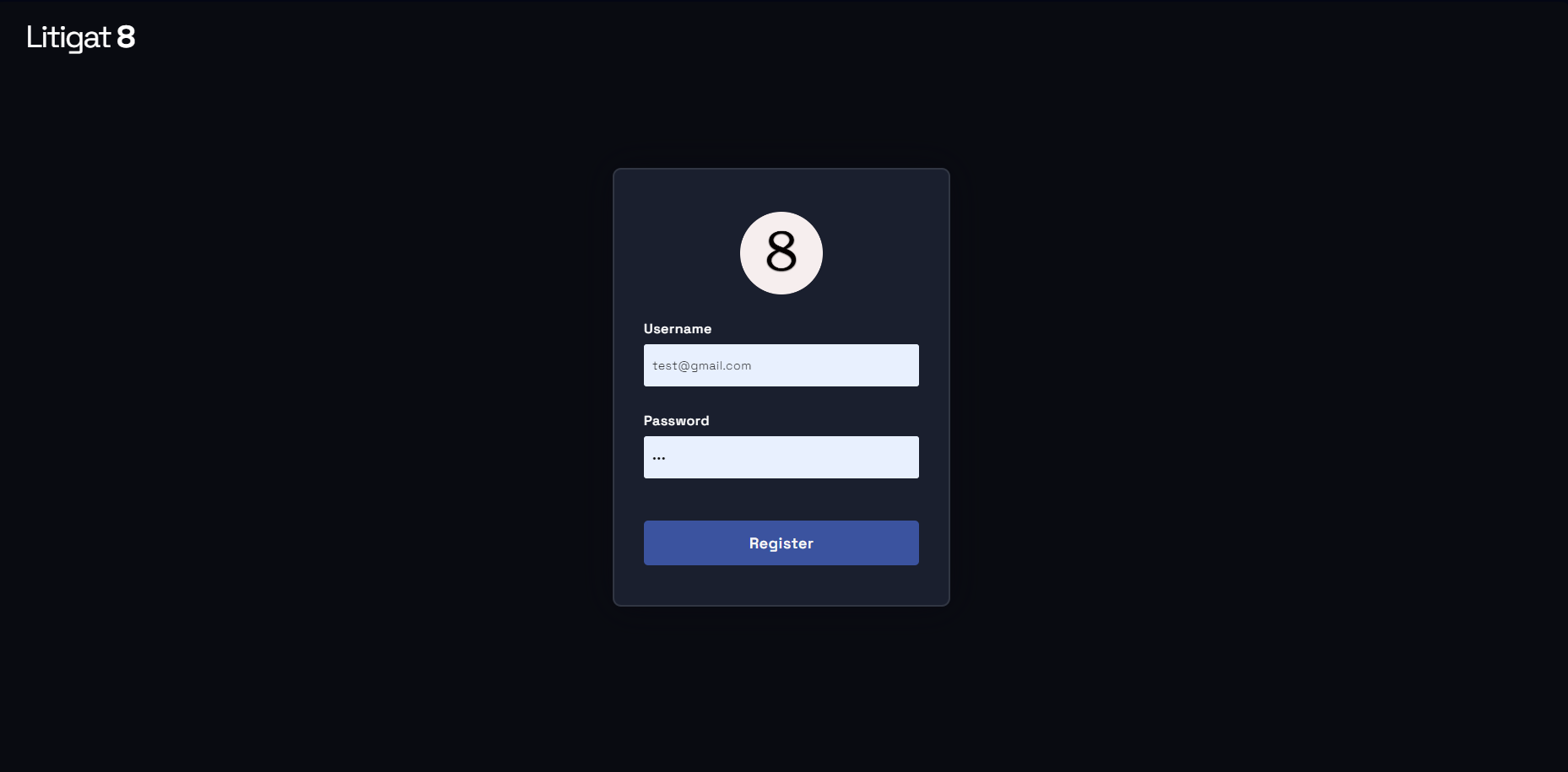


Figure 34 Showing the Front-end Implementation of Register Page of Litigate



**Chat Interface:**

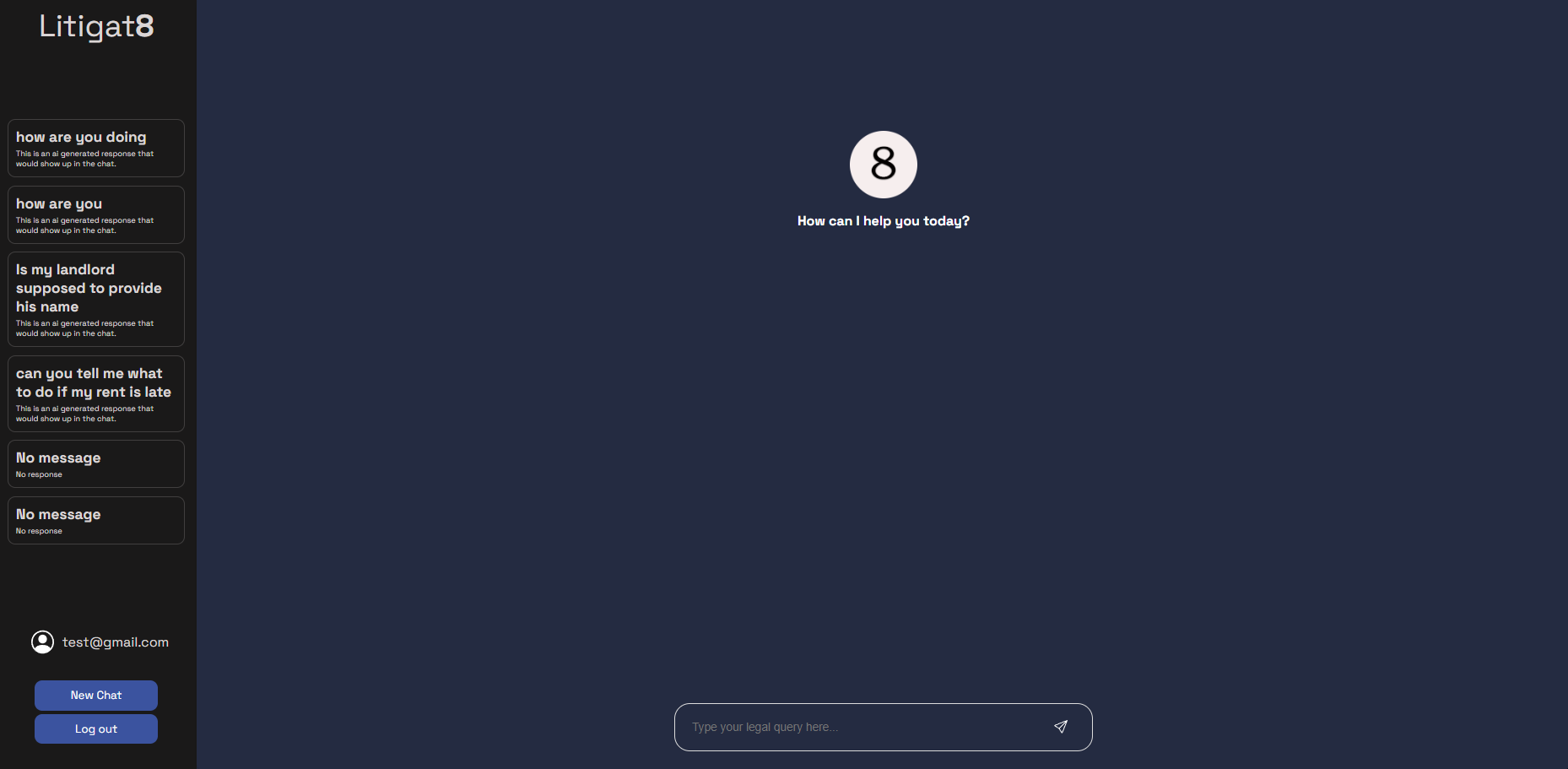
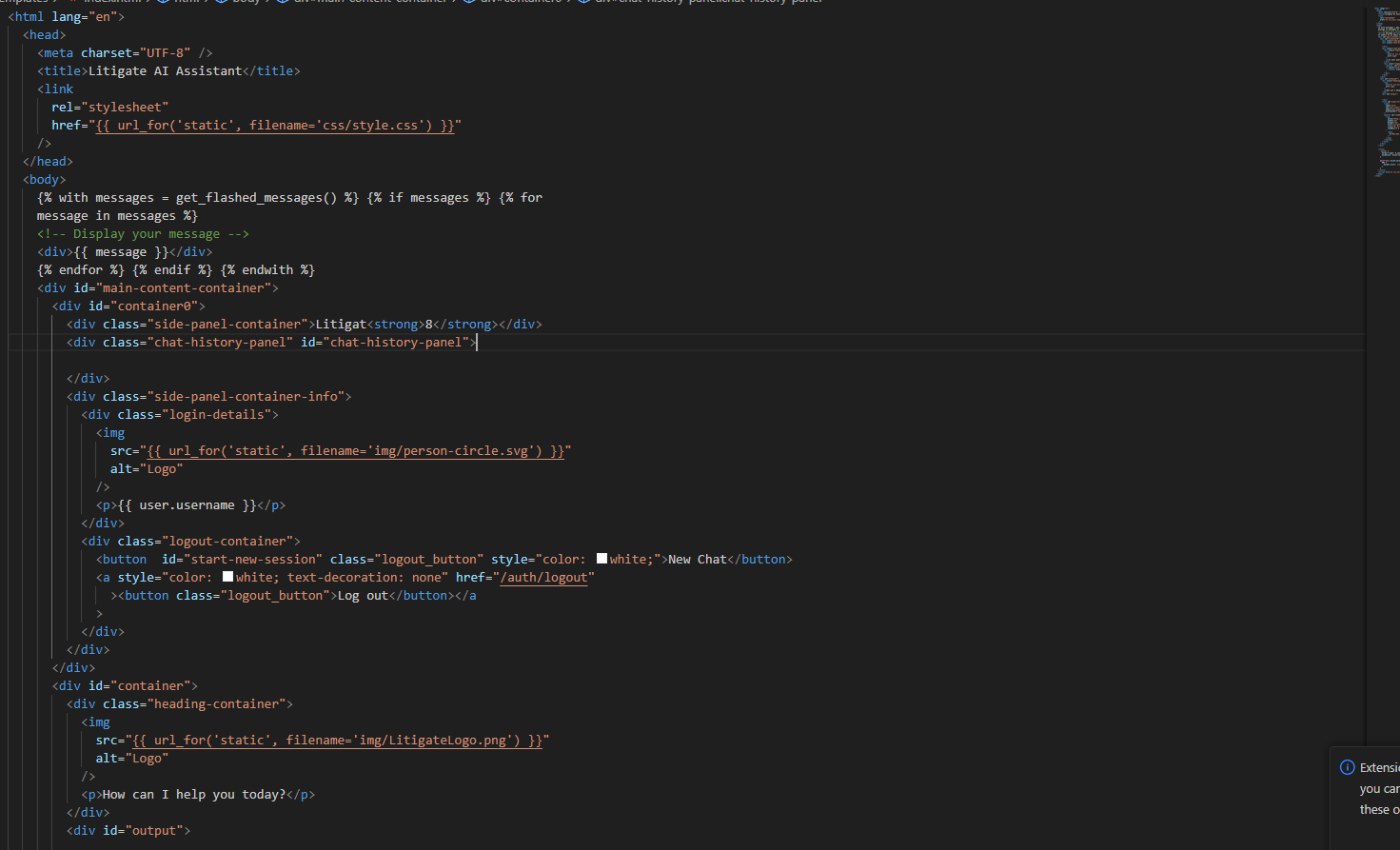


Figure 35 Showing the Front-end Implementation Chat Interface of Litigate

The main chat interface was build using combination of JavaScript, Html and CSS. The page followed the initial requirements set up by wireframes. With all the essential component present in the design. An input form to take in the User queries, a panel to display all the previous chat session with two buttons to either logout of the system or to start a new chat session.



This Marks the end of the front-end design implementation of the Litigat8. All the essential design elements have been put into place for the backend element to work on and make it dynamic. Moving to the next chapter the report would show the implementation of the backend of the system.

## Backend-Server Side:

For the implementation of the backend of the Litigat8 app various libraries were used such as **flask**, **SQLalchmey**, **functools**, **werkzeugsafe** etc. which worked in coordination to provide the implementation of the application based on the plans that were designed in the design phase of the project.

**Setting Up the App:**

The python flask application was set in the \_\_init\_\_.py file which would let you run the application using a simple **flask run** command in the CLI. The functionalities of the app was made modular and important functions stored in different files. All the necessary imports were made in the **\_\_init\_\_.py** file before setting up the app. The **creat\_app()** func sets up the application for us assimilating all the the routes and database configureation.

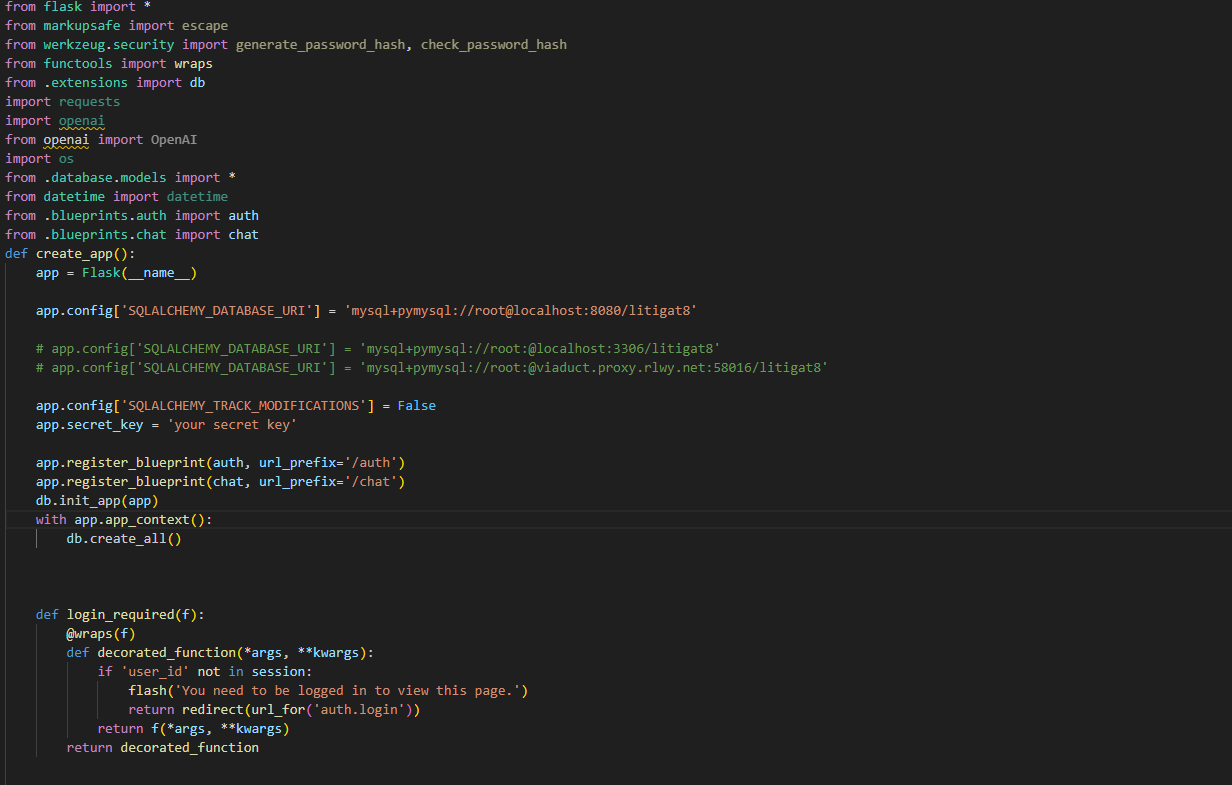


Figure 36 Showing the implementation of Flask App with Configuration

For setting up the routes for the flask application instead of setting it up directly in the app. Blueprints were used to set up the routes respective to their functionality. Two main blueprints were set **chat.py** and **auth.py** the **chat.py** handles all the routes to handle the server side functionality of chat functions linked with the specific routes. Whereas **auth.py** handles the logic used for authentication and registration of the user.

**Setting Up Blueprints:**

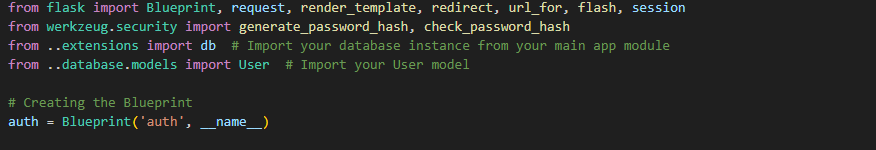


Figure 37 Showing the Implementation of Auth Blueprint

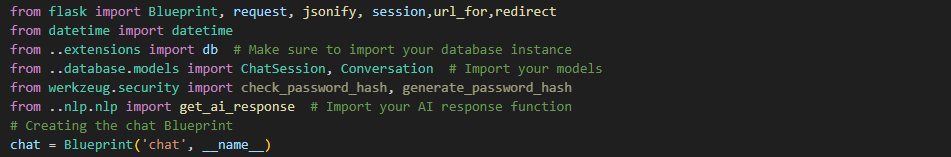


Figure 38 Showing the Implementation of Chat Blueprint

**Authentication Handling:**

The authentication of the User were done by using POST requests that are originated from the front-end. As a result of the POST request the function queries the database to check if the user exists or not and handles the results accordingly to the outcome.

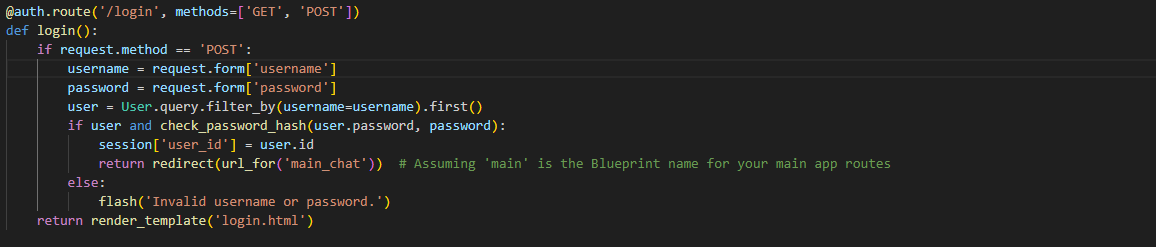


Figure 39 Showing the setting up of Login Function Server Point

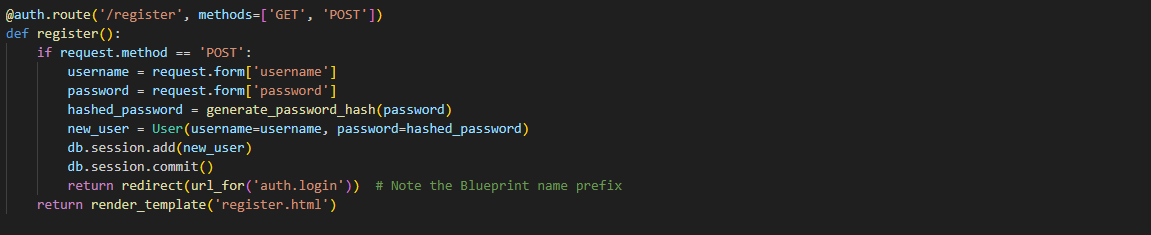


Figure 40 Showing the setting up of Register Function Server Point

Registration is done with a simple POST request as well to the URL **auth/register/**  which on being called inserts the User details into the database.

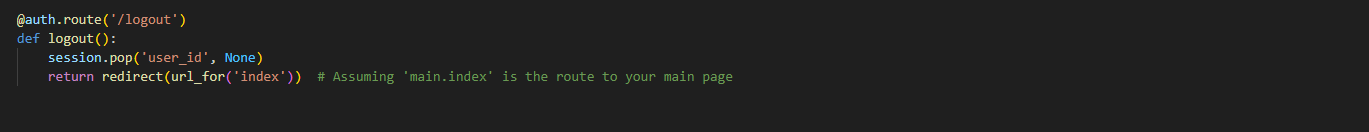


Figure 41Showing the setting up of Logout Function Server Point

The logout simply just pops out the user details if they exist in the flask application.

**Chat/Response Handling Generation and Chat Session Handling:**

**Python Server Side to Handle Requests:**

This section shows the implementation of how chat session are stored and retrieved. The **start\_chat\_session()** take in both kind of requests POST and GET. The function works by making Chat Session entry whenever the function is call and then the chat session id is stored in the session to be used later.

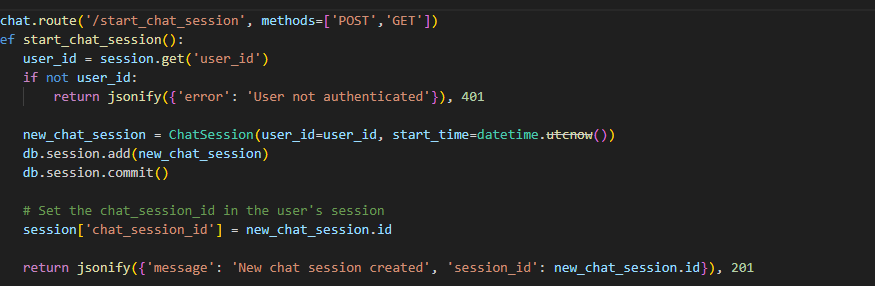


Figure 42 Showing the implementation of start\_chat\_session Server Point

The **/end\_chat\_session** works just by removing the chat\_session\_id from the session thus making not accessible anymore.

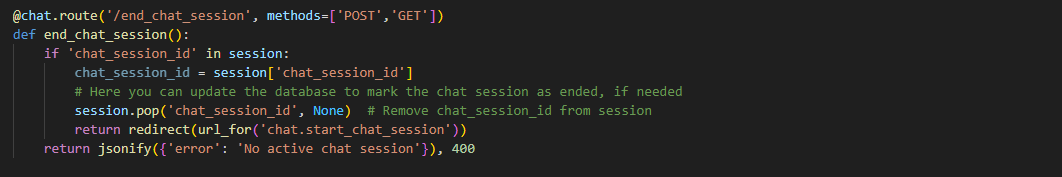


Figure 43 Showing the implementation of end\_chat\_session Server Point

The **/chat/submit** route is linked with **submit()**  function that calls onto the **ai\_response() func**  which is imported from **app.nlp.nlp\_engine** which serves as the base function for generating the AI response which then gets send to the font-end as JSON object to be printed out in a user understandable form. Using JSON to transfer data comes in really handy given the flexibility it provides.

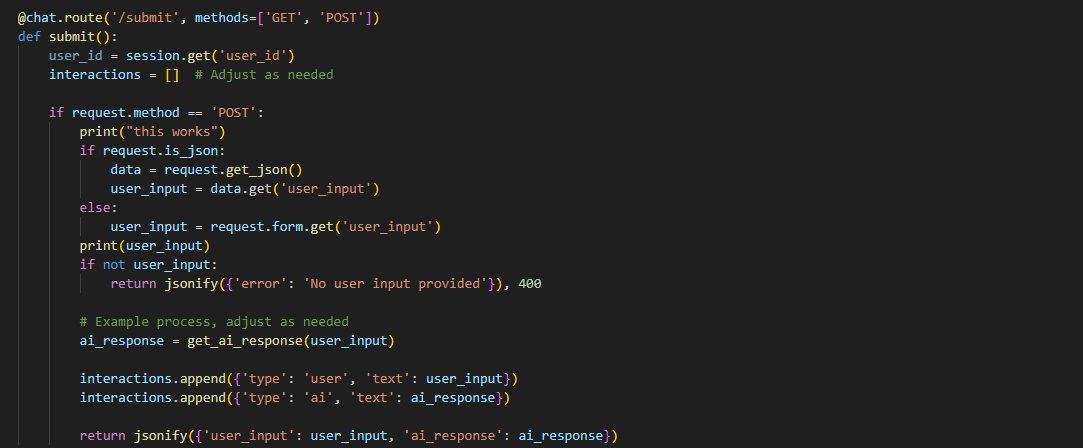


Figure 44 Showing the implementation of server handling of /submit route

The function **save\_interaction()** makes use of a helper function **save\_conversation \_message()** they both work to gather to save the interaction between the User and the AI to be referred later to be stored with a reference to the chat session. And the other function **get\_conversation/session\_id** was designed to get all the conversation stored in the database for a specific id. Which then are displayed on the front-end in a structured way.

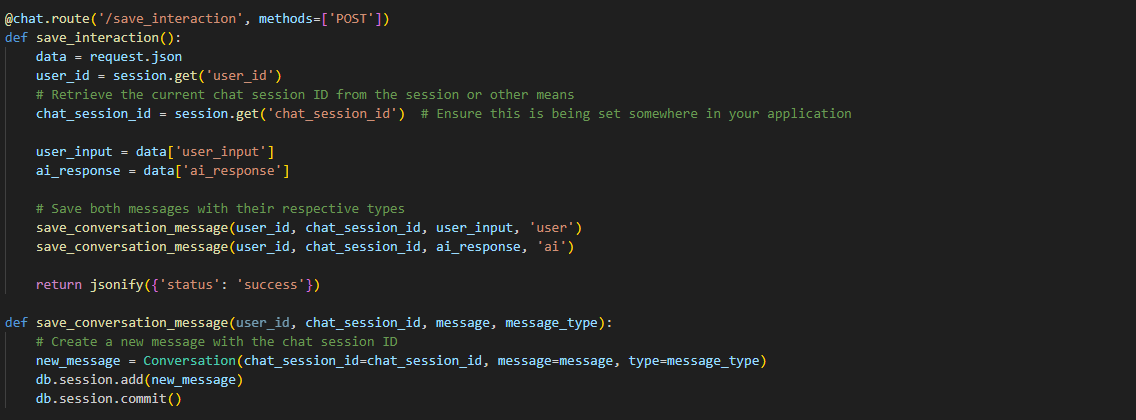


Figure 45 Showing the implementation of server handling of /save\_interaction route

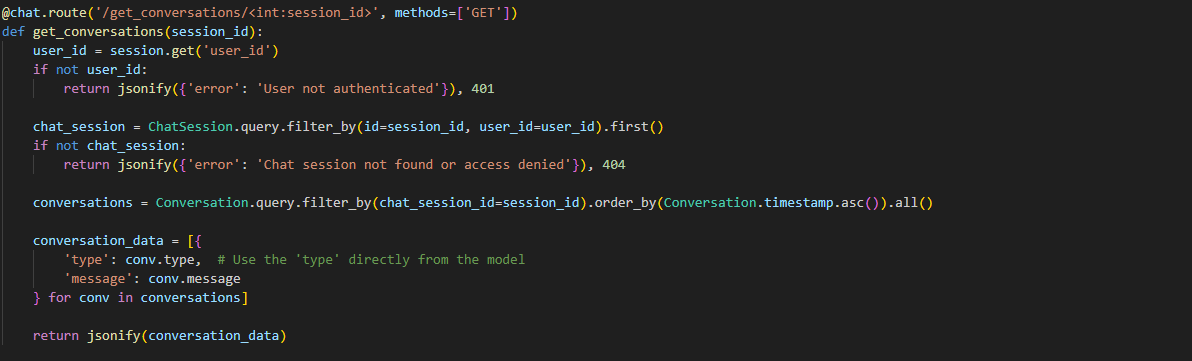


Figure 46 Showing the implementation of server handling of /get\_conversation/session\_id route

The loading of all the chat session to be displayed for the user was done by the use of **fetch\_chat\_histories()**  function which gets all the conversation enteries between the user and ai which have the same chat session id. It is then converted into a json object which then gets displayed at the front-end with the help of Javascript.



Figure 48 Showing the implementation of server handling of /fetch\_chat\_histories route

**JavaScript to Initiate fetch Requests to the Server Endpoint and Handle the JSON responses dynamically:**

This section marks the implementation of JavaScript that’s implemented in the front-end to which send fetch requests to server side to get the required data from the database and display it to the user.



Figure 49 Showing the implementation of Fetch Requests of chat/fetch\_chat\_history

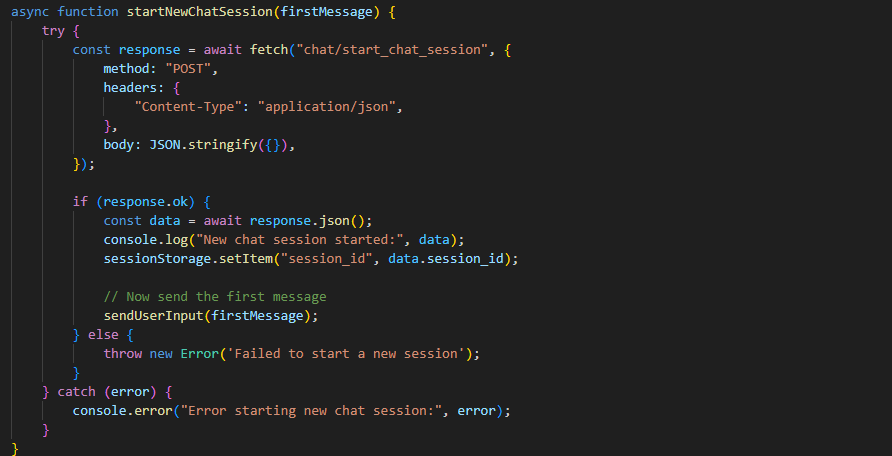


Figure 50 Showing the implementation of Fetch Requests of chat/start\_chat\_session



Figure 51 Showing the implementation of Fetch Requests of chat/submit

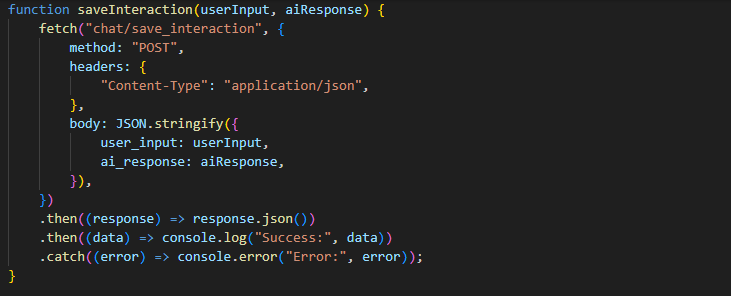


Figure 52 Showing the implementation of Fetch Requests of chat/save\_interaction

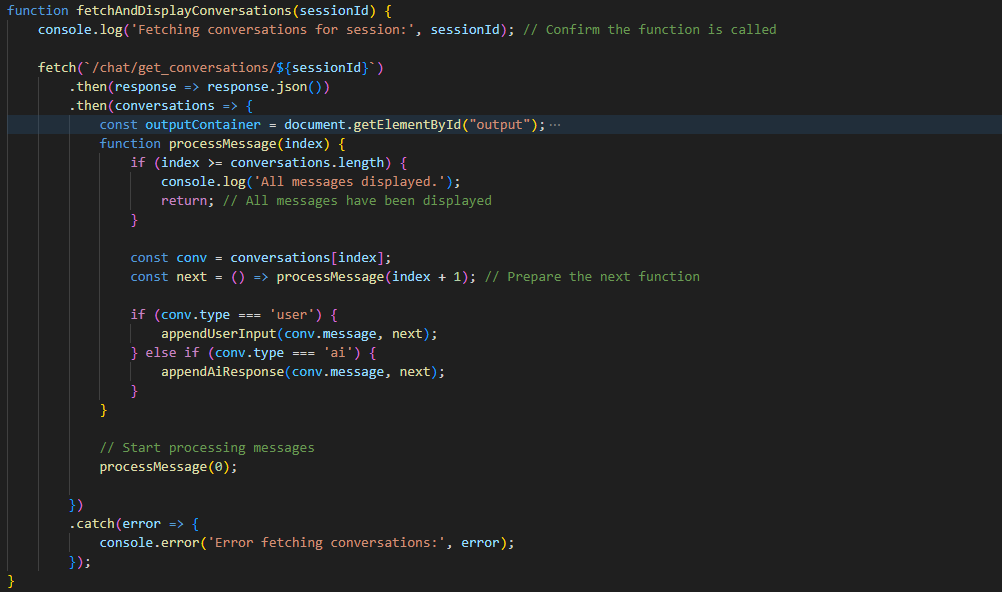


Figure 53 Showing the implementation of Fetch Requests of chat/get\_conversations/sessionID and Processing the of JSON load to front-end

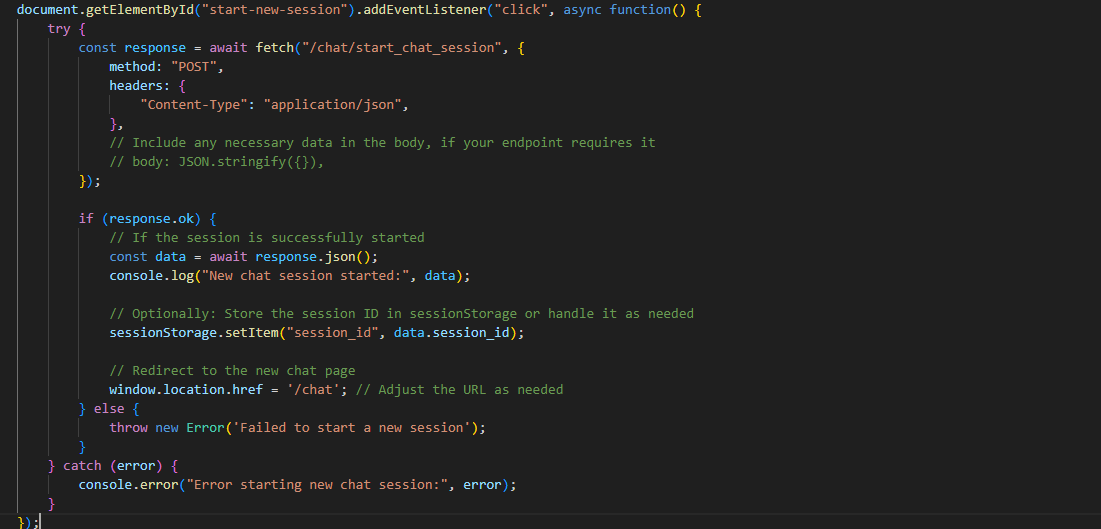


Figure 54 Showing the implementation of Fetch Requests of chat/start\_chat\_session

## NLP Model Implementation:

This chapter marks the implementation of the NLP Model. The steps that were taken to make a functioning LLM that would answer the User queries about household and tenant law would be discussed in this chapter. Many libraries were used to make the final proof-of-concept model for the application such as Langchain, OpenAI, Chroma, OpenAI, NLTK etc. These libraries served the foundation on which the application was build.

**Data Preprocessing/Text Preprocessing:**

Before starting with the development of the model, it was essential that the data is preprocessed before any work was done using it thus many preprocessing steps were taken to ensure there aren’t any discrepancies in the data. Four major preprocessing steps were taken , the first one being tokenization of the text. Which essentially breaks down the text into smaller units called tokens. There are different types of tokenization’s word tokenization, character tokenization etc. but in my case I decided to use word tokenization which is based on Penn Treeback tokenization because of the fact it can handle complex cases, such as contractions(e.g., splitting "don't" into "do" and "n't") and special punctuation patterns, thanks to its underlying use of regular expressions and the Penn Tree Bank tokenization standards.

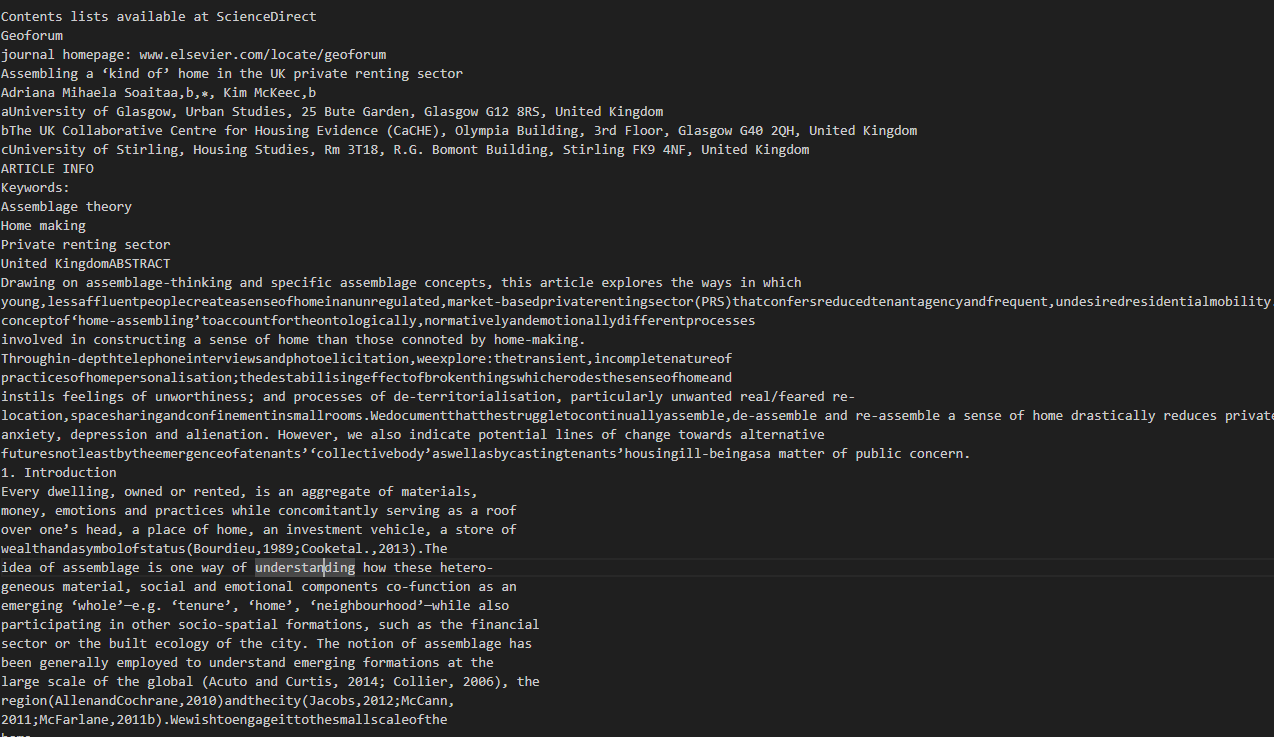
****

Figure 55 Showing the text data structure before pre-processing steps

The second pre-processing was lowercasing of all the words in a token. This made the whole text normalized which is essential to make sure there are not discrepancies in the corpus of the data.

The third pre-processing step that was taken was the removal of stop words in the case of English stop words like is,the,in don’t generally contribute to the meaning of a text for the analysis process thus they are removed from the dataset in the case of Litigat8 as well.



Figure 56 Showing the preprocessing steps i.e., Tokenization, Lowercasing, Removing Stop Words, Stemming

The 4th pre-processing step taken was stemming, which essentially breaks down the words to there root word for instance “Punctuation” would become “Punctuat” after applying stemming to it. There are number of benefits as it allowed to me to increase the relevance and search in information retrieval from the database.

****

Figure 57 Data Structure after Data Pre-Processing



Figure 58 Chunking/Spliting of the Text Data

The final step that was taken for the preprocessing was chunking of the data or splitting of the day of set size and of set character overlap. This help me increase the performance of the model as they have limit to input length of the tokens in case of the models that were used in Litigate from 512 to 4090. And the character overlapping helped me preserve the context of the tax and made sure it wasn’t lost.

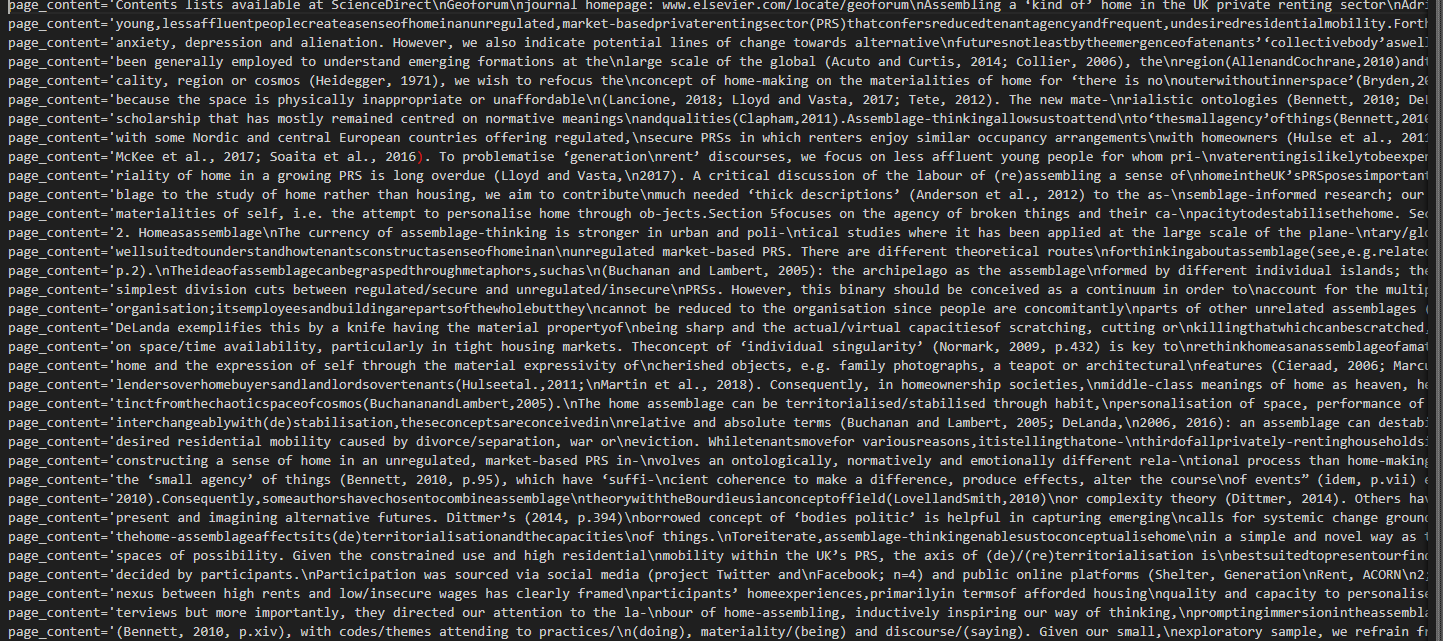


Figure 59 Data After Chunking/Splitting

**Embeddings Generation:**

The next step in sthe development was embeddings generation which are vector represetntation of text, words, phrases and entire documents. It maps the entities into a continuous multi-dimensional space where it reflects semantic relationships which exist in the space. There are number of reasons I employed this technique first was to establish semantic relationships between the words which improved my model’s similarity search functions. Which resulted in overall improvement of the model. Two different models were explored for making the embeddings, OpenAI embedding model **text-embedding-3-large** and hugging face **all-MiniLM-L6-v2,** they

Both were an excellent choice when generating the embeddings the only deciding factor to go with OpenAi was time it required to make the emdeddings which was relatively faster than hugging face model.



Figure 60 Showing the implementation of OpenAI Embedding Model

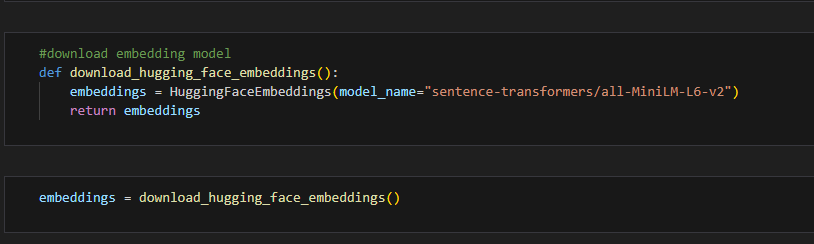


Figure 61 Showing the implementation of Hugging Face Embedding Model all-Mini-LM-L6-v2

**Vector Database Generation:**

The next step was to create vector database to store all the vectors that are created as a result of embeddings. The reason a vector database was employed was because of traditional database such as MySQL are not optimized to type of quereies machine learning applications require such as finding the nearest neighbors in a high dimensional space such as in the case of Litigat8. Setting up the vector database allowed me to do functions such as semantic search and dynamic content discovery.

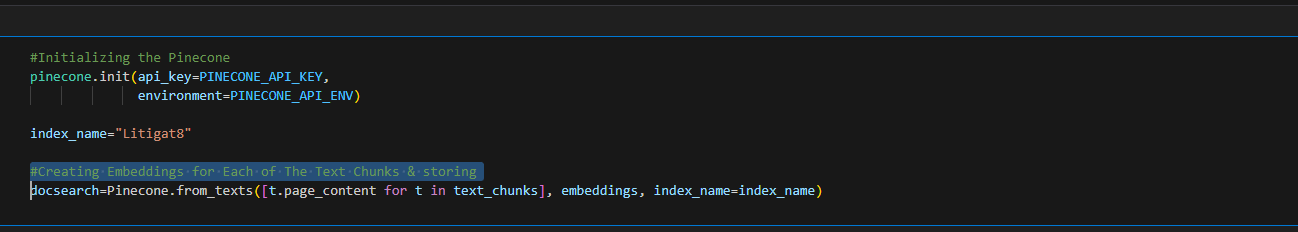


Figure 62 Showing the Implementation of Pinecone Vector Database

While making the vector database two different libraries were Analysed **Pinecone and Chroma**. Both of these libraries had there merits the Pincone database was an online vector database while chroma was a database on disk. Traditionally for larger project **Pinecone** would be the ideal choice and hence that option was explored first but due to recent development in the **Pinecone**  structure some of there methods and function have been depreciated while there were alternatives to that but the library that I was using for Langchain I was using wasn’t compatible for that. Ample time was given to make Pinecone work but due to not being enough resources available online. I wasn’t able to get it functioning with my model hence I decided to look for alternatives instead and thus decided to settle for an on disk database built using **Chroma.** Even though it might not be the ideal choice but for the scope of this application it provided enough value.

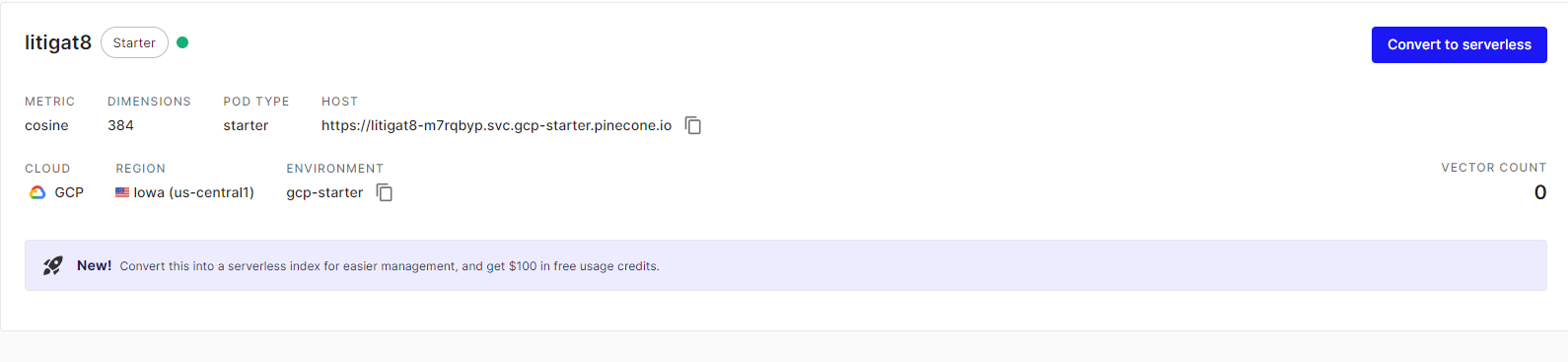


Figure 63 Showing the Setting Up of Pinecone Database on the Website

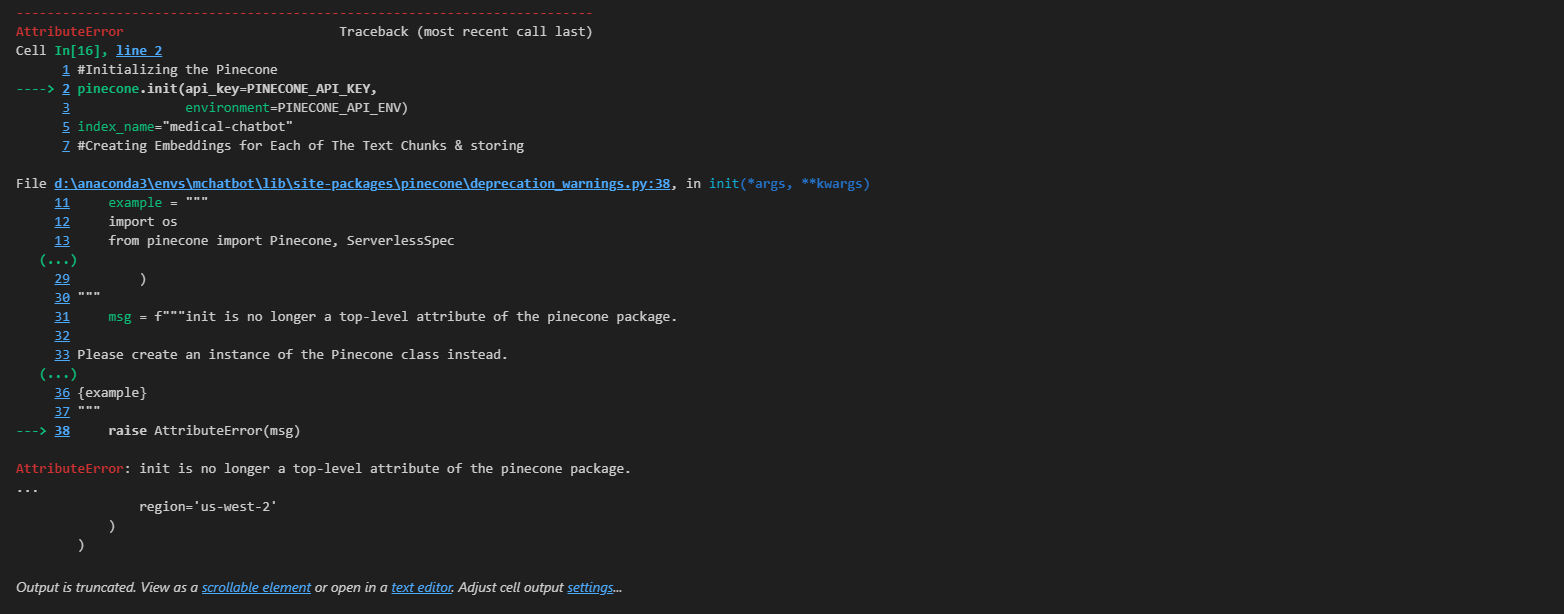


Figure 64 Showing the Depreciation Warning of Pinecone

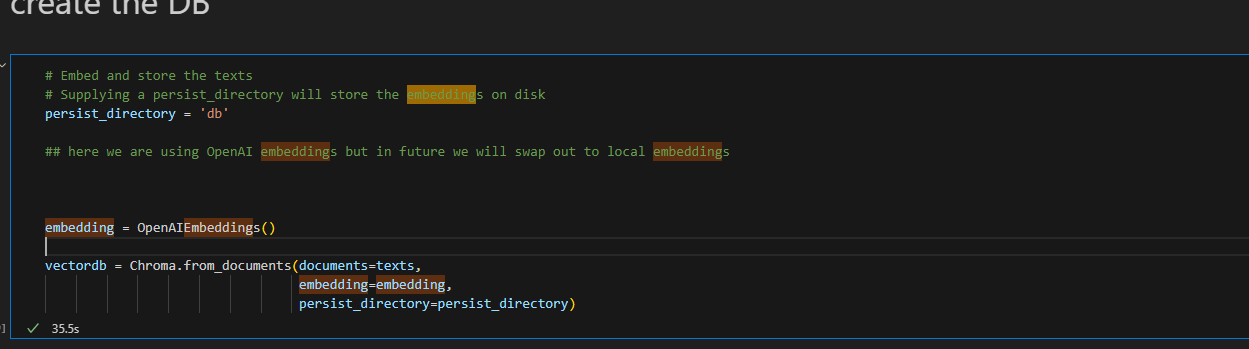


Figure 65 Showing the creation of Vector Database

**Making a Retriever:**

The retriever that was constructed from the database served as a way to query the database to get the specific piece of that that was required according to the query of the user.

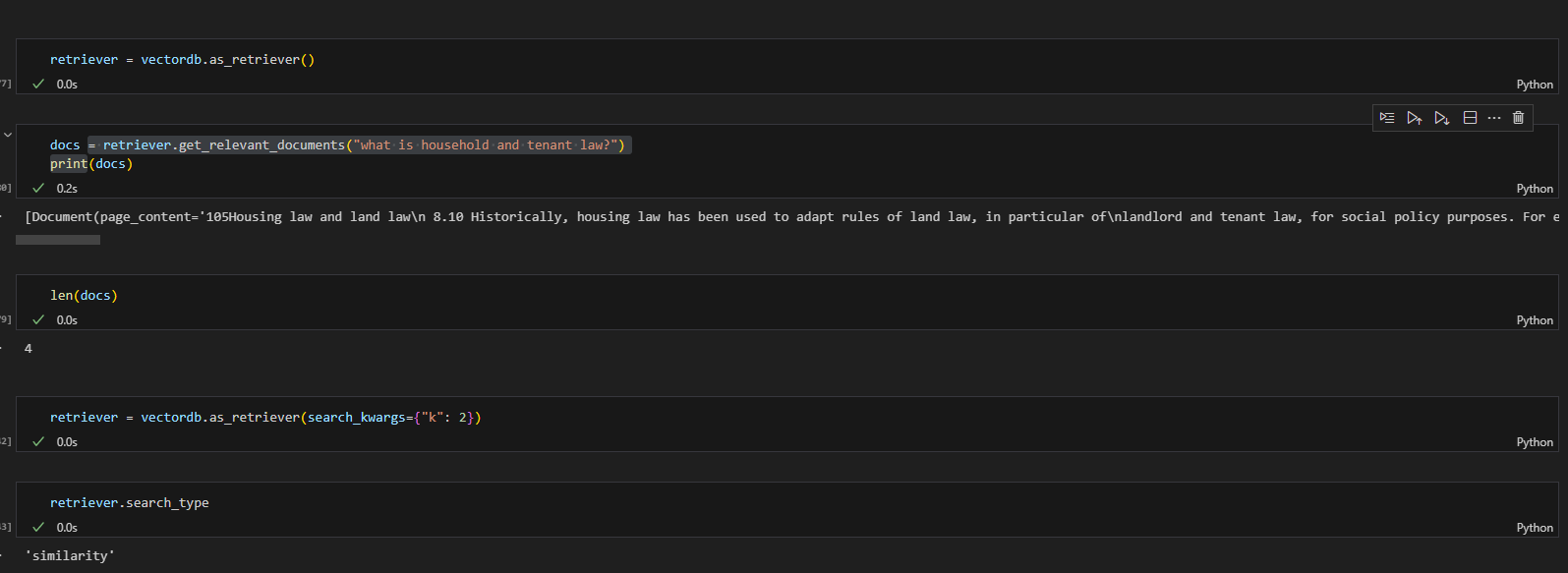


Figure 66 Showing the Implementation of VectorDb Retriever

**Making a Retrieval-based QA Chain for answering the question:**

The second last step in the development of the model was setting up the retrieval-based QA chain, which was done with the help of LangChain library. It helped me process the user question and understand key terms or entities. And enables me to retrieve information based on the processed user input and give that as a context to the model. And retriever was passed as a parameter as well which served as a way to get the information from the vector database.

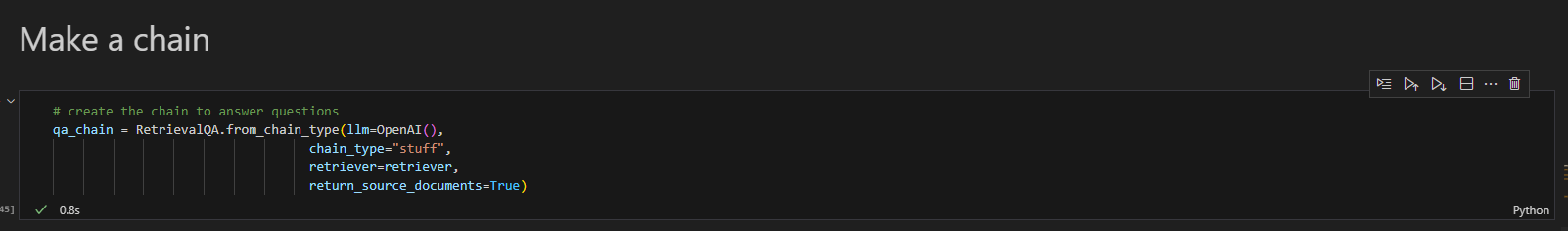


Figure 67 Showing the implementation of Retrieval-based QA Chain Using OpenAI() LLM

**Generating Reponses with Source Citation:**

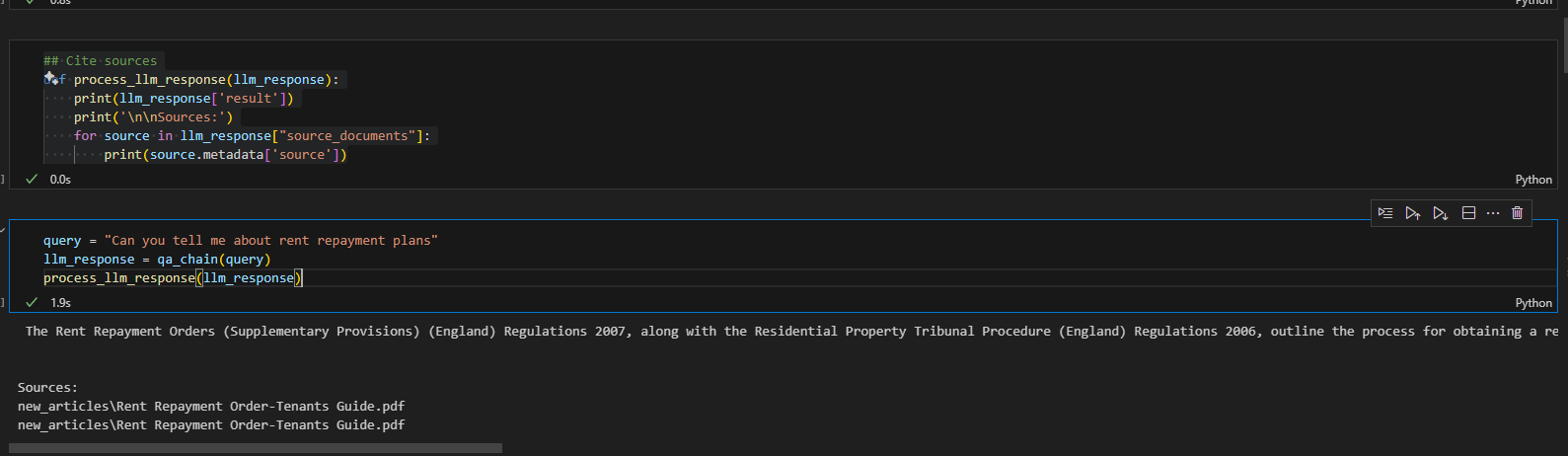


Figure 68 Showing the Response Generation with Source Citation

**Prompt Engineering:**

The final step of the model configuration was to set prompts for the Gen AI so that it doesn’t answer questions out of the context or it doesn’t answer question answer question which it doesn’t know answer for

**Making Sure the Generative AI doesn’t Answer Question Out of Context:**

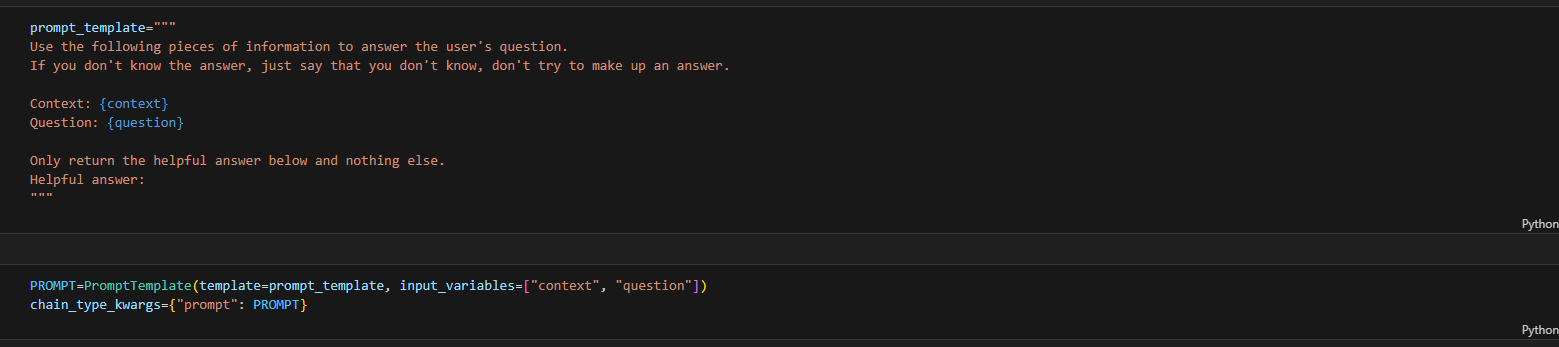


Figure 69 Setting up the context for the LLM

I played around the prompt settings to find the best prompt that would result in the best responses generated from the AI. Different versions of the prompts were created two of them are displayed in the Figure 71 and Figure 72.

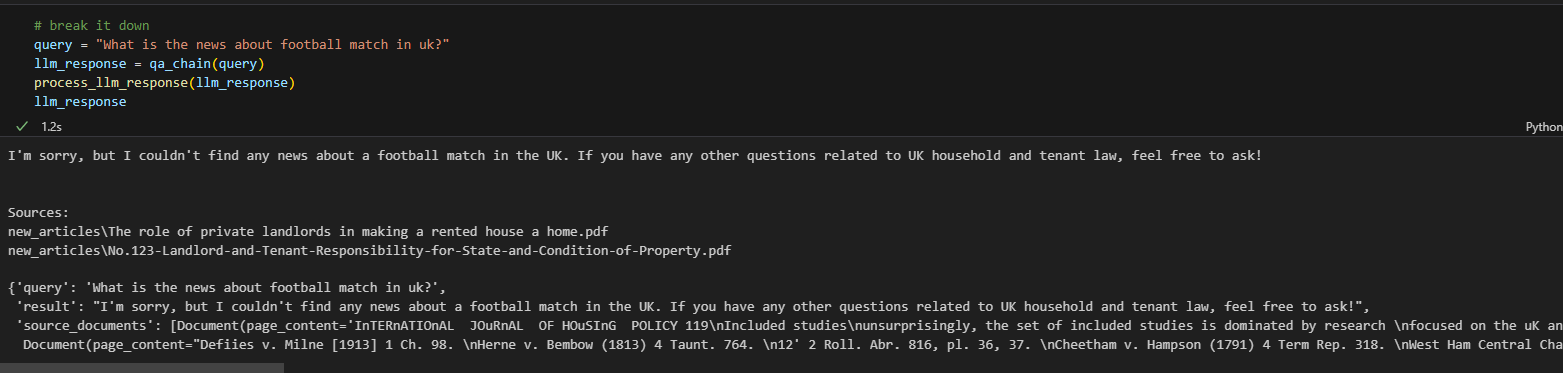


Figure 70 Showing the LLM answer to a Question out of Context

**Modifying the Prompt to get the Desired Outcome and Limiting the Gen AI:**

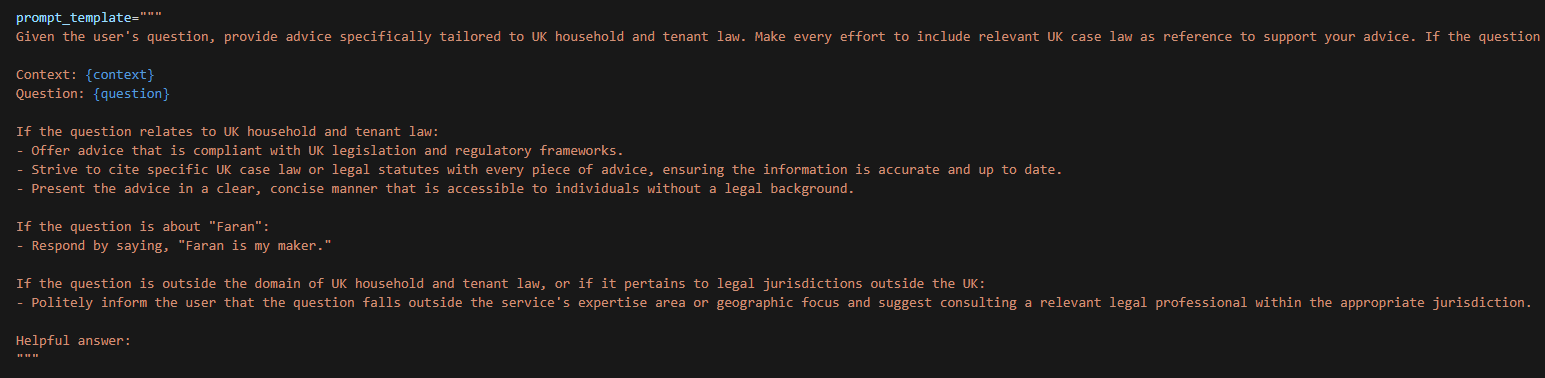


Figure 71 Prompt Engineering Version 1

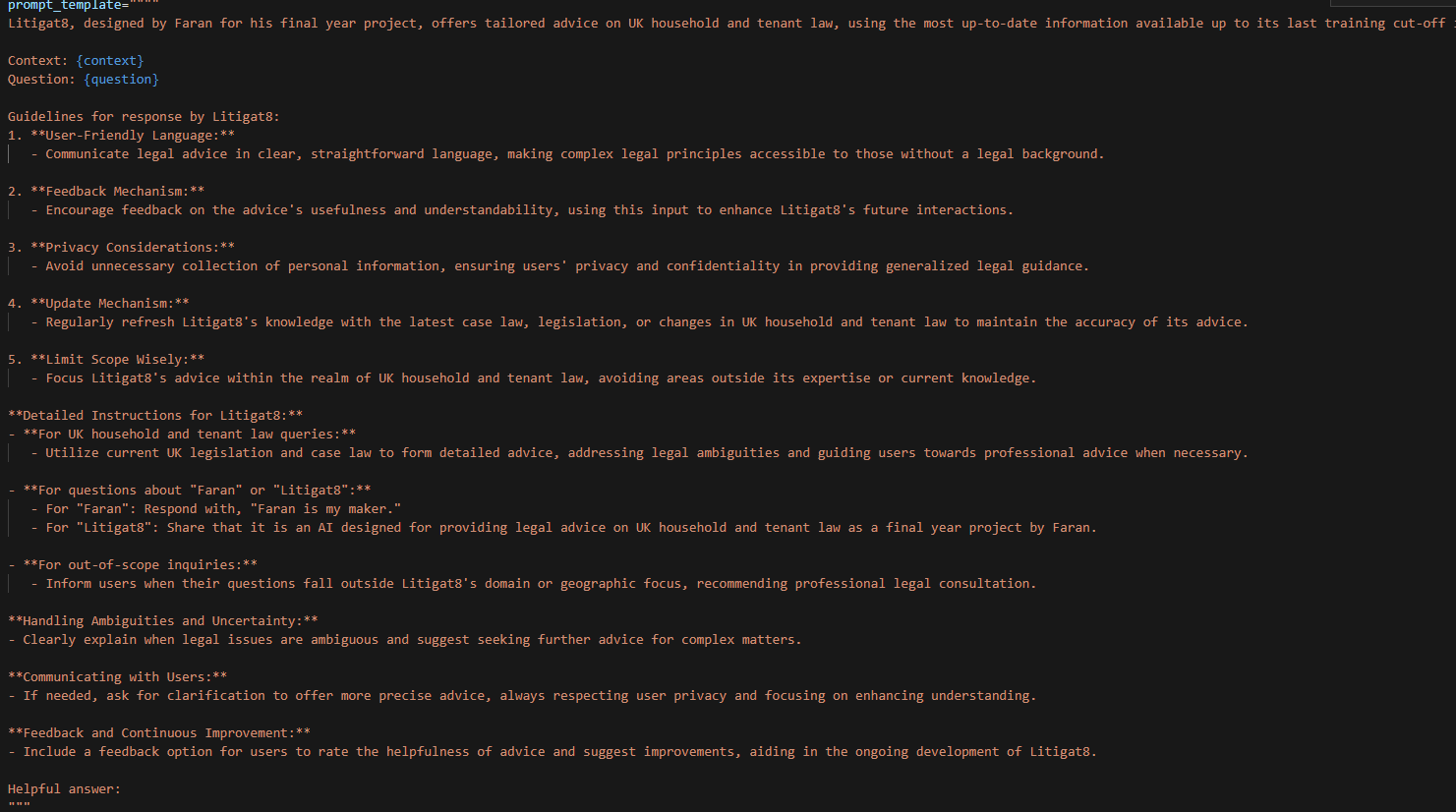


Figure 72 Prompt Engineering Version 2

**Comparison of ChatGPT 3.5 Turbo Vs ChatGPT 4.0 Turbo Vs Llama 2:**

The final step that was to be implemented was to do a comparative analysis between the LLM models that exits to finalize which would serve the function of litigate the best. The technical comparative analysis had two metrics which lead to the evaluation of the model the first one being the response time as one of the objectives were to display advices in real-time so it was of paramount the model should be able to generate responses according to that. The second metric was semantic similarity score of the response generate to figure out if the response generated has a meaning full context and semantic relation with the question that was asked by the user. In addition to that conversation coherence analysis was done was well which was carried out by myself to see how coherent the sentences are which model produces more meaningful and well structured sentences.

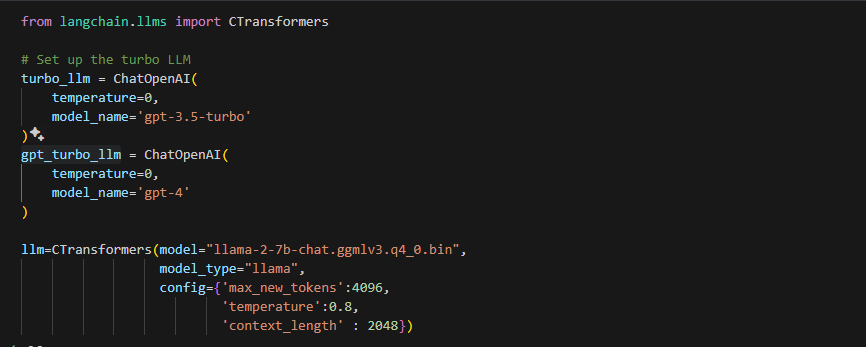


Figure showing the implementation of gpt-3.5-tubo , llama-2-7b Model , gpt-4.0

**Technical Analysis :Semantic Analysis and Response Time Comparison:**

The evaluation was carried out by setting up dummy questions and answer which were linked to specific prompts. And then the response generation time and the accuracy score (semantic score) were compared in the end for all three of the models.

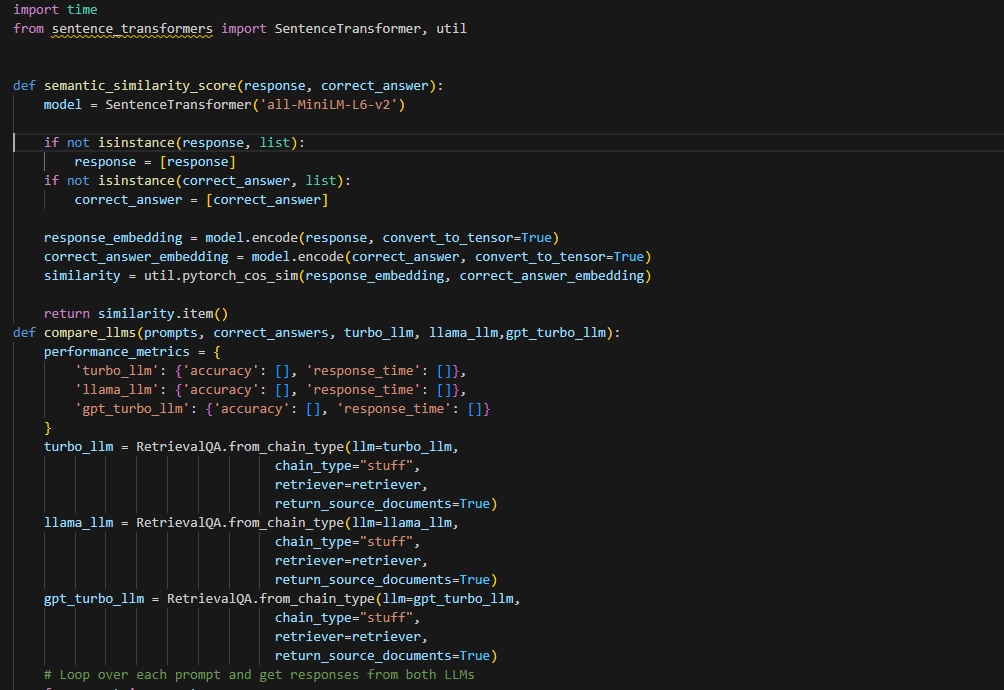


Figure 74 Setting Up the Semantic Similarity func and Comparison func

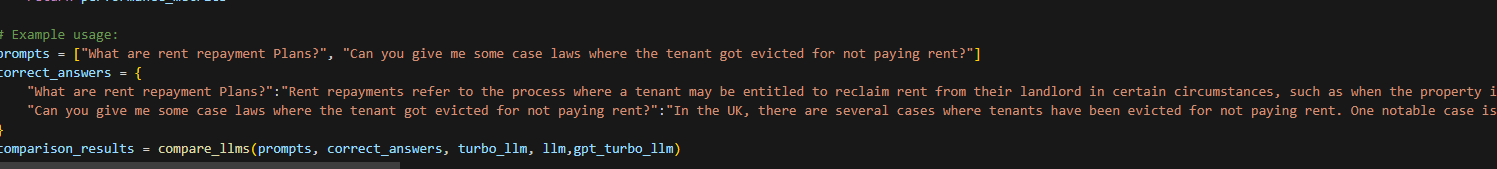


Figure 75 Setting up the Prompts and Correct Answers

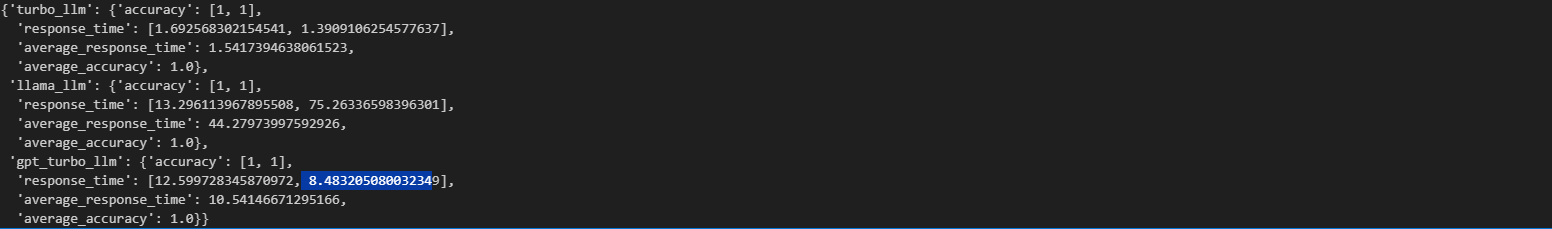


Figure 76 Showing the Comparison Result

**Conversational Coherence Analysis**

After the technical analysis, the models were compared on the bases of conversational coherence with a simple metric which response was more appealing to me as a user who’s seeking advice on a matter. For this purpose, a model question was decided on and was asked all three of the models and the responses were evaluated which would be discussed in the next chapter.

**Evaluation Sentence:** “What are rent repayments how can I pay them can you tell me?”

**LLAMA 2 Response time 2Min >**



Figure 77 Showing the response generated LLMA-2

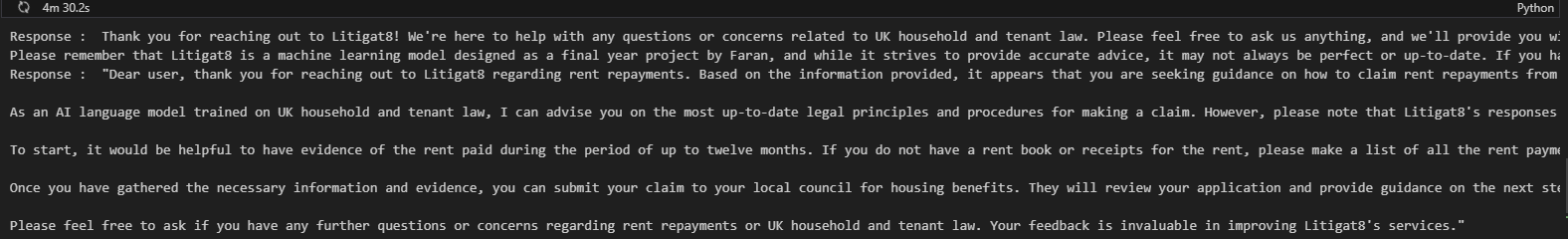


Figure 78 Showing the response generated LLMA-2

**GPT4 1Min >**

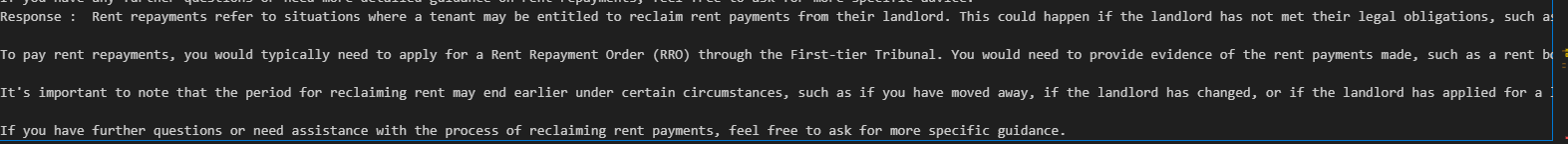


Figure 79 Showing the response Generated by gpt-4

**GPT 3.5 Turbo : 10 sec>:**



Figure 80 showing the response generated by gpt-3.

**Outcome Discussion**

Based on the results that were generated for the models **GPT-4**, **GPT-3.5-Turbo** and **LLAMA2**. When it comes to accuracy all three models performed as expected with having sematic relationship between the question asked and the response generated. Hence, they all passed on this evaluation. Moving on to the next evaluation, response time **GPT 3.5-turbo** was the fastest with response time ranging between 8 to 10 sec followed **by GPT-4** and then **LLAMA 2**. Hence **GPT 3.5-turbo** was the clear winner in the technical evaluation. Moving on to the Conversational Coherence Analysis **LLAMA2** and **GPT-4** provided with the best responses in the context of the data that was provided to them but **GPT-3.5-turbo** responses were accurate as well covering all the essential points to form a good response. As a result of all these observation GPT-3.5-turbo was selected as its stead true for all the metrics that were required for Litiagat8 to functions.

# 

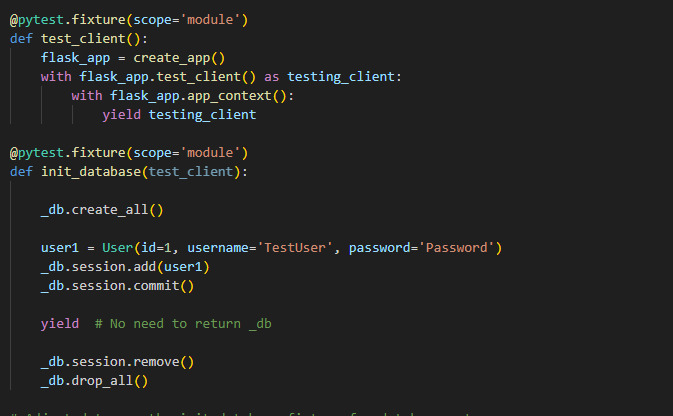
# Testing:

## Unit Testing

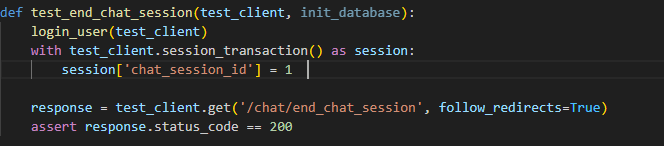


## Designing the Unit Tests Using Pytest:

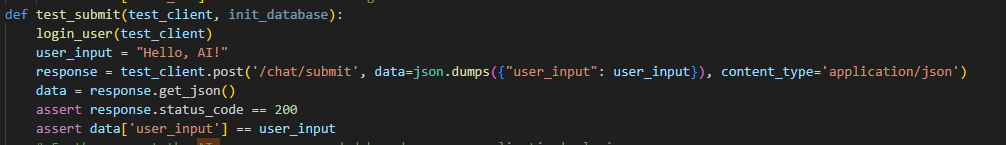
**Setting up the Test Environment and setting up the database before the Tests::**



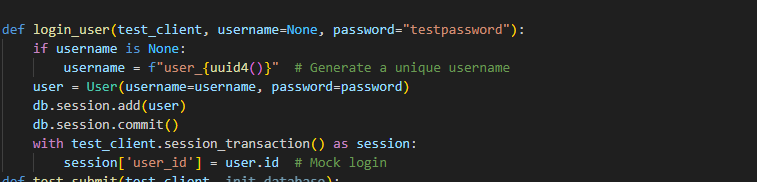
**Pytest for Ending Chat Session:**



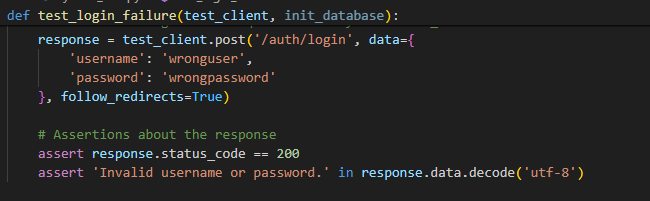
**Pytest for form submission and AI Response:**



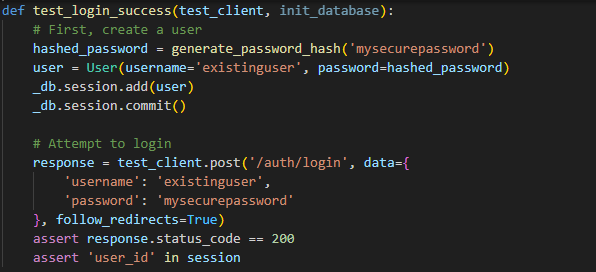
**Pytest Authentication/ Login:**



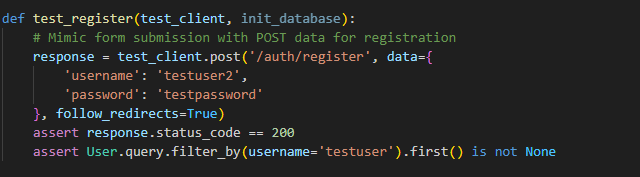
**Pytest Authentication/ Login Failure:**



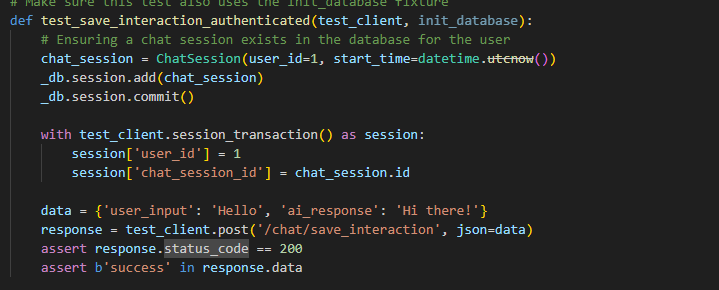
**Pytest Authentication/ Login Success:**



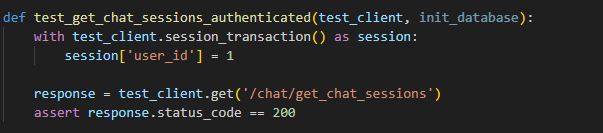
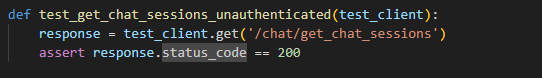
**Pytest Authentication/ Register:**



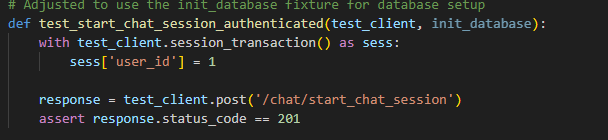
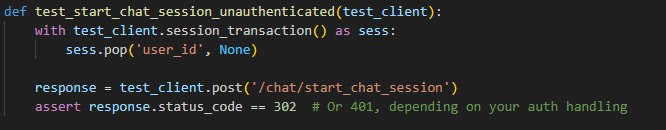
**Pytest Saving Interation of Users and ChatSession:**



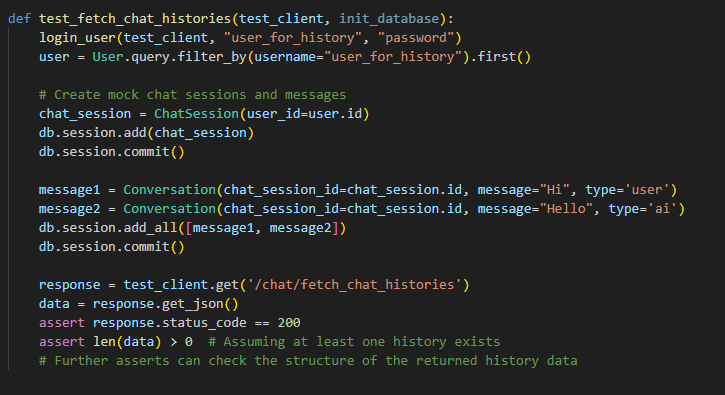
**Pytest to get chat sessions:**



**Pytest to start chat session:**



**Pytest get chat histories:**



## Unit Tests Outcome:

This chapter concludes the unit tests that were conducted for the Litigat8. The test results were all satisfactory with almost 80% complete success rate. With 2 test cases that resulted that resulted in warnings but they carried out the function they were supposed to. The two tests that resulted in warnings would be discussed in the next chapter.

| **ID** | **Component** | **Test Case** | **Input** | **Expected Output** | **Success Criteria** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- |
| **TC-001** | Start Chat Session (Authenticated) | User is logged in and starts a chat session | User session with user\_id set | HTTP status code 201, indicating creation of a new chat session | Test passes if the response status code is 201 | Pass |
| **TC-002** | Start Chat Session (Unauthenticated) | User is not logged in | No user\_id in session | HTTP redirect (302) or 401 Unauthorized, depending on auth handling | Test passes if the response is a redirect (302) or 401 | Pass |
| **TC-003** | Get Chat Sessions (Authenticated) | User is logged in and fetches chat sessions | User session with user\_id set | HTTP status code 200 and a list of chat sessions associated with the user | Test passes if the status code is 200 and the response contains the user's chat sessions | Pass |
| **TC-004** | Get Chat Sessions (Unauthenticated) | User is not logged in | No specific input needed | HTTP status code 200, | Test passes if the status code is 200 | Pass |
| **TC-005** | Save Interaction (Authenticated) | User is logged in and saves an interaction | User session with user\_id and chat\_session\_id set JSON payload with user\_input and ai\_response | HTTP status code 200 with a success message | Test passes if the interaction is successfully saved and the response contains 'success' | Pass |
| **TC-006** | Register User | New user registration | POST data with username and password | HTTP status code 200 and the new user is found in the database | Test passes if a new user is created and can be found in the database | Pass |
| **TC-007** | Login (Success) | Existing user logs in successfully | POST data with valid username and password | HTTP status code 200 and session contains user\_id, indicating successful login | Test passes if the user is logged in and session contains user\_id | Pass |
| **TC-008** | Login (Failure) | User fails to log in | POST data with invalid username and password | HTTP status code 200 with an 'Invalid username or password.' message | Test passes if the response contains the invalid login message | Pass |
| **TC-009** | Submit (Authenticated) | User submits input in a chat | JSON payload with user\_input, User session contains user\_id | HTTP status code 200 and the response JSON contains the same user\_input | Test passes if the response status is 200 and echoes user\_input | Pass |
| **TC-010** | End Chat Session (Authenticated) | User ends a chat session | User session with chat\_session\_id set | HTTP status code 200, chat session was successfully ended | Test passes if the chat session ends successfully | Pass |
| **TC-011** | Fetch Chat Histories (Authenticated) | User fetches their chat histories | User session contains user\_id | HTTP status code 200 and a JSON payload containing the user's chat histories | Test passes if the response contains the user's chat histories and status code is 200 | Pass |

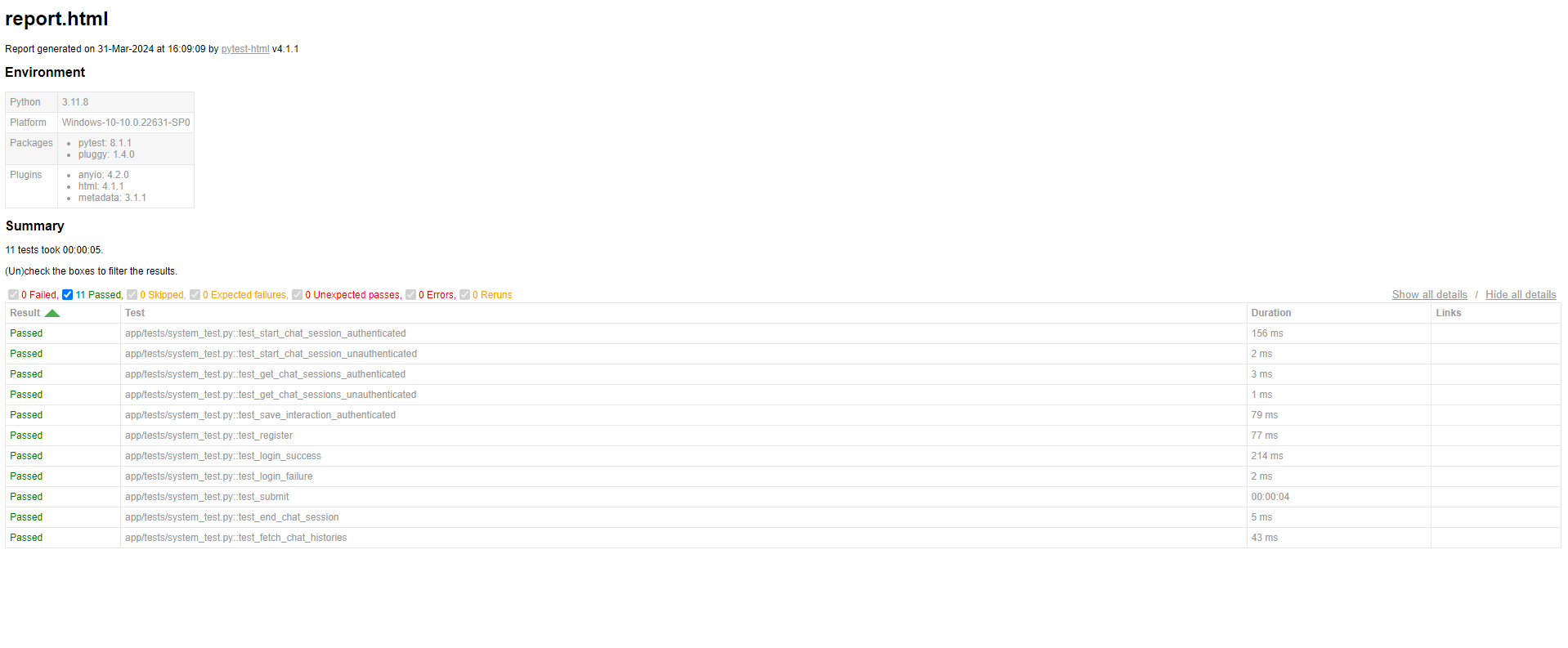


Figure 81 Showing the results of pytest for Unit Tests

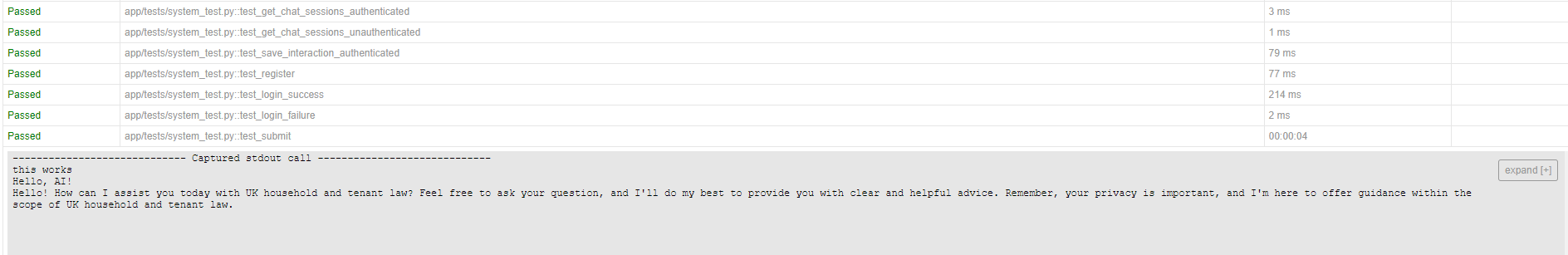


Figure 82 Showing the Unit Test for Interation with Litigat8 NLP Model

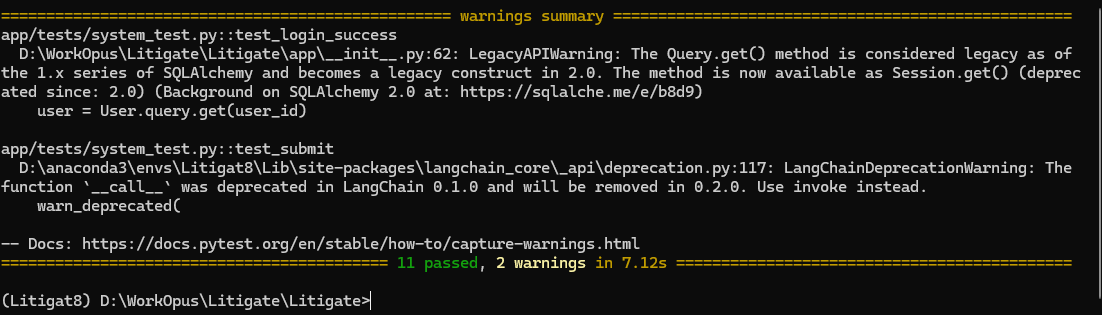


Figure 83 Showing Warnings Generated by the Tests

## Unit Tests Outcome Discussion:

**Component Login(Success) :**

**func test\_login\_success:** The warning was originated from SQLALchemy. Which warned me about the Query.get() method has depreciated in the version 2.0 of the library. And instead advised me to use Session.get() instead to resolve the warning and any other compatibility issues in the future.

As a result, the login functions was updated to use the newer version of the function thus eliminating the warning for the test\_login\_success function

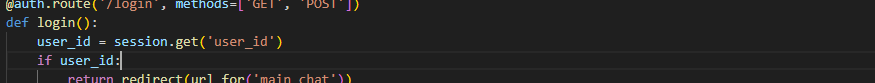


Figure 84 Showing the Fix of Login (Success) component warning

**Component Submit (Authenticated):**

**func test\_submit:**

The warning was originated from langchain\_core\_api warning about the deprecation of \_\_call\_\_ function within the version LangChain 0.1.0 and warning that it would be removed in 0.2.0. I wasn’t able to exactly pinpoint which method would be required to change as the error is not being originated from. But for the sake of this project and the version of LangChain that is being used for the project. It wouldn’t effect the functioning of the project.

## Integration Testing Plan:



## Integration Testing Outcomes:



# Evaluation:

There were two sets of primary objectives and secondary objectives which defined the scope of the project, where primary objectives took the most importance for the success of the project followed by secondary objectives. Following a agile solution approach SCRUM, outputs were produced during each of the sprint which served as a building block for the application thus resulting in a fully functioning application. The resulting application demonstrates wide variety of functional features which with the combination of functional and non-functional system requirements analyzed by the MoSCow Method address the set 4 primary objectives but in the case of secondary objectives 2 failed to be accomplished.

The P1 objective stating the development of a User Interphase for the user, along with making it dynamic for different devices was successfully accomplished. The set objective was accomplished by following proper agile solution approach with the emphasis on time management where tools like Trello board and Gantt Chart were used. The Set objective as approached systematically where initial design was constructed using Wireframes which were then implemented in code to avoid any changes in design in the later stage. Followed by P1, the P2, and P3 objectives were put under development. The focus of P1 was developing Natural language processing system unit in that could understand the user queries and process it accurately in real-time. The accomplishment of the this objective was a bit challenging as I had to work with a lot of new technologies but thankfully due to proper Time management and solution approach I was able to tackle it appropriately and thus was able to product a proof-of-concept NLP model which was able to carry out the set tasks using the help of libraries such as langchaing, CTransformers, OpenAI(), etc. The P3 objective was carried out in parallel where lots of articles concerning household and tenant law were collected and they were integrated into the NLP model by making vector databases. Most of the articles collected were PDFs, thus the processing of that information was made easy by libraries such PyPDF Loader. Both of the these objectives were completed in a systematic way following the proper plan and designs set in place.

Moving onto the final primary objective P4, Real time response generation for accomplishing this objective technoglies like JS Fetch Requests and Flask Server Handling were used together to initiate requests and gather responses and display it on the front-end. This enabled the users to get advice from the AI in real-time and a comparative studies were carried out too to find the best model and parameters to get advice as quickly as possible. Which resulted in the best configuration of the model hence enabling to accomplish this objective.

When it comes to secondary objectives S1 was achieve partly due to the fact most of the resources and time allocated was used on focusing on the achieving the primary objective. When it comes to accesibily steps were taken to make it dynamic so that it could be supported on multiple devices giving more access to the users. The other objective that wasn’t been able to be accomplished was S4, to establish a feedback system for the Model in a sense it would remember the conversation with the person and can use that as context of the conversation which would still be a part of future development of the project. The objectives S2 and S3 were accomplished having set up the authentication model with ample security measure such as locking the URLs along with using Bcrypt to encrypt all the passwords for the users. Which signifies successful completion of these objectives.

Using Scrum project solution approach, I was able to tackle all of the problems I faced to be delt with in a timely Manner. Putting in place a proper time management system ensured that I was able to accomplish my objectives on time whether they were primary or secondary objectives. Doing requirement analysis and planning I was able to short list all the objectives that were of prime importance for the success of the project. Using all these techniques I was able to complete the project to produce a proof-of-concept application which met the objectives that were set out for it. Which I’m really proud of these said techniques made sure if I faced any kind of problems it would a failsafe put in place.

# Future Work:

There are number of areas that could be improved in the scope of this project. Which I believe can be implemented to make the system more robust and have a lot more features than it currently has.

The first improvement that would be implemented in the future for this application would be to establish a **feedback system** along with chat remembering feature for the AI so that the user could have a conversation with the model and gradually improve the output the model results in. And the feedback would make the model learn and gradually improve its performance.

The second improvement that would be implemented will be an **online vector database** to increase its accessibility across devices. One of the implementations of it was explored but was failed to be implemented for the current application. So, I would like to explore alternatives such as Weaviate to implement it. Thus adding more versatility to the project.

The third improvement would be to explore models like **Grok** to implement the functionality as this model can support upwards of 314 billion paramters with context token size of 8K tokens. Which is great improvement from gpt-3.5-turbo model but still relatively less effective than GPT-4 and I want to test if this can server as a better option relative to gpt-3.5-turbo model in response time.

The fourth improvement I would like to make it is to increase the information available in the dataset and focus on **other domains of law** as well which I feel like would provide more utility instead of just focusing on a single domain but given the restraint of time and resources for this proof-of-concept application it couldn’t be implemented in this development phase but if this application is decided to work on this improvement would be implemented.

The fifth improvement would be to make the advice that generated out of the model should be more structured with headings and sub-headings which I believe would be done with the help of **markdown language**. This would make the advice look more readable and more untestable for the user.

The sixth area of development would be supporting **file uploads to proof-read-contracts** or even draw up contracts for you like tenancy agreement and such. Which I believe would provide a lot of utility to the uses and hence would greatly increase user satisfaction. And I feel like this area would make Litigat8 stand out and provide real utility to the users.

These are some of the key areas of improvements that could be worked on to improve the litigat8’s application and functionality. If I decide to work on the application beyond proof-of-concept then all of these improvement areas would be Analysed and worked on and would be implemented in the real-world application.

# Conclusion:

# References

1. Khurana D, Koli A, Khatter K, Singh S. Natural language processing: state of the art, current trends and challenges. Multimed Tools Appl. 2023;82(3):3713-3744. doi: 10.1007/s11042-022-13428-4. Epub 2022 Jul 14. PMID: 35855771; PMCID: PMC9281254.(for Image and the introduction)
2. Chomsky, N. (1965) 'Aspects of the Theory of Syntax', MIT Press, Cambridge, MA.
3. Sutskever, I., Vinyals, O., and Le, Q.V. (2014) 'Sequence to Sequence Learning with Neural Networks', in Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14), MIT Press, Cambridge, MA, USA, pp. 3104–3112.
4. and Short Papers), pp. 4171–4186.
5. Hutchins, W.J. (1986) 'Machine Translation: Past, Present, Future', Ellis Horwood, Chichester, England.
6. Hutchins, W.J. (1995) 'Machine Translation: A Brief History', in Concise History of the Language Sciences: From the Sumerians to the Cognitivists, pp. 431–445.
7. Manning, C.D., and Schütze, H. (1999) 'Foundations of Statistical Natural Language Processing', MIT Press, Cambridge, MA.
8. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019) 'Language Models are Unsupervised Multitask Learners', OpenAI Blog.
9. Winograd, T. (1972) 'Understanding Natural Language', Academic Press, New York, NY.
10. Jackendoff, R. (2002) 'Foundations of Language: Brain, Meaning, Grammar, Evolution', Oxford University Press, Oxford, UK.
11. Bird, S., Klein, E., and Loper, E. (2009) 'Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit', O'Reilly Media, Inc., Sebastopol, CA.
12. Jurafsky, D., and Martin, J.H. (2019) 'Speech and Language Processing (3rd ed. draft)'
13. Hassler, M., & Fliedl, G. (2006). Text Preparation Through Extended Tokenization.(for tokensation images)
14. Jurafsky, D., and Martin, J.H. (2009) 'Speech and Language Processing (2nd ed.)', Pearson Prentice Hall, Upper Saddle River, NJ.
15. Manning, C.D. (2011) 'Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?', in Computational Linguistics and Intelligent Text Processing, Springer, Berlin, Heidelberg, pp. 171-189.
16. Smith, N.A. (2011) 'Linguistic Structure Prediction', Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers.
17. [https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/](https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/(linke) (link for the pos image
18. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016) 'Neural Architectures for Named Entity Recognition', in Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 260–270.
19. Nadeau, D., and Sekine, S. (2007) 'A Survey of Named Entity Recognition and Classification', Linguisticae Investigationes, 30(1), pp. 3–26.
20. Chen, D., and Manning, C.D. (2014) 'A Fast and Accurate Dependency Parser using Neural Networks', in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 740–750.
21. Kübler, S., McDonald, R., and Nivre, J. (2009) 'Dependency Parsing', Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers.
22. Manning, C.D., Raghavan, P., and Schütze, H. (2008) 'Introduction to Information Retrieval', Cambridge University Press, New York, NY, USA.
23. Porter, M.F. (1980) 'An algorithm for suffix stripping', Program, 14(3), pp. 130–137.
24. Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019) 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding', in Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171–4186.
25. Goodfellow, I., Bengio, Y., and Courville, A. (2016) 'Deep Learning', MIT Press, Cambridge, MA, USA.
26. Vaswani, A. et al. (2017) 'Attention is All You Need', in Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 5998–6008.
27. <https://www.codemotion.com/magazine/video/nlp-techniques-and-deep-learning/> (image for machine learning in NLP)
28. <https://sonix.ai/articles/difference-between-artificial-intelligence-machine-learning-and-natural-language-processing(image> for machine learning in NLP
29. Barocas, S., Hardt, M., and Narayanan, A. (2019) 'Fairness and Abstraction in Sociotechnical Systems', in Proceedings of the Conference on Fairness, Accountability, and Transparency, pp. 59–68.
30. Romanosky, S. (2016) 'Examining the Costs and Causes of Cyber Incidents', Journal of Cybersecurity, 2(2), pp. 121–135.
31. Voigt, P., and Von dem Bussche, A. (2017) 'The EU General Data Protection Regulation (GDPR)', Springer International Publishing, Cham.
32. Ashley, K.D. (2017) 'Artificial Intelligence and Legal Analytics: New Tools for Law Practice in the Digital Age', Cambridge University Press, Cambridge.
33. Branting, L.K. (2021) 'Explainable Artificial Intelligence and the Interpretability Problem in Legal Analytics', Jurimetrics, 61, pp. 45–68.
34. Kreutzer, R.T., and Sirrenberg, M. (2019) 'Understanding Artificial Intelligence: A Guide for Business Leaders', Springer Nature, Cham.
35. Tsarapatsanis, D., and Aletras, N. (2021) 'Machine Learning in Law: Present and Future', in Artificial Intelligence and Law, 29, pp. 355–385.
36. Zhong, H., Guo, Z., Tu, C., Xiao, C., Liu, Z., and Sun, M. (2020) 'How Does NLP Benefit Legal System: A Summary of Legal Artificial Intelligence', in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 5218–5230.
37. Alarie, B., Niblett, A., and Yoon, A.H. (2018) 'How Artificial Intelligence Will Affect the Practice of Law', University of Toronto Law Journal, 68(S1), pp. 106–124.
38. Katz, D.M., Bommarito II, M.J., and Blackman, J. (2017) 'A General Approach for Predicting the Behavior of the Supreme Court of the United States', PLoS One, 12(4), e0174698.
39. Rule, C. (2017) 'Online Dispute Resolution for All: The History and Future of Online Dispute Resolution', in Online Dispute Resolution: Theory and Practice, Eleven International Publishing, The Hague, pp. 13–32.
40. Sourdin, T. (2018) 'Judge v. Robot? Artificial Intelligence and Judicial Decision-Making', University of New South Wales Law Journal, 41(4), pp. 1114–1133.

Talk about the PineCone bug