LEGAL OUTCOME PREDICTION USING MACHINE LEARNING

An Industrial Oriented Mini Project (CS653PC)

Submitted

in the partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology VI Semester

in

Computer Science and Engineering

by

CH. SATHVIK (22261A05D9)

and

D. KRISHNA KARTHIK (22261A05E1)

Under the guidance of

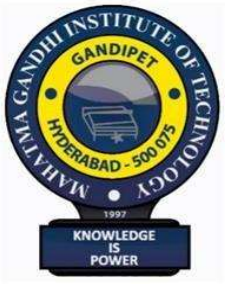
Dr. K. SREEKALA

(Assistant Professor)

and

Mr. Y. PAVAN NARASIMHA RAO

(Assistant Professor)



Department Of Computer Science and Engineering

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY(A),

GANDIPET, HYDERABAD-500 075, INDIA

2024 - 2025

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)
Gandipet, Hyderabad-500 075, Telangana (India)

CERTIFICATE

This is to certify that the Industry Oriented Mini Project (CS653PC) entitled "**Legal Outcome Prediction Using Machine Learning**", is being submitted by **Mr. CH. Sathvik** bearing **Roll No: 22261A05D9** and **Mr. D. Krishna Karthik** bearing **Roll No: 22261A05E1** in partial fulfillment for completion **of Bachelor of Technology VI Semester** in **Computer Science and Engineering** to **Mahatma Gandhi Institute of Technology** is a record of bona-fide work carried out by them under our guidance and supervision.

Project Guide
Dr. K. Sreekala
Asst. Professor, Dept. of CSE

Head of the Department
Dr. C.R.K Reddy
Professor, Dept. of CSE

Project Guide
Mr. Y. Pavan Narasimha Rao
Asst. Professor, Dept. of CSE

EXTERNAL EXAMINER

DECLARATION

This is to certify that the work reported in Industry Oriented Mini Project (CS653PC) entitled "**Legal Outcome Prediction Using Machine Learning**" is a record of work done by us in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad. No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by us and not copied from any other source. The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

CH.SATHVIK
(22261A05D9)

D.KRSHNA KARTHIK
(22261A05E1)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible because success is the abstract of hard work and perseverance, but steadfast of all is encouraging guidance. So, we acknowledge all those whose guidance and encouragement served as a beacon light and crowned my efforts with success.

.

We would like to express our sincere thanks to, **Prof G. Chandra Mohan Reddy, Principal MGIT**, for providing the working facilities in college.

We wish to express our sincere thanks and gratitude to **Dr. C R K Reddy, Professor and HOD**, Department of CSE, MGIT, for all the timely support and valuable suggestions during the period of project.

We are extremely thankful to **Dr. K. Sreekala, Assistant Professor, Department of CSE, MGIT and Mr. Y. Pavan Narsasimha Rao, Assistant Professor, Department of CSE, MGIT**, Industrial Oriented Mini Project Guides for their encouragement and support throughout the project.

We are extremely thankful to **Mr. G. Nagi Reddy, Assistant Professor, Department of CSE, MGIT and Mr. N. Rama Krishna, Assistant Professor, Department of CSE, MGIT** Industry Oriented Mini Project Coordinators for their encouragement and support throughout the project.

Finally, we would also like to thank all the faculty and staff of CSE Department who helped us directly or indirectly in completing this project.

CH.SATHVIK
(22261A05D9)

D.KRISHNA KARTHIK
(22261A05E1)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI : Artificial Intelligence

BERT : Bidirectional Encoder Representations from Transformers

CPU : Central Processing Unit

CSS : Cascading Style Sheets

CSV : Comma-Separated Values

FAISS : Facebook AI Similarity Search

GPU : Graphics Processing Unit

HTML : Hyper Text Markup Language

HTTP : Hypertext Transfer Protocol

JSON : JavaScript Object Notation

JS : JavaScript

LIME : Local Interpretable Model-agnostic Explanations

NLP : Natural Language Processing

NoSQL : Not Only SQL

OS : Operating System

RAM : Random Access Memory

SHAP : SHapley Additive exPlanations

VS Code : Visual Studio Code

ABSTRACT

Legal Outcome Prediction Using Machine Learning is an advanced machine learning-based system designed to assist both legal professionals and ordinary citizens within the Indian legal system. The system predicts case outcomes (win/lose), recommends relevant case precedents, and estimates numerical results such as compensation, sentencing, and damages. It fine-tunes Legal-BERT using Indian legal data to provide accurate predictions and effective case retrieval, applying machine learning algorithms to uncover patterns from historical cases. Additionally, the system utilizes prompt-based techniques for precise numerical estimates and integrates explainable AI to offer transparent justifications for its predictions. A user-friendly web interface allows both legal professionals and citizens to input case details, receive predictions, explore similar cases, and understand the reasoning behind the system's decisions. By improving the speed, transparency, and accessibility of legal decision-making, this system empowers both legal professionals and the general public with data-driven insights, enhancing case preparation, risk assessment, and overall legal outcomes in India.

Keywords: Legal Outcome Prediction, Machine Learning, Legal-BERT, Historical Case Patterns, web interface, explainable AI

1.INTRODUCTION

The Indian legal system, with its vast and intricate web of laws, judgments, and statutes, often presents a daunting challenge when it comes to predicting the outcomes of legal cases. Lawyers and judges traditionally rely on their experience, intuition, and established precedents to make educated guesses about how a case might unfold. However, given the complexity and ever-evolving nature of legal texts, along with the subjective interpretation involved in legal proceedings, predicting the outcome of a case with accuracy remains a significant hurdle. This project, "Legal Outcome Prediction Using Machine Learning," seeks to bridge this gap by using advanced machine learning techniques to predict legal outcomes, such as whether a case will be won or lost, as well as to recommend relevant legal precedents and estimate potential compensation or sentencing. By specifically tailoring this system to the Indian legal context, the goal is not only to support legal professionals but also to empower ordinary citizens who may be unfamiliar with the complexities of the law. The system aims to bring clarity, transparency, and efficiency to the decision-making process, ultimately making legal insights more accessible to everyone. By analyzing historical case data, legal precedents, and other influential factors, this system offers a tool to help individuals navigate the intricate legal landscape with greater confidence and understanding.

1.1 PROBLEM DEFINITION:

The Indian legal system is vast and complex, making it difficult for individuals to predict case outcomes or find relevant precedents. Legal professionals rely on experience and manual analysis, which is time-consuming and inaccessible to the general public. This project aims to use machine learning to predict legal outcomes, suggest relevant precedents, and estimate compensation or sentencing, thereby improving legal transparency and accessibility.

1.2 OBJECTIVES

- **Develop a Machine Learning Model:** Fine-tune Legal-BERT on Indian legal data (case laws, statutes, judgments) to predict case outcomes (win/lose).
- **Numerical Outcome Prediction:** Estimate compensation amounts and sentencing based on historical legal data.
- **Explainable AI:** Implement explainable AI techniques (e.g., SHAP and LIME) to ensure predictions are transparent and easily understandable by users.
- **System for Both Lawyers and Citizens:** Ensure the system is beneficial for both legal professionals and non-experts, making legal information more accessible to the public.

1.3 EXISTING SYSTEM

Several systems exist that attempt to predict legal outcomes using machine learning models. Notable models, such as Legal-BERT, have been pre-trained on large legal corpora to aid in tasks like case classification, document retrieval, and decision prediction.

DRAWBACKS

- **Localized Focus**: Many models are trained on Western or general legal data, making them less effective for Indian legal contexts.
- **Lack of Numerical Predictions**: Most systems predict binary outcomes (win/lose), but fail to estimate compensation amounts or sentencing.
- **Complexity**: Current systems are often too technical for ordinary citizens and are designed mainly for legal professionals.

1.4 PROPOSED SYSTEM

The proposed system is designed specifically for the Indian legal context, aiming to make legal predictions more accessible and practical. It leverages localized Indian legal data by fine-tuning the Legal-BERT model to improve the accuracy of predictions. The system provides not only outcome predictions, such as whether a case will be won or lost, but also estimates numerical outcomes like compensation amounts and sentencing durations. A user-friendly web interface is developed to ensure ease of use for both legal professionals and ordinary citizens. To promote transparency and trust, the system integrates explainable AI techniques such as SHAP and LIME, allowing users to understand the reasoning behind predictions. Additionally, it offers comprehensive resources by recommending relevant legal precedents to support case preparation effectively.

ADVANTAGES

- **Higher Accuracy**: Tailored for Indian legal data.
- **Increased Usability**: Accessible for both experts and non-experts.
- **Transparency**: Explainable AI builds trust.
- **Comprehensive Prediction**: Includes both case outcomes and numerical predictions.

1.5 HARDWARE REQUIREMENTS

The hardware requirements define the necessary system configuration for implementing the proposed breast cancer detection model. These specifications ensure efficient processing of medical images and seamless execution of deep learning models.

- **PROCESSOR**: Intel Core i5/i7 or AMD Ryzen 5/7 (Quad-core or higher)
- **RAM**: 8GB DDR4 (Minimum), 16GB
- **HARD DISK**: 500GB HDD (Minimum) or 256GB SSD (Recommended for faster processing)
- **GPU**: NVIDIA GTX 1080/RTX 2060 or higher (preferred for faster training of deep learning models like BERT).

1.6 SOFTWARE REQUIREMENTS

- **Operating System**: Windows, Linux, or macOS.
- **Development Tools**: Python, Jupyter Notebook, Integrated Development Environment (IDE)
- **Machine Learning Libraries**: Hugging Face Transformers, PyTorch, Scikit-learn
- **Web Development**: Flask/Django, HTML/CSS/JavaScript, Database (MongoDB)

2.LITERATURE SURVEY

The literature survey table provided in the presentation captures various studies related to fitness programs and diet planners, offering insights into methodologies, benefits, and limitations of these approaches. Below is a detailed theoretical explanation for each study mentioned in the table:

[1] Dina, N. Z., Ravana, S. D., & Idris, N. (2025) – "Legal judgment prediction using natural language processing and machine learning methods: A systematic literature review"

Dina et al. conducted a systematic literature review focusing on how Natural Language Processing (NLP) and Machine Learning (ML) are used for legal judgment prediction. The review spans recent advancements and categorizes various models based on input data types, algorithms used, and prediction tasks. It highlights that transformer-based models like BERT have become dominant due to their contextual learning capacity. While offering an extensive overview of methods and datasets, the study is limited by the relatively small number of high-quality publicly available legal datasets across jurisdictions.

[2] Bharati, R. (2025) – "Predictive Justice in Indian Courts: Machine Learning Approaches to Case Outcome Forecasting"

This paper focuses on applying machine learning techniques specifically within the Indian legal system to forecast judicial outcomes. Bharati explores various ML algorithms, including random forest, SVM, and deep learning models, evaluating them on datasets comprising Indian court judgments. The study emphasizes critical challenges such as fairness, transparency, class imbalance, and the interpretability of predictions. It also highlights the unique linguistic, structural, and procedural characteristics of Indian legal texts, which complicate model generalization. The work suggests that integrating explainable AI techniques (e.g., SHAP, LIME) is essential to building user trust in predictive legal models. While the study offers a valuable framework for legal AI in India, it does not benchmark results against international models like Legal-BERT or LawLLM, nor does it explore transformer-based architectures in depth. Nonetheless, it lays important groundwork for ethical and practical applications of ML in Indian court systems.

[3] Shu, D., Zhao, H., Liu, X., Demeter, D., Du, M., & Zhang, Y. (2024) – "LawLLM: A Large Language Model for the U.S. Legal System"

This paper introduces LawLLM, a domain-specific large language model pre-trained on extensive U.S. legal corpora, including court rulings, statutes, and legal commentary. The model is optimized for legal tasks such as outcome prediction, legal question answering, summarization, and citation retrieval. It incorporates legal-specific training objectives, such as legal entailment and issue spotting, which enhance its contextual understanding and interpretability. LawLLM demonstrates improved performance over general-purpose models like GPT and Legal-BERT in benchmark tasks. The paper also emphasizes the importance of fine-tuning on jurisdiction-specific data to boost accuracy. However, since LawLLM is trained exclusively on U.S. legal documents, its direct applicability to other legal systems (e.g., India's) is limited without significant domain adaptation.

[4] Nigam, S. K., Patnaik, B. D., Mishra, S., Shallum, N., Ghosh, K., & Bhattacharya, A. (2025) – "TathyaNyaya and FactLegalLlama: Advancing Factual Judgment Prediction in the Indian Legal Domain"

This paper presents two state-of-the-art models—TathyaNyaya and FactLegalLlama—developed specifically for factual judgment prediction in the Indian legal context. The models use a multi-stage pipeline architecture that aligns facts with summaries and judgment outcomes, enhancing factual consistency and reducing bias. The paper reports notable improvements in outcome prediction accuracy across benchmark Indian legal datasets and includes explainability features to promote transparency. The authors argue that the architecture allows better generalization to varied case types while maintaining trustworthiness. However, the study also notes limitations, including the need for larger, more diverse datasets and further testing across different courts and jurisdictions. Despite this, the paper marks a significant step forward in factual and transparent legal AI for India.

[5] Patnaik, B. D., Mishra, S., Shallum, N., & Ghosh, K. (2024) – "NyayaAnumana & INLegalLlama: The Largest Instruction-Tuned Legal LLMs for Indian Case Outcome Prediction"

This paper introduces NyayaAnumana and INLegalLlama, two instruction-tuned large language models specifically trained on extensive Indian legal data. The authors focus on enhancing prompt-based usability by aligning the models with instruction-following tasks, enabling legal professionals and non-experts to query the models without technical interfaces. These LLMs outperform traditional fine-tuned Legal-BERT models in classification and information retrieval tasks, especially in Indian court contexts. The models leverage multi-task fine-tuning with summarization, classification, and precedent retrieval capabilities. Despite these strengths, the paper acknowledges that the models currently underperform in predicting numerical outcomes such as compensation, imprisonment duration, or fines—suggesting this as a future research direction to improve utility in practical legal decision-support systems.

[6] Azmi, M., Atkinson, K., Mumford, J., & Bench-Capon, T. (2024) – "Legal Judgment Prediction via Argument Analysis"

Azmi et al. present a hybrid framework that integrates argument mining with deep learning for legal judgment prediction. Their method involves identifying and structuring arguments from legal texts using manual and automated annotation, followed by encoding this structured data into a BERT-based classification pipeline. The model significantly outperforms traditional text-only baselines by focusing on legal reasoning structures rather than just semantic similarity. Additionally, the system improves explainability by linking predicted outcomes to the underlying argument components, offering transparent reasoning paths. However, the approach relies heavily on rich annotated datasets, making it resource-intensive and potentially difficult to scale across domains or jurisdictions without standardized argumentation annotation tools.

[7] Zeleznikow, J. (2023) – "The Benefits and Dangers of Using Machine Learning to Support Making Legal Predictions"

This conceptual paper offers a critical evaluation of the implications of using machine learning in legal

decision-making systems. Zeleznikow discusses ethical dilemmas such as algorithmic bias, lack of transparency, and fairness in predictions. The paper introduces a risk-mitigation framework that emphasizes human oversight, explainability, and legal accountability as essential elements of trustworthy AI in the legal domain. While the study contributes significantly to theoretical discussions, it does not present empirical findings or validate its framework through practical implementations. As such, it serves more as a guideline for ethical deployment rather than a technical or application-driven paper.

[8] Dharma, P. Y., Widyawan, W., & Pratama, A. R. (2023) – "Legal Judgment Prediction: A Systematic Literature Review"

This systematic review aggregates and analyzes scholarly work on legal judgment prediction up to 2023. The paper classifies models by machine learning methodology (e.g., SVMs, neural networks, transformers), jurisdiction (e.g., China, India, U.S.), and dataset usage. The review identifies a clear trend toward transformer-based models such as BERT and RoBERTa due to their superior performance in contextual understanding. It also highlights emerging interest in explainable AI and multilingual legal modeling. Although the review is comprehensive in its scope, it primarily draws from conference and workshop papers, and lacks an in-depth meta-analysis of performance metrics or comparative results across jurisdictions. Nevertheless, it is a valuable resource for identifying research gaps and trends.

[9] Mumford, J., Atkinson, K., & Bench-Capon, T. (2022) – "Reasoning with Legal Cases: A Hybrid ADF-ML Approach"

In this work, the authors propose a hybrid legal reasoning model that combines Abstract Dialectical Frameworks (ADF) with machine learning classifiers. ADFs represent legal argumentation structures symbolically, while ML models are trained to predict outcomes based on these structured representations. This fusion improves both interpretability and predictive accuracy, as the symbolic model can offer rational justifications for the decisions made by the ML component. The framework is particularly useful for case law where legal reasoning plays a pivotal role. However, the integration of formal logic and data-driven methods requires domain-specific design of argumentation structures and is computationally complex, making real-world deployment challenging without expert legal-technical collaboration.

[10] Silva, H., António, N., & Bação, F. (2022) – "A Rapid Semi-automated Literature Review on Legal Precedents Retrieval"

Silva et al. design a rapid semi-automated methodology to conduct literature reviews in the domain of legal precedent retrieval. By integrating keyword expansion techniques and NLP-powered document filtering, their system reduces manual workload while maintaining relevance and coverage. This tool is especially useful in tracking emerging trends and clustering research themes in precedent retrieval. The authors highlight its ability to assist researchers in systematically navigating legal AI literature. However, the tool's effectiveness depends heavily on the initial keyword choices, and the process does not yet achieve full automation, which can limit scalability across different legal domains or languages.

[11] Travaini, G. V., Bosia, M., & De Micco, F. (2022) – "Machine Learning and Criminal Justice: A Systematic Review of Advanced Methodology for Recidivism Risk Prediction"
This paper offers an extensive review of machine learning models used to forecast recidivism risk in criminal justice. Travaini et al. analyze a variety of ML techniques, including decision trees, ensemble methods, deep learning, and more novel approaches such as causal inference models. The review also addresses ethical concerns related to fairness, bias, and transparency in recidivism prediction tools, offering a structured framework to assess model accountability. While the review is deeply informative for jurisdictions concerned with criminal reform, its focus on criminal recidivism limits its direct applicability to broader legal areas such as civil litigation or administrative law.

[12] de Oliveira, R. S., Reis Jr., A. S., & Nascimento, E. G. S. (2022) – "Predicting the Number of Days in Court Cases Using Artificial Intelligence"
This paper explores how machine learning can be applied to predict the expected duration of legal proceedings. By training regression models—particularly tree-based ensembles like Random Forest and Gradient Boosting—on historical court data, the authors achieve relatively accurate forecasts for case timelines. The approach can support judicial resource planning, scheduling, and administrative efficiency. The model's interpretability and relatively simple deployment are notable strengths. However, the dataset used is specific to a localized court system, making the findings less generalizable across different jurisdictions or case types without retraining or data adaptation.

[13] Cui, J., Shen, X., Nie, F., Wang, Z., Wang, J., & Chen, Y. (2022) – "A Survey on Legal Judgment Prediction: Datasets, Metrics, Models and Challenges"
Cui et al. provide a thorough and structured survey of the field of legal judgment prediction, compiling insights on the most commonly used datasets (e.g., CAIL, ECtHR, SCOTUS), evaluation metrics (accuracy, F1-score, macro-averages), and core machine learning models including CNNs, RNNs, and transformer-based architectures. They categorize challenges such as data imbalance, explainability, domain adaptation, and legal logic inconsistency. The paper serves as a foundational reference for researchers entering the field. Despite its comprehensiveness, the review remains descriptive and does not provide comparative empirical results or benchmarking, which could have strengthened its practical utility.

[14] Almuslim, I., & Inkpen, D. (2022) – "Legal Judgment Prediction for Canadian Appeal Cases"
In this work, the authors apply supervised machine learning classifiers—such as Logistic Regression, SVM, and neural networks—to predict outcomes of Canadian appellate court decisions. The study focuses on extracting predictive linguistic features from textual case documents and analyzes their impact on outcome prediction performance. The models exhibit strong classification accuracy and are further enhanced by examining the role of legal citation patterns and argumentation language. However, the legal and linguistic structures specific to Canadian appellate jurisprudence restrict the model’s adaptability to other legal systems, particularly those that follow civil law traditions like India or Brazil.

[15] Dong, Q., & Niu, S. (2021) – "Legal Judgment Prediction via Relational Learning" Dong and Niu propose a novel legal judgment prediction framework based on relational learning. Unlike traditional text-only methods, their model leverages graph embeddings to capture inter-case dependencies, effectively modeling how precedent and legal relationships influence judicial outcomes. This approach enhances accuracy, especially in cases involving complex legal hierarchies or repeated patterns across judgments. The results show a notable improvement over baseline models. However, the increased model complexity impacts interpretability and presents scalability challenges, particularly when applied to large, heterogeneous legal corpora where graph construction can become computationally intensive.

[16] Shirsat, K., Keni, A., Chavan, P., & Gosavi, M. (2021) – "Legal Judgment Prediction System" This paper presents a basic yet functional ML pipeline that applies Support Vector Machines (SVM) with hand-engineered features to classify legal case outcomes. The approach emphasizes simplicity and practicality, making it accessible for deployment in resource-constrained environments or as a teaching tool for legal informatics. Despite its merits in terms of ease of implementation and speed, the system suffers from limitations such as reliance on small, curated datasets, lack of semantic feature extraction, and difficulty scaling to complex legal cases or multilingual legal corpora. Consequently, the model's predictive capacity remains modest.

[17] Rosili, N. A. K., Hassan, R., Zakaria, N. H., & Kasim, S. (2021) – "A Systematic Literature Review of Machine Learning Methods in Predicting Court Decisions" Rosili et al. provide a comprehensive literature review of machine learning methods applied to legal outcome prediction. Their analysis spans a range of algorithms—including Naive Bayes, Decision Trees, and deep learning models—and covers various jurisdictions and data modalities. The paper identifies common trends, such as increased adoption of NLP and concerns about fairness, as well as research gaps in interpretability and dataset diversity. It serves as a valuable mapping of the field up to 2021. However, the review is descriptive in nature and does not introduce new experimental validations or model innovations, which somewhat limits its impact.

[18] Malik, V., Sanjay, R., Nigam, S. K., Ghosh, K., Guha, S. K., & Modi, A. (2021) – "ILDC for CJPE: Indian Legal Documents Corpus for Court Judgment Prediction and Explanation" This paper introduces the Indian Legal Documents Corpus (ILDC), a significant contribution aimed at standardizing and facilitating NLP research in the Indian legal context. The authors fine-tune a BERT-based model on this corpus to perform both outcome prediction and judgment explanation, establishing baselines for legal NLP in India. ILDC fills a critical gap by offering a large, labeled dataset of Indian court judgments. While the work emphasizes data availability and resource creation, it lacks deeper exploration into advanced modeling techniques or comparative benchmarking with newer transformer architectures such as Legal-BERT or LawLLM.

[19] Sharma, S., Shandilya, R., & Sharma, S. (2021) – "Predicting Indian Supreme Court Judgments, Decisions, or Appeals"

Sharma et al. present a study using machine learning techniques for predicting decisions of the Indian Supreme Court. Their methodology incorporates textual features from case documents along with structured inputs like citations and case metadata. The model achieves moderate predictive accuracy, reflecting the complexity and variability of Indian legal language. A notable challenge addressed in the paper is dataset imbalance, which negatively impacts model fairness and consistency. Furthermore, while deep learning methods improve performance, they reduce interpretability, raising concerns around legal transparency and trust in AI-generated judgments.

[20] Katz, D. M., Bommarito II, M. J., & Blackman, J. (2017) – "A General Approach for Predicting the Behaviour of the Supreme Court of the United States"

Katz et al. conduct one of the pioneering large-scale applications of machine learning in the legal domain by predicting decisions of the U.S. Supreme Court. Their model combines ensemble techniques with features such as case issue, time period, and justice voting patterns, achieving considerable predictive accuracy over multiple decades. This study laid the groundwork for data-driven legal analytics and inspired future research in judicial behaviour modeling. However, with the advent of transformer-based LLMs and contextual embeddings, more recent approaches now surpass its performance, positioning this work as a historical and conceptual benchmark rather than a state-of-the-art solution.

Table 2.1: Literature Survey Table

S.No	Authors	Title	Year	Name of the Journal	Methodology	Merits	Demerits
1	N. Z. Dina et al.	Legal judgment prediction using NLP and ML methods: A systematic literature review	2025	SAGE Open	Systematic literature review using NLP & ML	Comprehensive; identifies trends in LJP	Limited to published studies
2	R. Bharati	Predictive Justice in Indian Courts: ML Approaches to Case Outcome Forecasting	2025	SSRN	ML on Indian legal corpus	India-specific; practical ML use	Not peer-reviewed
3	D. Shu et al.	LawLLM: A large language model for the U.S. legal system	2024	arXiv	LLM fine-tuned on U.S. legal text	Large-scale LLM for U.S. law	U.S.-centric; no Indian benchmarking
4	S. K. Nigam et al.	TathyaNyaya & FactLegalLlama: Factual Judgment Prediction in Indian Legal Domain	2025	arXiv	LLM on factual Indian court data	High performance; Indian focus	Limited benchmarks
5	B. D. Patnaik et al.	NyayaAnumana & INLegalLlama: Instruction-Tuned Legal LLMs for India	2024	arXiv	Instruction-tuned transformers	Open-source; large dataset	Resource-intensive models
6	M. Azmi et al.	Legal judgment prediction via argument analysis	2024	Frontiers in AI and Applications	Argument mining + NLP	Combines reasoning & ML	Complex annotation needed
7	J. Zelezniko w	Benefits and dangers of using ML for legal predictions	2023	Wiley Interdisciplinary Reviews	Conceptual analysis	Highlights ethical & bias risks	No empirical testing
8	P. Y. Dharma et al.	Legal judgment prediction: A systematic literature review	2023	ICAMIMIA	Review of LJP techniques	Recent review of models	Limited scope

9	J. Mumford et al.	Reasoning with legal cases: A hybrid ADF-ML approach	2022	Frontiers in AI and Applications	Argumentation Framework + ML	Blends symbolic and statistical reasoning	Complexity in implementation
10	H. Silva et al.	Rapid Semi-automated Literature Review on Legal Precedents Retrieval	2022	Lecture Notes in Computer Science	Semi-automated review	Accelerated literature analysis	Keyword-dependence
11	G. Travaini et al.	ML and Criminal Justice: Review for Recidivism Prediction	2022	Int. J. Environ. Res. Public Health	Survey of ML methods	Covers criminal law; public health tie	Narrow domain
12	R. de Oliveira et al.	Predicting number of days in court cases using AI	2022	PLOS ONE	Regression ML models	Predicts legal delay times	One-jurisdiction focus
13	J. Cui et al.	A survey on legal judgment prediction	2022	arXiv	Dataset & model review	Rich dataset and challenge review	No experimental validation
14	Almuslim & D. Inkpen	Legal Judgment Prediction for Canadian Appeal Cases	2022	IEEE	Classification for appeals	Real-world application	Canadian law specific
15	Dong & S. Niu	Legal judgment prediction via relational learning	2021	ACM SIGIR	Graph-based relational learning	Captures inter-case links	High model complexity
16	K. Shirsat et al.	Legal judgment prediction system	2021	IJCA	ML classification pipeline	Simple, easy-to-implement system	Accuracy concerns; small dataset
17	N. Rosili et al.	SLR of ML methods in predicting court decisions	2021	IAES Int. J. Artif. Intell.	Systematic review	Wide comparison of techniques	No deep dive into performance
18	V. Malik et al.	ILDC for CJPE: Indian legal documents corpus	2021	arXiv	Dataset development; BERT fine-tuning	Provides dataset for Indian law	Less focus on outcome modeling

19	S. Sharma et al.	Predicting Indian Supreme Court judgments	2021	arXiv	Text classification	Indian SC-specific model	Dataset imbalance
20	D. M. Katz et al.	General approach for predicting behavior of SCOTUS	2017	PLOS ONE	Ensemble ML models	First large-scale court prediction	Obsolete compared to LLMs

Table 2.1 provides a comparative overview of 20 key research papers in the field of Artificial Intelligence applications in the legal domain. It includes details such as authors, publication year, journal, methodology used, merits, and demerits. The works reviewed span diverse approaches including traditional machine learning, BERT-based models, hybrid systems, graph-based relational learning, and large language models. The table highlights advancements in legal judgment prediction, precedent retrieval, and legal text classification, while also noting limitations such as domain-specificity, data constraints, and computational challenges. This summary helps identify research trends and gaps for future legal AI developments.

3.METHODOLOGY

3.1 TECHNOLOGIES USED

Programming Language: Python (for data processing, model training, and backend development)

Deep Learning Framework: PyTorch (for fine-tuning Legal-BERT)

Pretrained Models: Legal-BERT (nlpaueb/legal-bert-base-uncased) for domain-specific language understanding, Sentence Transformers (e.g., all-MiniLM-L6-v2) for semantic similarity in precedent retrieval

Data Processing Libraries: Pandas, NumPy, regex

Similarity Search: FAISS (Facebook AI Similarity Search) for efficient retrieval of relevant precedents

Explainability Tools: SHAP, LIME for model interpretation

Web Framework: Flask (to build the prediction system's web interface)

Database: MongoDB (to store case data, user inputs, and retrieval indexes)

3.2 DEVELOPMENT PROCESS

Step 1: Data Collection and Preparation

The system is designed as a web-based application where users can input case summaries to receive legal outcome predictions. It leverages a fine-tuned Legal-BERT model to predict whether the case is likely to be won or lost. Additionally, it retrieves similar past precedents using semantic search and estimates numerical outcomes such as fines or imprisonment. The backend processes model inference, while the frontend displays results along with explainable AI visualizations using SHAP and LIME to ensure transparency and user understanding.

Step 2: Data Preprocessing

The input legal text data is first tokenized and encoded using a tokenizer compatible with the Legal-BERT model to ensure proper formatting for model processing. The dataset is then split into training, validation, and test sets while preserving the class balance between outcomes (e.g., win/loss) to prevent model bias. Finally, categorical outcome labels are converted into numerical format to make them suitable for supervised learning with classification models like Legal-BERT.

Step 3: Model Development

Legal-BERT is fine-tuned on the labeled dataset to perform the classification task of predicting case outcomes based on summary text. In parallel, case precedent retrieval system is developed by encoding the summaries of past legal cases into dense vector embeddings using Sentence Transformers. These embeddings are then indexed using FAISS, for retrieval of the most relevant precedent cases.

Step 4: Model Training and Evaluation

The classification model is trained using optimization techniques like learning rate scheduling and early stopping to improve performance and prevent overfitting. Its effectiveness is assessed using evaluation metrics such as accuracy and F1-score for classification tasks, while Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) is used to evaluate numerical predictions like compensation or sentencing. The performance of the precedent retrieval system is validated by measuring the precision and recall of the retrieved similar cases to ensure their relevance and usefulness.

Step 5: Explainability Integration

SHAP and LIME are used to explain the model's predictions, enhancing transparency and user trust. These tools help visualize which portions of the input text had the most influence on the predicted outcome, making the decision-making process more interpretable and understandable for both legal professionals and ordinary users.

Step 6: System Integration

A Flask-based web application was developed to allow users to input case details or summaries for analysis. The backend integrates the outcome prediction model, precedent retrieval system, and explanation modules (SHAP/LIME). A database is also connected to store user-submitted cases and queries, enabling personalized access and future reference.

4. DESIGN OF LEGAL OUTCOME PREDICTION USING MACHINE LEARNING

The system is designed as a web application that takes case summaries as input and predicts outcomes using a fine-tuned Legal-BERT model. It also retrieves similar precedents and estimates numerical outcomes, with results visualized using SHAP and LIME for interpretability.

4.1 SYSTEM ARCHITECTURE

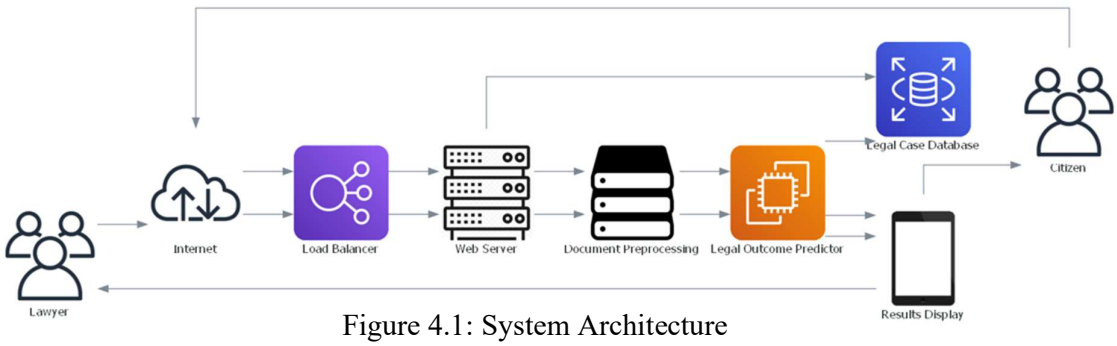


Figure 4.1: System Architecture

Figure 4.1 illustrates the system architecture for the Legal Outcome Prediction platform.

Actors (Users): Lawyers & citizens submit case details via a web interface.

Internet & Load Balancer: Ensures smooth user request handling and system availability.

Web Server: Manages user interactions and forwards data for processing.

Document Preprocessing: Extracts key features, cleans legal text for ML analysis.

Legal Outcome Predictor (ML Model): Uses historical data to predict case outcomes, compensation, and sentencing.

Legal Case Database: Stores past cases & legal precedents for reference.

Results Output: Displays predictions with explanations on the user's interface.

Data Flow: User inputs → Processing → ML model prediction → Output to user

4.2 USE CASE DIAGRAM

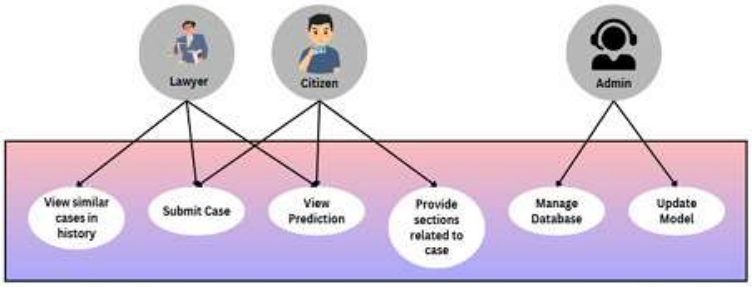


Figure 4.2: Use Case Diagram

Figure 4.2 illustrates the system architecture for the Legal Outcome Prediction platform.

Lawyer: Submit legal cases for analysis. View predicted outcomes. Access similar past cases for reference.

Citizen: Submit cases for legal outcome prediction. View predicted case results. Get relevant legal sections and acts.

Admin: Manage and update the case database. Update the ML model with new legal data.

4.3 SEQUENCE DIAGRAM

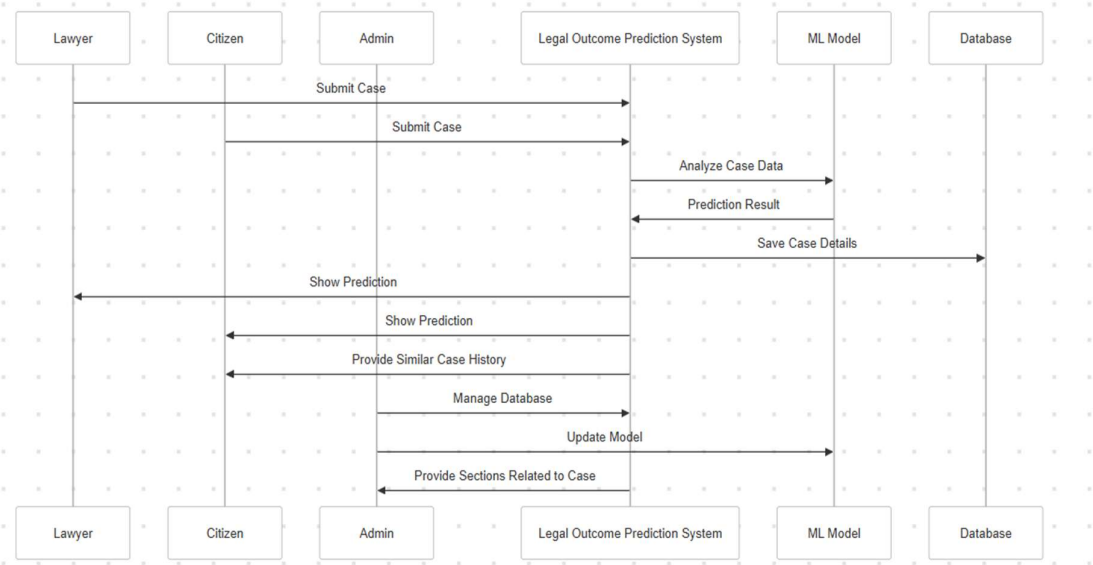


Figure 4.3: Sequence Diagram

Figure 4.3 presents the use case interaction for the Legal Outcome Prediction System. Lawyers and citizens begin by submitting case details through the system interface. The system then processes this data using a trained machine learning model that analyses historical Indian legal cases to predict outcomes. These predicted results such as whether a case is likely to be allowed or dismissed are presented back to the users. Alongside, the system retrieves and displays similar past cases to aid understanding and provide legal context. Additionally, relevant legal sections are shown to citizens. The system is maintained by an admin who oversees the legal case database.

4.4 ACTIVITY DIAGRAM

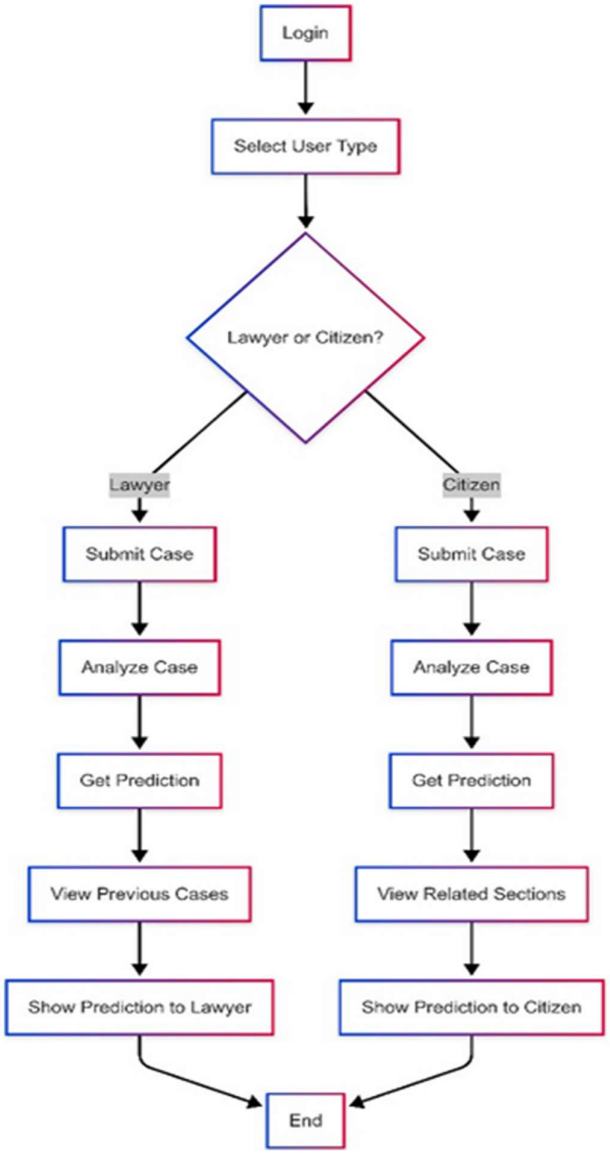


Figure 4.4: Activity Diagram

Figure 4.4 illustrates the activity diagram of the Legal Outcome Prediction System. The process begins when users either lawyers or citizens submit a legal case to the system. The system first preprocesses the input case details, preparing the data for analysis. The machine learning model then processes the cleaned data to predict the legal outcome. Once the prediction is generated, it is presented to the user. Lawyers are given access to similar past cases to support legal research and argument building, while citizens are provided with relevant legal sections to improve their understanding of the judgment. Meanwhile, the admin oversees system performance by updating the case database and retraining the machine learning model regularly to maintain prediction accuracy and system reliability.

4.5 CLASS DIAGRAM

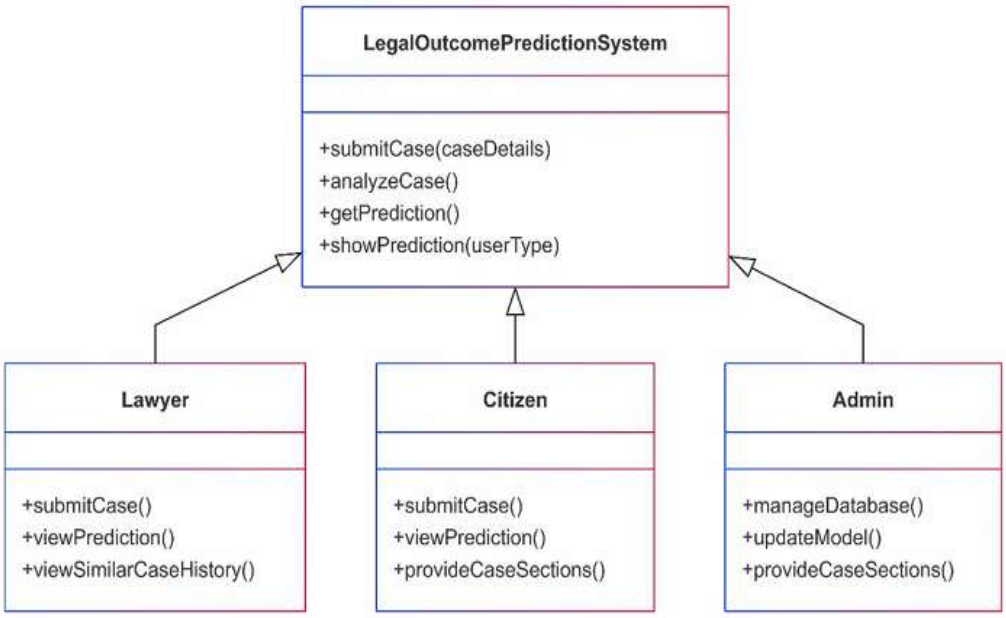


Figure 4.5: Class Diagram

Figure 4.5 depicts the class diagram representing the structural design of the Legal Outcome Prediction System. The system comprises three primary user classes: Lawyer, Citizen, and Admin. Both lawyers and citizens can submit case details and view the system-generated predictions. The Admin class is responsible for maintaining the system's database and periodically updating the machine learning model to ensure accurate predictions. The ML Model class processes the input case data and generates corresponding outcomes. Additionally, the Database class manages storage of historical case data and user-submitted information, while relevant legal sections are also maintained for user reference. This structure ensures modular interaction between users, prediction engine, and the underlying legal data.

5.IMPLEMENTATION

The implementation of the Legal Outcome Prediction System involved setting up multiple modules including data preprocessing, machine learning model training, precedent case retrieval, numerical outcome prediction, and web deployment. This section describes the frontend and backend components used to build the system and key functionalities implemented.

5.1 FRONTEND COMPONENTS:

HTML/CSS: Used for building the structure and styling of the web interface of legal outcome prediction using machine learning

JavaScript (with jQuery): Enabled dynamic interaction and asynchronous communication with the backend.

Bootstrap: For responsive design and quick layout development.

Flask Templates (Jinja2): Rendered prediction results and precedent case details dynamically.

Chart.js / SHAP visualizations: Used to display explanations for model predictions.

5.2 BACKEND COMPONENTS:

Python (Flask Framework): Core server-side language and web framework used to manage routing, session handling, and prediction processing.

Hugging Face Transformers: Used for implementing and fine-tuning the Legal-BERT(nlpaueb/legal-bert-base-uncased) model on the case summaries for outcome classification.

Sentence Transformers + FAISS: Used for encoding case summaries and retrieving top-k similar precedents based on semantic similarity.

MongoDB: Used to store case data, user inputs, and indexed case embeddings.

Regular Expressions (re module): Applied to extract fine, imprisonment, and compensation values from raw judgment text.

5.3 KEY FUNCTIONALITIES:

- **User Input and Preprocessing:**
  Users input case summaries or full judgments via a web form. The system processes this input text by tokenizing and preparing it for the model.
- **Outcome Prediction Module:**
  The system utilizes a fine-tuned Legal-BERT model to predict the outcome of a case as either "Allowed" or "Dismissed." To address the issue of class imbalance during training, a class-weighted loss function was applied, ensuring the model does not favor the majority class and achieves more balanced performance.

- **Precedent Retrieval Module:**

  A FAISS-based retrieval engine is employed to search a precomputed vector index of court case embeddings. The system retrieves and displays the top five semantically similar cases, along with their corresponding similarity scores, enabling users to reference relevant precedents efficiently
- **Numerical Outcome Estimation:**

  Based on the retrieved similar cases, the system aggregates their fine, imprisonment, and compensation data. These aggregated values are then displayed as "Estimated" outcomes for the user's input case, providing meaningful context and enhancing the transparency of the prediction.
- **Explainability Module:**

  To enhance interpretability, SHAP or LIME is used to explain the predictions made by the BERT model. This allows users to visualize which specific words or phrases in the input text influenced the classification outcome, promoting trust and understanding of the system's decision-making process.
- **Security and Session Handling:**

  The application features user registration and login functionalities with encrypted password storage to ensure data security. Additionally, session data is securely managed using Flask sessions, providing a safe and reliable user experience.

6. TESTING AND RESULTS

Home Page:

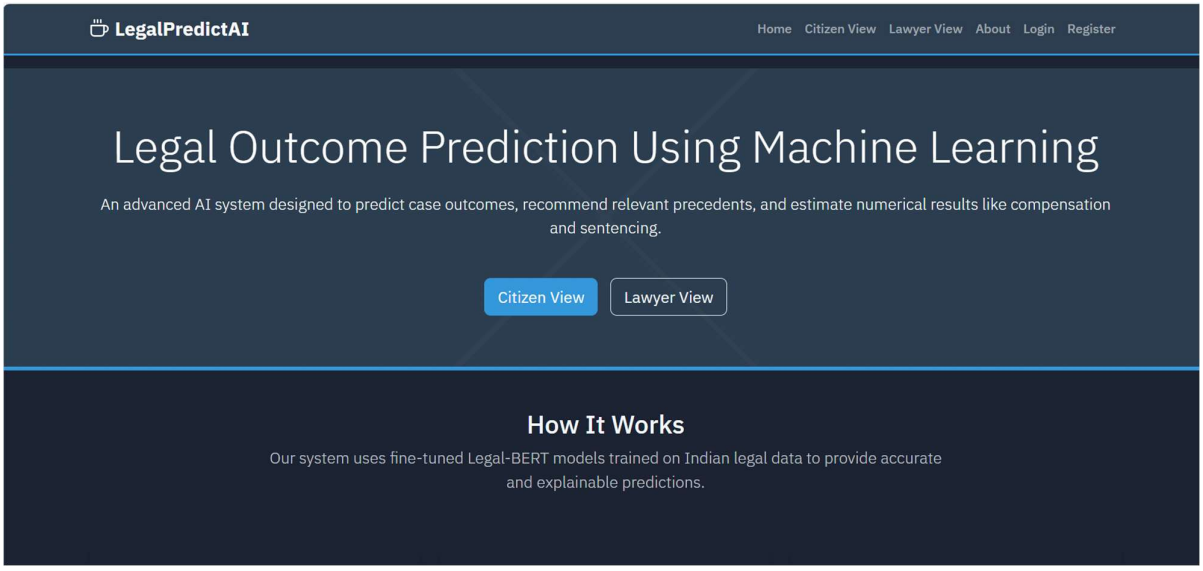


Figure 6.1: Home Page

Figure 6.1 illustrates the Home Page, which serves as the landing interface of the Legal Outcome Prediction System. It introduces users to the platform, highlighting its core functionalities.

Register Page:

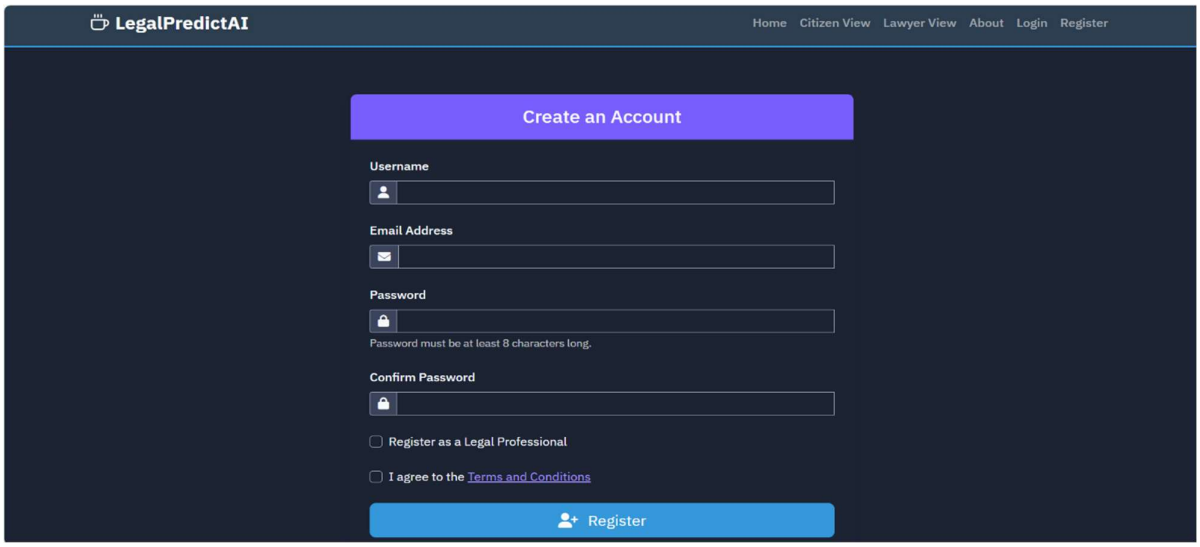


Figure 6.2: Register Page

Figure 6.2 shows the Registration Page, which allows new users either citizens or lawyers to create an account by providing personal and role-specific details necessary for accessing the platform's features.

Login Page:

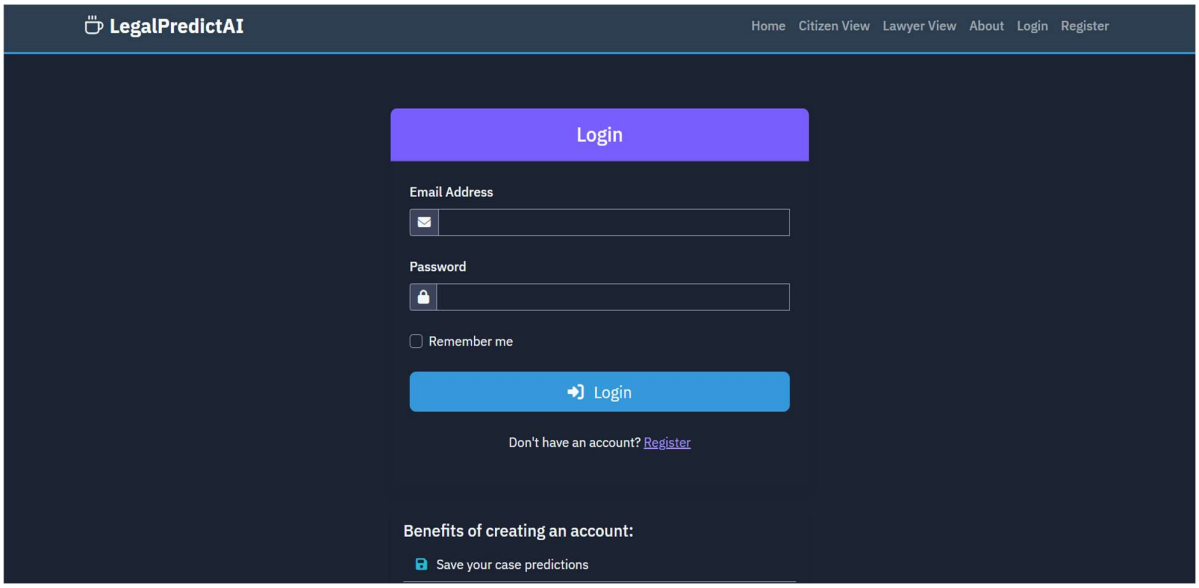


Figure 6.3: Login Page

Figure 6.3 illustrates the Login Page, which enables registered users to securely log in and access their personalized dashboard and legal outcome prediction tools.

Citizen View:

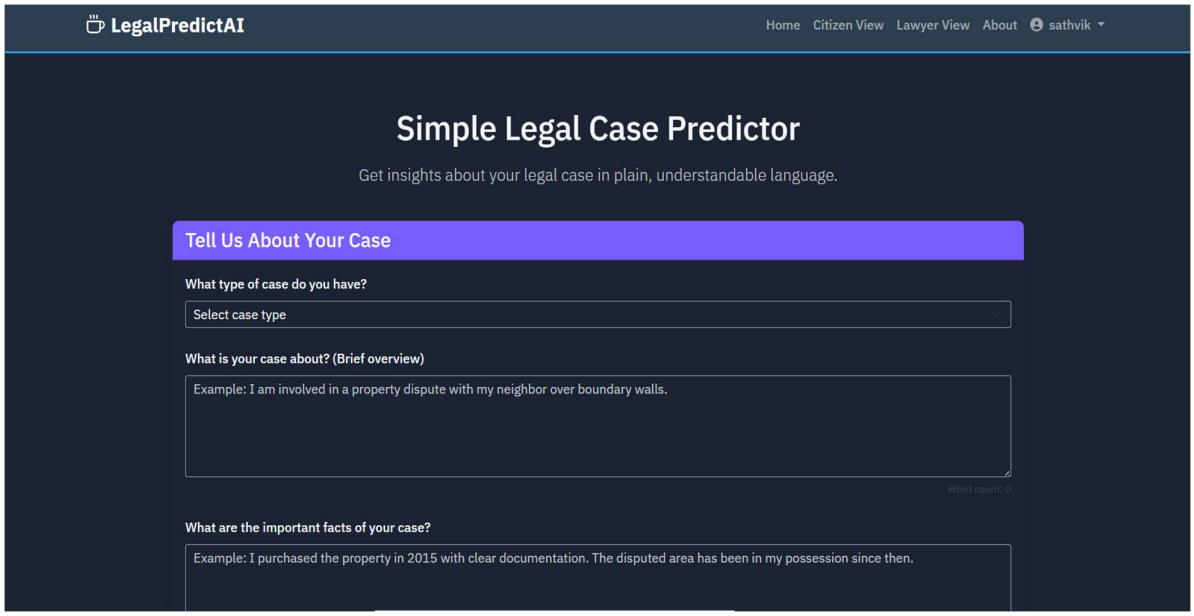


Figure 6.4: Citizen View

Figure 6.4 depicts the Citizen View, which provides citizens with options to submit case summaries, view predicted outcomes, and explore similar past legal cases for better understanding.

Lawyer View:

Figure 6.5: Lawyer View

Figure 6.5 illustrates the Lawyer View, offering lawyers advanced tools to input detailed case information, review predictive insights, and access relevant legal precedents to support their arguments.

Prediction Page:

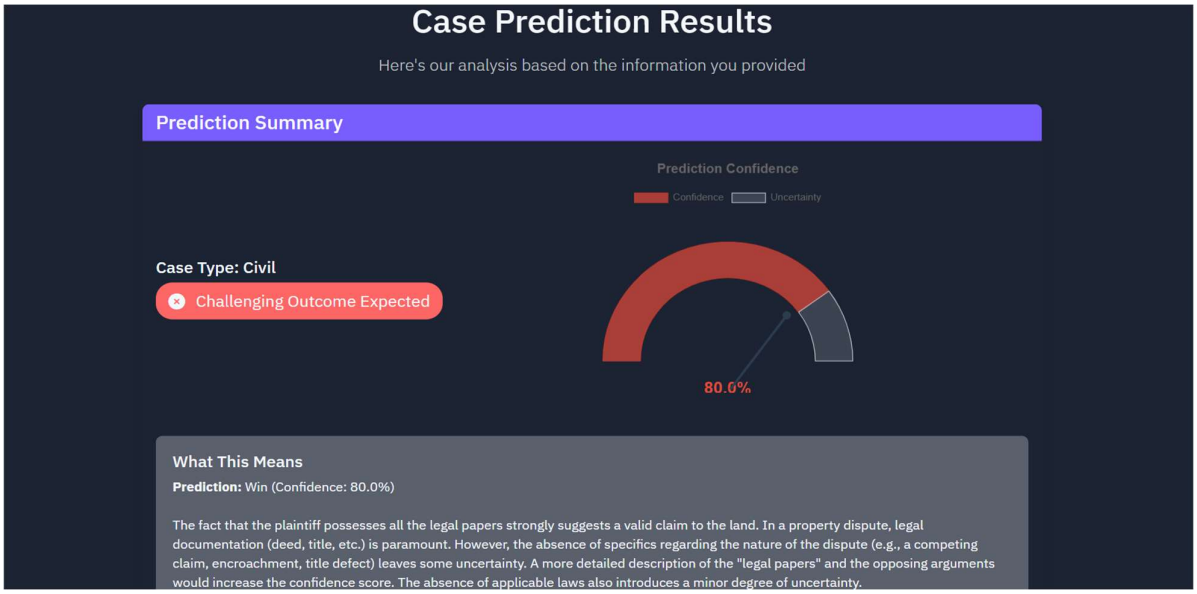


Figure 6.6: Prediction Page

Figure 6.6 shows the Prediction Page, which displays the predicted outcome of the uploaded case along with relevant precedent cases, estimated numerical outcomes and explainability features to enhance transparency.

Training Results:

```
Fold 1 results: {'eval_loss': 1.4986298084259033, 'eval_accuracy': 0.7806060606060606, 'eval_runtime': 15.2201, 'eval_sa
mples_per_second': 54.205, 'eval_steps_per_second': 3.417, 'epoch': 10.0}
Fold 2 results: {'eval_loss': 1.112125039100647, 'eval_accuracy': 0.8121212121212121, 'eval_runtime': 15.1633, 'eval_sam
ples_per_second': 54.408, 'eval_steps_per_second': 3.429, 'epoch': 10.0}
Fold 3 results: {'eval_loss': 1.4257022142410278, 'eval_accuracy': 0.8072727272727273, 'eval_runtime': 15.141, 'eval_sam
ples_per_second': 54.488, 'eval_steps_per_second': 3.434, 'epoch': 10.0}

Fold 4 results: {'eval_loss': 1.4837357997894287, 'eval_accuracy': 0.7951515151515152, 'eval_runtime': 15.1426, 'eval_sa
mples_per_second': 54.482, 'eval_steps_per_second': 3.434, 'epoch': 10.0}

Fold 5 results: {'eval_loss': 1.2566680908203125, 'eval_accuracy': 0.8024242424242424, 'eval_runtime': 15.1842, 'eval_sa
mples_per_second': 54.333, 'eval_steps_per_second': 3.425, 'epoch': 10.0}

Average cross-validation results: {'eval_loss': 1.3553721904754639, 'eval_accuracy': 0.7995151515151515, 'eval_runtime':
15.170240000000002, 'eval_samples_per_second': 54.3832, 'eval_steps_per_second': 3.4278000000000004, 'epoch': 10.0}
```

Figure 6.7: Training Results

Figure 6.7 shows the Model Evaluation Results using 5-fold cross-validation. It reports fold-wise evaluation metrics including loss and accuracy, where the average accuracy achieved across folds is approximately 79.95%, with an average evaluation loss of 1.35. These metrics validate the consistency and performance of the Legal Outcome Prediction model.

7. CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The Legal Outcome Prediction System is a significant step toward integrating machine learning and artificial intelligence into the legal domain. By leveraging domain-specific transformer models such as Legal-BERT, the system effectively predicts case outcomes based on historical Indian court judgments. It also enhances user understanding by retrieving similar precedent cases and estimating numerical outcomes like fines, imprisonment durations, or compensation amounts. The addition of explainability tools such as SHAP ensures that the model's predictions are transparent and interpretable, fostering trust among users. Through a clean, web-based interface, the system makes legal analytics more accessible to a broader audience, including lawyers, law students, and citizens, thereby simplifying legal research and improving decision-making efficiency. Overall, this project demonstrates how advanced natural language processing techniques can be harnessed to bring greater clarity and support within the legal system.

7.2 FUTURE SCOPE

In the future, the Legal Outcome Prediction System has great potential for expansion and refinement. One promising direction is the inclusion of data from various courts across the country, including High Courts and District Courts, to provide broader jurisdictional coverage. Multilingual support can be integrated to handle judgments written in regional Indian languages, ensuring inclusivity across different linguistic demographics. Another valuable enhancement would be predicting case timelines, which would help users estimate how long a case might take to resolve based on historical patterns. Additionally, implementing a legal summarization module could automatically generate concise case briefs, aiding quicker understanding of lengthy judgments. There is also scope to use graph-based methods to analyse how cases cite each other, thereby improving precedent relevance and discovery. A mobile application could be developed to make the system more accessible on smartphones. Integrating real-time legal databases such as Indian Kanoon or Judis can ensure the information stays up to date. Finally, incorporating user feedback mechanisms will allow the system to learn and improve continuously, ensuring its relevance and accuracy over time.

REFERENCES

[1] N. Z. Dina, S. D. Ravana, and N. Idris, "Legal judgment prediction using natural language processing and machine learning methods: A systematic literature review," SAGE Open, vol. 15, no. 2, pp. 13–24, 2025. [Online]. Available: https://journals.sagepub.com/doi/10.1177/21582440251329663

[2] R. Bharati, "Predictive Justice in Indian Courts: Machine Learning Approaches to Case Outcome Forecasting," SSRN, 2025. [Online]. Available: https://ssrn.com/abstract=5089255

[3] Dong Shu, Haoran Zhao, Xukun Liu, David Demeter, Mengnan Du, Yongfeng Zhang, "LawLLM: A large language model for the U.S. legal system," arXiv preprint arXiv:2407.21065, pp. 1–10, 2024. [Online]. Available: https://arxiv.org/abs/2407.21065

[4]Shubham Kumar Nigam, Balaramamahanthi Deepak Patnaik, Shivam Mishra, Noel Shallum, Kripabandhu Ghosh, Arnab Bhattacharya, "TathyaNyaya and FactLegalLlama: Advancing Factual Judgment Prediction in the Indian Legal Domain," arXiv preprint arXiv:2504.04737, 2025. [Online]. Available: https://arxiv.org/abs/2504.04737

[5] Balaramamahanthi Deepak Patnaik, Shivam Mishra, Noel Shallum, Kripabandhu Ghosh "NyayaAnumana & INLegalLlama: The Largest Instruction-Tuned Legal LLMs for Indian Case Outcome Prediction," arXiv preprint arXiv:2412.08385, Dec. 2024. [Online]. Available: https://arxiv.org/abs/2412.08385

[6] Azmi, Meladel Mistica, et al., "Legal judgment prediction via argument analysis," in Frontiers in Artificial Intelligence and Applications, Springer, 2024, pp. 51–65.[Online]. Available: https://dl.acm.org/doi/10.1007/978-981-96-0348-0_4

[7] Zeleznikow, J. (2023, July 1). The benefits and dangers of using machine learning to support making legal predictions. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. John Wiley and Sons Inc. https://doi.org/10.1002/widm.1505

[8] P. Y. Dharma, W. Widyawan, and A. R. Pratama, "Legal judgment prediction: A systematic literature review," in Proc. 2023 ICAMIMIA, Nov. 2023, pp. 1–8, doi: 10.1109/ICAMIMIA60881.2023.10427855. [Online].Available:https://www.researchgate.net/publication/378201556_Legal_Judgment_Prediction_A_Systematic_Literature_Review

[9] J. Mumford, K. Atkinson, and T. Bench-Capon, "Reasoning with legal cases: A hybrid ADF-ML approach," in Legal Knowledge and Information Systems, E. Francesconi et al., Eds., vol. 362, Frontiers in Artificial Intelligence and Applications, IOS Press, 2022, pp. 93–102. https://ebooks.iospress.nl/volumearticle/62039

[10] Silva, H., António, N., Bacao, F. (2022). "A Rapid Semi-automated Literature Review on Legal Precedents Retrieval". In: Progress in Artificial Intelligence. EPIA 2022. Lecture Notes in Computer Science(), vol 13566. Springer, Cham. https://doi.org/10.1007/978-3-031-16474-3_5

[11] G. V. Travaini , M. Bosia, and F. De Micco, “Machine learning and criminal justice: A systematic review of advanced methodology for recidivism risk prediction,” Int. J. Environ. Res. Public Health, vol. 19, p. 10594, pp. 1–18, 2022. [Online]. Available: https://www.mdpi.com/1660-4601/19/17/10594

[12] R. S. de Oliveira, A. S. Reis Jr., and E. G. S. Nascimento, “Predicting the number of days in court cases using artificial intelligence,” PLOS ONE, vol. 17, no. 5, e0269008, pp. 1–12, 2022. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0269008

[13] J. Cui, X. Shen, F. Nie, Z. Wang, J. Wang, and Y. Chen, “A survey on legal judgment prediction: Datasets, metrics, models and challenges,” arXiv preprint arXiv:2206.04022, pp. 1–25, 2022. [Online]. Available: https://arxiv.org/pdf/2204.04859

[14] Almuslim and D. Inkpen, "Legal Judgment Prediction for Canadian Appeal Cases," *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)*, Riyadh, Saudi Arabia, 2022, pp. 163-168, https://ieeexplore.ieee.org/document/9736341

[15] Dong and S. Niu, “Legal judgment prediction via relational learning,” in Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Info. Retr. (SIGIR ’21), Virtual Event, Canada, Jul. 2021, pp. 983–992, https://www.researchgate.net/publication/353191774_Legal_Judgment_Prediction_via_Relational_Learning

[16] K. Shirsat, A. Keni, P. Chavan, and M. Gosavi, “Legal judgment prediction system,” Int. J. Comput. Appl., vol. 183, no. 22, pp. 10–15, 2021. [Online]. Available: https://www.irjet.net/archives/V8/i5/IRJET-V8I5149.pdf

[17] N. A. K. Rosili, R. Hassan, N. H. Zakaria and S. Kasim, “A systematic literature review of machine learning methods in predicting court decisions,” IAES Int. J. Artif. Intell., vol. 10, no. 4, pp. 1091–1102, 2021. [Online]. Available: https://www.researchgate.net/publication/356686477

[18] V. Malik, R. Sanjay, S. K. Nigam, K. Ghosh, S. K. Guha, and A. Modi, “ILDC for CJPE: Indian legal documents corpus for court judgment prediction and explanation,” arXiv preprint arXiv:2105.13562, pp. 1–10, 2021. [Online]. Available: https://arxiv.org/abs/2105.13562

[19] S. Sharma, R. Shandilya, and S. Sharma, “Predicting Indian Supreme Court judgments, decisions, or appeals,” arXiv preprint arXiv:2110.09251, pp. 1–15, 2021. [Online]. Available: https://arxiv.org/abs/2110.09251

[20] Katz DM, Bommarito MJ II, Blackman J (2017) A general approach for predicting the behavior of the Supreme Court of the United States. PLoS ONE 12(4): e0174698. https://doi.org/10.1371/journal.pone.0174698

[21] Hugging Face, “rishiai/indian-court-judgements-and-its-summaries,” Hugging Face, 2025. [Online]. Available: https://huggingface.co/datasets/rishiai/indian-court-judgements-and-its-summaries

APPENDIX A

SOURCE CODE SNIPPETS

```
from datasets import load_dataset, Dataset
import pandas as pd
import numpy as np
from transformers import AutoTokenizer, AutoModelForSequenceClassification, TrainingArguments, Trainer
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
import torch
from sklearn.metrics import accuracy_score
import os
os.environ["WANDB_DISABLED"] = "true"

# Metric function for accuracy
def compute_metrics(pred):
    labels = pred.label_ids
    preds = np.argmax(pred.predictions, axis=1)
    acc = accuracy_score(labels, preds)
    return {"accuracy": acc}

# Load dataset
ds = load_dataset("rishiai/indian-court-judgements-and-its-summaries")
df = ds["train"].to_pandas()
```

```
# Weak labeling function
def infer_outcome(text):
    text = text.lower()
    if "petition allowed" in text or "appeal allowed" in text:
        return "allowed"
    elif "petition dismissed" in text or "appeal dismissed" in text:
        return "dismissed"
    return None

# Apply labeling and clean
df['outcome'] = df['Judgment'].apply(infer_outcome)
df = df.dropna(subset=['outcome'])
df.loc[:, 'outcome'] = df['outcome'].str.lower()

# Encode labels
label_encoder = LabelEncoder()
df['label'] = label_encoder.fit_transform(df['outcome'])

# Use only necessary columns
df = df[['Summary', 'label']].copy()
df = df.dropna(subset=['Summary'])
```

```
# Convert to Hugging Face Dataset
dataset = Dataset.from_pandas(df.reset_index(drop=True))

# Tokenizer & model setup
model_name = "nlpaueb/legal-bert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)

def tokenize(batch):
    return tokenizer(batch['Summary'], padding=True, truncation=True, max_length=512)

dataset = dataset.map(tokenize, batched=True)
dataset.set_format('torch', columns=['input_ids', 'attention_mask', 'label'])

# Cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
results = []
```

```
for fold, (train_idx, val_idx) in enumerate(kf.split(dataset)):
    print(f"Training fold {fold + 1}...")

    train_dataset = dataset.select(train_idx)
    val_dataset = dataset.select(val_idx)

    output_dir = f"./legal_outcome_model_fold_{fold + 1}"

    training_args = TrainingArguments(
    output_dir=output_dir,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=10,
    logging_dir=output_dir,
    logging_steps=50,
    save_steps=500,  # Save every 500 steps
    save_total_limit=2,  # Keep only last 2 checkpoints
    do_eval=True,
    logging_first_step=True,
    report_to=[]
    )
```

```
    # Fresh model each fold
    model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=2)

    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        compute_metrics=compute_metrics ,
        eval_dataset=val_dataset
    )

    trainer.train()
    eval_results = trainer.evaluate()
    print(f"Fold {fold + 1} results:", eval_results)
    results.append(eval_results)

# Average results
average_results = {metric: np.mean([result[metric] for result in results]) for metric in results[0]}
print("\nAverage cross-validation results:", average_results)
```

```
import os
import logging
import random
import json
from datetime import datetime
from bson.objectid import ObjectId
from flask import Flask, render_template, request, redirect, url_for, flash, session, jsonify
from flask_pymongo import PyMongo
from werkzeug.security import generate_password_hash, check_password_hash

# Set up logging
logging.basicConfig(level=logging.DEBUG)

# Create Flask app
app = Flask(__name__)
app.secret_key = os.environ.get("SESSION_SECRET", "dev_secret_key")

# MongoDB Configuration
mongodb_uri = os.environ.get("MONGODB_URI", "mongodb://localhost:27017/legal_prediction")
print(f"DEBUG: MongoDB URI: {mongodb_uri}")

# Ensure proper MongoDB URI format
if mongodb_uri and "mongodb+srv" in mongodb_uri:
    app.config["MONGO_URI"] = mongodb_uri
else:
    # Fallback to a local MongoDB for development
    app.config["MONGO_URI"] = "mongodb://localhost:27017/legal_prediction"
    print("DEBUG: Using local MongoDB fallback")
```

```
try:
    mongo = PyMongo(app)
    # Test the connection
    mongo.db.command('ping')
    print("DEBUG: MongoDB connection successful")
except Exception as e:
    print(f"ERROR: MongoDB connection failed: {str(e)}")
    # Create a mock MongoDB for development if connection fails
    from unittest.mock import MagicMock
    mongo = MagicMock()
    mongo.db.users = MagicMock()
    mongo.db.cases = MagicMock()
    mongo.db.predictions = MagicMock()
    # Setup mock functions
    mongo.db.users.find_one.return_value = None
    mongo.db.users.insert_one.return_value = MagicMock(inserted_id="mock_id")
    mongo.db.cases.insert_one.return_value = MagicMock(inserted_id="mock_case_id")
    mongo.db.predictions.insert_one.return_value = None

    # Create a mock ObjectId function
    class MockObjectId:
        def __call__(self, id_str):
            return id_str

    # Replace the original ObjectId with our mock version
    global ObjectId
    ObjectId = MockObjectId()

    print("DEBUG: Using mock MongoDB for development")
```

```
# Import prediction utilities
from prediction_utils import predict_outcome, get_case_precedents, predict_numerical_values
from explainable_ai import generate_explanation, generate_shap_visualization, generate_lime_explanation

# Helper functions for session-based auth
def is_logged_in():
    return 'user_id' in session

def get_current_user():
    if not is_logged_in():
        return None
    return mongo.db.users.find_one({'_id': ObjectId(session['user_id'])})

# Routes
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')
        user = mongo.db.users.find_one({'email': email})

        if user and check_password_hash(user['password_hash'], password):
            session['user_id'] = str(user['_id'])
            session['username'] = user['username']
            session['is_lawyer'] = user['is_lawyer']
            next_page = request.args.get('next')
            return redirect(next_page or url_for('index'))
        else:
            flash('Login unsuccessful. Please check your email and password.', 'danger')

    return render_template('login.html')
```

```python
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form.get('username')
        email = request.form.get('email')
        password = request.form.get('password')
        is_lawyer = 'is_lawyer' in request.form

        # Check if user already exists
        if mongo.db.users.find_one({'email': email}):
            flash('Email already registered.', 'warning')
            return redirect(url_for('register'))

        # Create new user
        new_user = {
            'username': username,
            'email': email,
            'password_hash': generate_password_hash(password),
            'is_lawyer': is_lawyer,
            'created_at': datetime.utcnow()
        }

        user_id = mongo.db.users.insert_one(new_user).inserted_id

        flash('Registration successful. Please log in.', 'success')
        return redirect(url_for('login'))

    return render_template('register.html')

@app.route('/logout')
def logout():
    session.clear()
    flash('You have been logged out.', 'info')
    return redirect(url_for('index'))
```

```python
@app.route('/citizen')
def citizen_view():
    if not is_logged_in():
        flash('Please log in to access this page.', 'warning')
        return redirect(url_for('login', next=request.url))
    return render_template('citizen_view.html')

@app.route('/lawyer')
def lawyer_view():
    if not is_logged_in():
        flash('Please log in to access this page.', 'warning')
        return redirect(url_for('login', next=request.url))

    if not session.get('is_lawyer'):
        flash('You need to be registered as a legal professional to access this page.', 'warning')
        return redirect(url_for('index'))

    return render_template('lawyer_view.html')
```

```
def predict():
    # Get form data
    case_type = request.form.get('case_type')
    case_details = request.form.get('case_details')
    facts = request.form.get('facts')
    arguments = request.form.get('arguments') or ""
    applicable_laws = request.form.get('applicable_laws') or ""
    user_type = request.form.get('user_type', 'citizen')

    # Combine all text for prediction
    combined_text = f"Case Type: {case_type}\nCase Details: {case_details}\nFacts: {facts}\nArguments:
        {arguments}\nApplicable Laws: {applicable_laws}"

    # Make predictions
    outcome, probability = predict_outcome(combined_text)
    relevant_cases = get_case_precedents(combined_text, case_type)
    numerical_predictions = predict_numerical_values(combined_text, case_type)

    # Generate explanations
    explanation = generate_explanation(combined_text, outcome)
    shap_visualization = generate_shap_visualization(combined_text)
    lime_explanation = generate_lime_explanation(combined_text)
```

```
if is_logged_in():
    user_id = session['user_id']

    # Create case document
    new_case = {
        'user_id': ObjectId(user_id),
        'case_type': case_type,
        'case_details': case_details,
        'facts': facts,
        'arguments': arguments,
        'applicable_laws': applicable_laws,
        'created_at': datetime.utcnow()
    }

    case_id = mongo.db.cases.insert_one(new_case).inserted_id

    # Create prediction document
    new_prediction = {
        'case_id': case_id,
        'predicted_outcome': outcome,
        'confidence': probability,
        'numerical_predictions': str(numerical_predictions),
        'explanation': explanation,
        'created_at': datetime.utcnow()
    }

    mongo.db.predictions.insert_one(new_prediction)

    # Store case_id in session for redirect
    session['last_case_id'] = str(case_id)
```

```python
@app.route('/results')
def results():
    # Get prediction data
    if is_logged_in() and session.get('last_case_id'):
        # Get specific case result for logged in user
        case_id = session.get('last_case_id')
        user_id = session['user_id']

        case = mongo.db.cases.find_one({
            '_id': ObjectId(case_id),
            'user_id': ObjectId(user_id)
        })

        if not case:
            flash('Case not found.', 'warning')
            return redirect(url_for('index'))

        prediction = mongo.db.predictions.find_one({'case_id': ObjectId(case_id)})

        if not prediction:
            flash('Prediction not found.', 'warning')
            return redirect(url_for('index'))

        # Format data for template
        result_data = {
            'case_type': case['case_type'],
            'outcome': prediction['predicted_outcome'],
            'probability': prediction['confidence'],
            'numerical_predictions': eval(prediction['numerical_predictions']),
            'explanation': prediction['explanation'],
            'relevant_cases': get_case_precedents(case['case_details'], case['case_type']),
            'shap_visualization': generate_shap_visualization(case['case_details']),
            'lime_explanation': generate_lime_explanation(case['case_details'])
        }
```

```python
        # Clear the last_case_id from session
        session.pop('last_case_id', None)

    elif 'last_prediction' in session:
        # Use session data for non-logged in users
        result_data = session['last_prediction']

    else:
        # No prediction data available
        flash('No prediction data found. Please submit a case first.', 'warning')
        return redirect(url_for('index'))

    print(f"Probability passed to template: {result_data.get('probability')}")
    return render_template('results.html', result=result_data)

@app.route('/about')
def about():
    return render_template('about.html')
```

```python
@app.route('/my_predictions')
def my_predictions():
    if not is_logged_in():
        flash('Please log in to view your predictions.', 'warning')
        return redirect(url_for('login', next=request.url))

    user_id = session['user_id']

    try:
        cases = list(mongo.db.cases.find({'user_id': ObjectId(user_id)}).sort('created_at', -1))
        case_ids = [case['_id'] for case in cases]
        predictions = {p['case_id']: p for p in mongo.db.predictions.find({'case_id': {'$in': case_ids}})}

        # Merge cases and predictions
        results = []
        for case in cases:
            prediction = predictions.get(case['_id'])
            if prediction:
                results.append({
                    'case_type': case['case_type'],
                    'case_details': case['case_details'],
                    'created_at': case['created_at'],
                    'predicted_outcome': prediction['predicted_outcome'],
                    'confidence': prediction['confidence'],
                    'numerical_predictions': eval(prediction['numerical_predictions']),
                    'case_id': str(case['_id'])
                })
    except Exception as e:
        logging.error(f"Error fetching predictions: {str(e)}")
        flash("Failed to load your predictions. Please try again later.", "danger")
        return redirect(url_for('index'))

    return render_template('my_predictions.html', results=results)
```

```python
import os
import random
import json
import torch
from transformers import AutoTokenizer, AutoModelForMaskedLM
from transformers import BertForSequenceClassification
model = BertForSequenceClassification.from_pretrained('./final_model')
# Define Device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
from transformers import AutoTokenizer, AutoModelForSequenceClassification
tokenizer = AutoTokenizer.from_pretrained("law-ai/CustomInLawBERT")
# model = AutoModelForSequenceClassification.from_pretrained("law-ai/CustomInLawBERT")  # Update this
model.to(device)
model.eval()
# Prediction Function for Legal Outcome
def predict_outcome(case_text):
    # Tokenize input text
    inputs = tokenizer(case_text, return_tensors="pt", truncation=True, padding=True, max_length=512)
    inputs = {k: v.to(device) for k, v in inputs.items()}
    with torch.no_grad():
        # Get the model's output
        outputs = model(**inputs)
        logits = outputs.logits
        # Apply sigmoid for binary classification
        probs = torch.sigmoid(logits)
        print(probs.shape)  # Check the shape of the output tensor
        predicted_class = probs.argmax(dim=-1).item()  # Get the index of the highest probability
        # Get confidence score
        confidence = probs[0][predicted_class].item()
    # Map the predicted class to a label (optional)
    label = "Win" if predicted_class == 1 else "Lose"
    return label, round(confidence, 2)
# Example Usage
case_text = "The defendant has been accused of fraud..."
label, confidence = predict_outcome(case_text)
print(f"Prediction: {label}, Confidence: {confidence}")
```

```python
def get_case_precedents(case_text, case_type=None, num_precedents=5):
    # Embed the input case text
    query_embedding = embedding_model.encode([case_text])

    # Search the FAISS index
    distances, indices = faiss_index.search(query_embedding, num_precedents)

    precedents = []
    for i, dist in zip(indices[0], distances[0]):
        precedents.append({
            "summary": precedent_summaries[i],
            "judgment": precedent_judgments[i],
            "relevance": round(float(1 / (1 + dist)), 4)  # Normalize as relevance score
        })

    return precedents
```

APPENDIX B

SOFTWARE INSTALLATION AND PROJECT EXECUTION STEPS:

Step 1: Open Kaggle → Create New Notebook

Step 2: Install required packages (if not preinstalled):

```
!pip install transformers datasets faiss-cpu shap lime
```

Step 3: Load and Prepare Dataset:

```
from datasets import load_dataset
dataset = load_dataset("rishiai/indian-court-judgements-and-its-summaries")
df = dataset['train'].to_pandas()
```

Step 4: Preprocess the Data: Summary extraction, convert outcome as binary and tokenize the data.

Step 5: Fine -Tune the model:

```
model = AutoModelForSequenceClassification.from_pretrained("nlpaueb/legal-bert-base-uncased")
```

Step 6: Save and download the model (model.save_pretrained).

Step 7: Open command prompt (or terminal) and install the required libraries one by one:

```
pip install flask
pip install torch
pip install transformers
pip install datasets
pip install faiss-cpu
pip install shap
pip install lime
pip install pandas numpy
```

Step 8: Make sure your project folder contains all files.

Step 9: In your terminal run:

```
python app.py
```

Step 10: Open your browser and go to http://127.0.0.1:5000