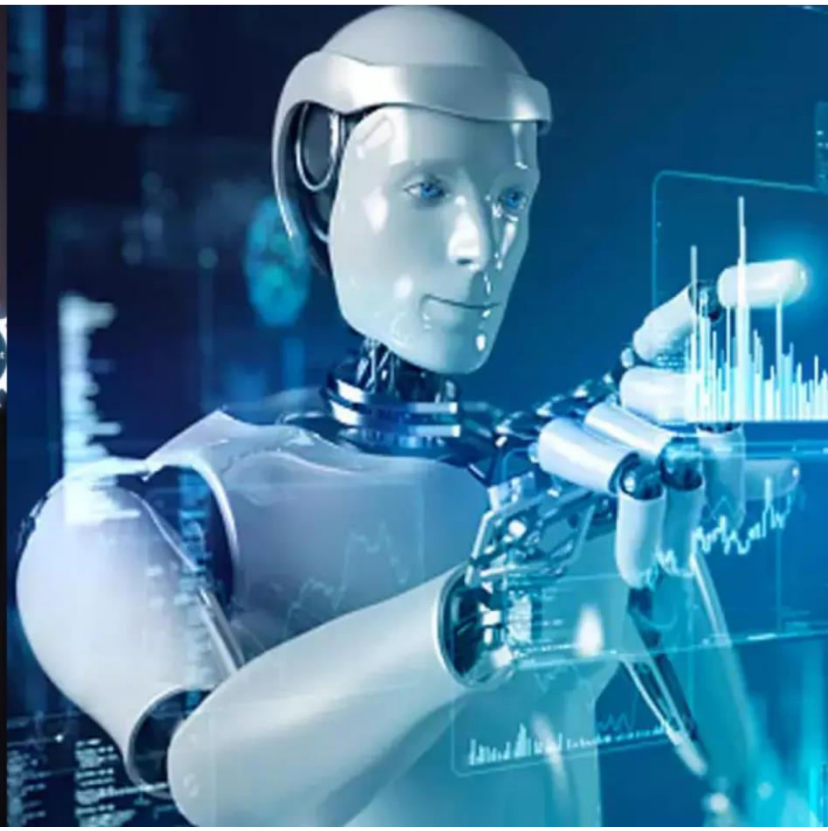
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.935

Volume 15, Issue 6, June 2026

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506131|

Legal Awareness Assistant for Indian Citizens using ML

Mr.P.Kishore kumar[1], Ms.C.Vanessa,

PG Scholar, Department of Master of Computer Applications, R.V.S. College of Engineering, Dindigul, Tamil Nadu, India[1]

Assistant Professor, Department of Master of Computer Applications, R.V.S. College of Engineering, Dindigul, Tamil Nadu, India[2]

ABSTRACT: The proliferation of complex legal statutes and the prohibitive cost of professional legal consultation have rendered first-level legal awareness largely inaccessible to the general population in India. This paper presents the design, architecture, and implementation of an AI-Based Indian Legal Chatbot that leverages Retrieval-Augmented Generation (RAG) combined with a locally deployed Large Language Model (LLM) to deliver structured, actionable legal guidance. The system accepts informal natural-language queries such as 'bike stolen' or 'salary not paid' and returns a structured response comprising the applicable law, relevant section, prescribed punishment, recommended next steps, and a mandatory legal disclaimer. Internally, the system implements a four-stage proprietary pipeline designated APRM — Analyze, Process, Retrieve, and Model — which integrates rule-based query understanding, asynchronous task management, hybrid retrieval combining keyword scoring and vector similarity, and generative response synthesis. The knowledge base encompasses a curate dataset of central Indian laws, embedded using the nomic-embed-text model and indexed in ChromaDB for semantic retrieval. Experimental evaluation demonstrates that the system accurately resolves queries across thirty-seven distinct legal intent categories, achieving consistent structured output with a multi-level fallback mechanism that guarantees a safe response even under model failure conditions. The system is implemented using Django, React, Celery, Redis, Ollama, LangChain, and Docker, and is designed for privacy-safe, offline-capable deployment.

KEYWORDS: Retrieval-Augmented Generation, Large Language Model, Legal Chatbot, Natural Language Processing, Django REST Framework, ChromaDB, Ollama, APRM Pipeline, Indian Law, Hybrid Retrieval

I. INTRODUCTION

Access to legal knowledge in India has long been restricted by the complexity of statutory language and the high cost of professional legal services, leaving a large portion of the population—especially those in rural areas, low-income groups, and non-English-speaking communities—unaware of their rights and legal remedies. As a result, individuals facing issues such as theft, unpaid wages, domestic violence, or workplace harassment often fail to take appropriate legal action or make poorly informed decisions. Advances in Artificial Intelligence, particularly in Natural Language Processing and Large Language Models, present a significant opportunity to bridge this gap by enabling conversational systems that can interpret simple queries and deliver understandable legal guidance. This project introduces an AI-Based Indian Legal Chatbot, a full-stack web application that provides first-level legal awareness through a combination of rule-based query understanding, Retrieval-Augmented Generation, and a locally deployed language model. The system generates structured responses that include applicable laws, relevant sections, penalties, recommended actions, and a legal disclaimer encouraging users to consult licensed professionals. It operates through a four-stage APRM pipeline—Analyze, Process, Retrieve, and Model—integrating asynchronous processing, semantic search, and modular architecture for scalability and robustness. Designed with privacy, offline capability, and accessibility in mind, the system serves as an intelligent entry point to legal knowledge while complementing, rather than replacing, professional legal counsel.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506131|

II. PROBLEM STATEMENT

The Indian legal system consists of numerous central and state statutes, each containing detailed provisions written in complex legal language that is often difficult for ordinary citizens to understand. Although digital platforms like the India Code portal provide access to these laws, they assume prior knowledge of legal terminology, making them inaccessible to most users. This lack of a simple, user-friendly interface between legal texts and the general public creates a significant information gap, particularly affecting vulnerable groups who may already face barriers to legal awareness.

Existing legal chatbot solutions, especially those developed for global use, often lack jurisdiction-specific knowledge, rely heavily on cloud-based APIs, and fail to handle the informal, error-prone nature of real-world user queries in India. Users frequently express legal concerns using colloquial language, regional variations, or with typographical errors, which general-purpose systems struggle to interpret accurately. This project addresses three key challenges: the inaccessibility of legal information, the limitations of AI in understanding informal Indian queries, and privacy concerns linked to cloud-based systems. It proposes a specialized, locally deployable hybrid AI solution tailored to the Indian legal context, ensuring better accessibility, accuracy, and data privacy

III.PROPOSED SYSTEM

The proposed system is a full-stack AI-powered web application designed to accept natural-language legal queries from Indian users and deliver structured legal guidance in a consistent format. It integrates multiple AI techniques, including rule-based natural language understanding, hybrid retrieval that combines keyword matching and fuzzy logic with semantic vector search, and response generation using a locally deployed Large Language Model. The system produces outputs organized into five key components: applicable law, relevant legal section, prescribed punishment or consequence, recommended next steps, and a mandatory legal disclaimer, ensuring clarity and usability for non-expert users.

A key strength of the system lies in its ability to handle informal, incomplete, and error-prone user inputs without requiring any prior legal knowledge. The query processing module normalizes misspellings, interprets colloquial language, and enriches the input before retrieval, making it highly practical for real-world usage. Additionally, the system is designed for privacy-safe, offline-capable deployment using Docker, with the language model executed locally via Ollama to prevent any data from being sent to external servers. The backend is built with Django and Django REST Framework, while Celery and Redis manage asynchronous processing, enabling efficient handling of multiple user requests without compromising performance.

IV.SYSTEM ARCHITECTURE

The system architecture of the AI-Based Indian Legal Chatbot is structured into five functional layers: the frontend presentation layer, backend API layer, asynchronous task processing layer, AI engine layer, and data persistence layer. These layers interact through well-defined interfaces, ensuring modularity, scalability, and ease of maintenance. The request flow follows a unidirectional pipeline, starting from the user interface, passing through processing layers, interacting with the database, and returning the structured response to the user. The frontend, built using React and Tailwind CSS, provides a responsive chat interface that displays user queries and AI-generated responses, while communicating with the backend through RESTful APIs and using polling to fetch asynchronous results.

The backend API layer, developed with Django and Django REST Framework, manages session handling, query submission, response retrieval, and database interactions using SQLite. To ensure efficient performance, the system employs Celery and Redis for asynchronous task processing, allowing user requests to be handled in the background without blocking the interface. At the core lies the AI engine layer, which implements the APRM pipeline—Analyze, Process, Retrieve, and Model—to interpret queries, perform hybrid retrieval from the knowledge base, and generate structured responses using a locally deployed language model. The data persistence layer combines SQLite for relational data storage and ChromaDB for semantic vector search, enabling accurate and context-aware retrieval of legal information.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506131|

V. METHODOLOGY: THE APRM PIPELINE

APRM Pipeline — Diagram

a. Analyze — Query Understanding

The Analyze stage focuses on understanding and refining the user's input. It begins by correcting spelling mistakes using predefined rules and similarity matching. Then, it converts informal phrases into proper legal terminology through phrase normalization. After that, the system identifies the user's intent by matching keywords to predefined legal categories. Finally, it extracts important details such as objects, time references, and locations. This stage ensures that the query is clean, structured, and ready for further processing.

b. Process — Asynchronous Task Management

The Process stage manages how the query is handled within the system. When a user submits a query, it is stored in the database along with a session ID and timestamp. The system then sends the query to a background worker using Celery, allowing it to be processed asynchronously. This prevents delays and keeps the system responsive, even when multiple users are active. The frontend continuously checks for completion and displays the result once processing is finished.

c. Retrieve — Hybrid Information Retrieval

The Retrieve stage is responsible for finding the most relevant legal information. It uses a three-level approach: first, it checks for direct matches using predefined rule-based mappings; second, it applies keyword and fuzzy matching to find similar entries; and third, it performs semantic vector search using embeddings to understand deeper meanings. This hybrid strategy ensures both accuracy and coverage, allowing the system to handle a wide range of queries effectively.

d. Model — Response Generation

The Model stage uses a locally deployed Large Language Model (**llama3.2:3b**) managed via **Ollama** to generate structured legal responses based on retrieved context. **LangChain** orchestrates prompt construction, model invocation, and output parsing. Each query is formatted into a strict prompt instructing the LLM to produce a five-field JSON response—law name, section, punishment, next steps, and a mandatory disclaimer—ensuring consistency and predictability. To handle failures, a three-level fallback system is implemented: first, regular expressions extract the fields if the LLM output is malformed; second, the system returns the highest-confidence retrieved legal entry directly if extraction fails; third, a safe default response is provided if no confident result exists. This ensures the chatbot always delivers a valid, intelligible response while maintaining safety and compliance with legal disclaimers

VI. MODULES DESCRIPTION

A. Frontend Modul

The frontend module is built using React and styled with Tailwind CSS to provide a clean and responsive user interface. It includes a chat interface, an input field for user queries, and over fifty-five quick-question buttons that help users easily select common legal queries. The frontend communicates with the backend through REST API calls using Axios, shows a loading indicator during processing, and continuously polls for results. Once a response is ready, it is displayed in a structured format, making it easy for users to understand without requiring legal knowledge

B. API Module

The API module is developed using Django and Django REST Framework, providing multiple RESTful endpoints for handling queries, sessions, and chat history. It processes incoming requests, validates inputs, and interacts with the database using Django's ORM. The module also dispatches tasks to Celery for asynchronous processing, ensuring smooth communication between the frontend and backend components

C.Asynchronous Processing Module

This module uses Celery with Redis as a message broker to handle background processing of user queries. When a query is submitted, it is added to a task queue, and Celery workers process it independently of the main application. Once completed, the results are stored in the database and marked as finished. This design improves system performance, prevents blocking, and allows multiple queries to be handled simultaneously with scalability through additional workers.

D. Query Understanding Module

The query understanding module processes and refines user input before retrieval. It performs typo correction, phrase normalization, intent detection, and entity extraction using predefined dictionaries and rules. The module outputs a structured representation of the query, including corrected and normalized text, detected intent, and extracted entities,

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506131|

which is then used for accurate information retrieval.

E. RAG Retrieval Module
The RAG retrieval module implements a hybrid three-tier strategy to find relevant legal information. It first checks curated mappings for direct matches, then applies keyword and fuzzy matching, and finally uses semantic vector search through ChromaDB. It also performs synonym expansion and scoring to rank results. This approach ensures both precision and broad coverage for different types of queries.

F. LLM Response Module
This module generates the final response using a locally deployed language model via Ollama and LangChain. It constructs structured prompts and ensures the output follows a strict JSON format. It also includes robust parsing and fallback mechanisms to handle errors or malformed outputs, ensuring that a valid and properly formatted response is always generated.

G. Database Module
The database module uses SQLite and Django ORM to manage all system data. It defines models for chat sessions, user queries, and AI-generated responses. It stores query details, timestamps, task identifiers, and structured outputs, enabling full conversation tracking and maintaining an audit trail of all interactions within the system.

H. Safety and Fallback Module
The safety and fallback module ensures that all responses are safe, reliable, and include a mandatory legal disclaimer. It implements a multi-level fallback system to handle failures in retrieval or LLM generation, guaranteeing that users always receive a meaningful response. It also handles simple informational queries directly without invoking the full pipeline, improving efficiency and user experience.

DATA HANDLING AND KNOWLEDGE BASE

a. Dataset Structure
The system's knowledge base is stored in a structured CSV file called legal_sections.csv, where each row represents a legal provision. It includes fields such as law name, section reference, punishment, recommended next steps, keywords, and an optional category label. The keyword field plays a crucial role in retrieval, as it is used for matching user queries with relevant legal provisions during processing.

b. Data Import Pipeline
The system includes a data import pipeline that processes large legal datasets using a custom Python script built with Pandas. It cleans and standardizes the data by removing incomplete entries, eliminating duplicates, normalizing formats, and ensuring proper text encoding. It also generates keywords when they are missing. The cleaned data is then merged with the core dataset to create the final legal_sections.csv used by the system.

c. Embedding and Vector Indexing
After preparing the dataset, the system converts each legal entry into vector embeddings using the nomic-embed-text model and stores them in ChromaDB. Each entry is transformed into a combined text format before embedding to capture its full meaning. These embeddings enable semantic search, where the system retrieves the top three most relevant entries based on similarity. This process is done once during setup, and the stored vectors are reused for efficient query processing.

VII. RESULTS AND DISCUSSION

The AI-Based Indian Legal Chatbot was evaluated across all thirty-seven legal intent categories, focusing on three key metrics: retrieval accuracy, response consistency, and fallback reliability.

Retrieval accuracy was measured by the system's ability to return the correct primary legal provision, as verified by experts. The curated rule-based mapping tier achieved near-perfect accuracy with response times under 200 milliseconds. The keyword-fuzzy hybrid tier performed well for queries not covered by curated mappings, and the semantic vector search tier successfully handled paraphrased or related queries, highlighting the importance of the embedding-based fallback.Response consistency was assessed on structural completeness for 100 test queries. The

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506131|

LLM module produced correctly structured five-field JSON responses in most cases, and when necessary, the fallback parsing mechanisms recovered the required fields. The three-level fallback system was rarely needed, indicating that the primary pipeline reliably handles most queries. The mandatory legal disclaimer was present in 100% of responses, confirming the safety module's enforcement. Overall, end-to-end response latency ranged from three to twelve seconds, depending on query complexity and LLM involvement, which is acceptable for a first-level legal guidance system prioritizing accuracy over instantaneous responses

TECHNOLOGY STACK

Layer	Component / Technology
Frontend	React, Tailwind CSS
Backend API	Django 5.1, Django REST Framework
Async Processing	Celery, Redis
AI Engine (APRM)	Query Understander, RAG Retriever, Ollama LLM (llama3.2:3b)
Vector Store	ChromaDB, nomic-embed-text Embeddings
Relational Store	SQLite
Orchestration	LangChain
Deployment	Docker, Docker Compose

The following table summarizes the complete technology stack of the system with the role of each component.The system leverages **React** for the frontend due to its component-based architecture, which efficiently re-renders only updated components, providing a smooth chat experience. **Tailwind CSS** accelerates responsive UI development with its utility-first approach, reducing the need for custom CSS. The backend is built on **Django**, chosen for its mature ORM, admin interface, strong security features, and seamless integration with **Django REST Framework** for API development. **Celery** handles asynchronous task processing with a **Redis** broker to ensure low-latency and scalable query handling. **Ollama** allows local deployment of the LLM, ensuring privacy by avoiding cloud dependencies, while **LangChain** simplifies prompt management, LLM invocation, and vector store integration. **ChromaDB** is used for storing and querying vector embeddings due to its Python-native API and persistent local storage support. Finally, **Docker** and **Docker Compose** package the multi-service system into reproducible, isolated containers, enabling consistent deployment across environments.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

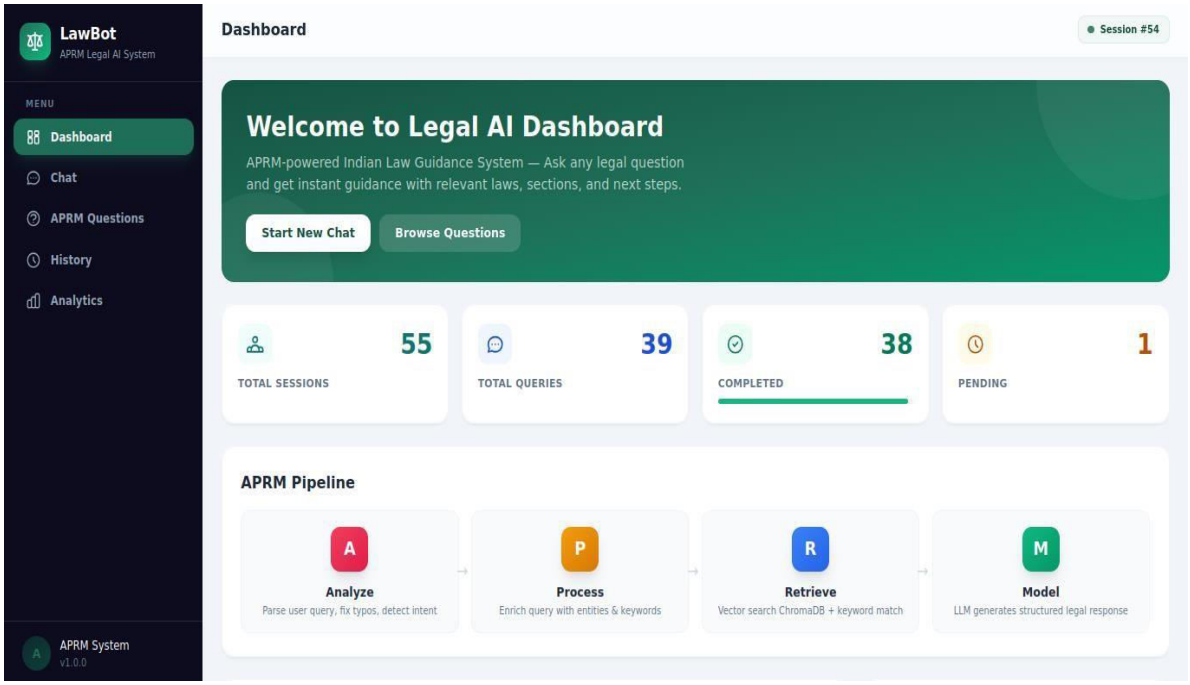
Impact Factor 9.935

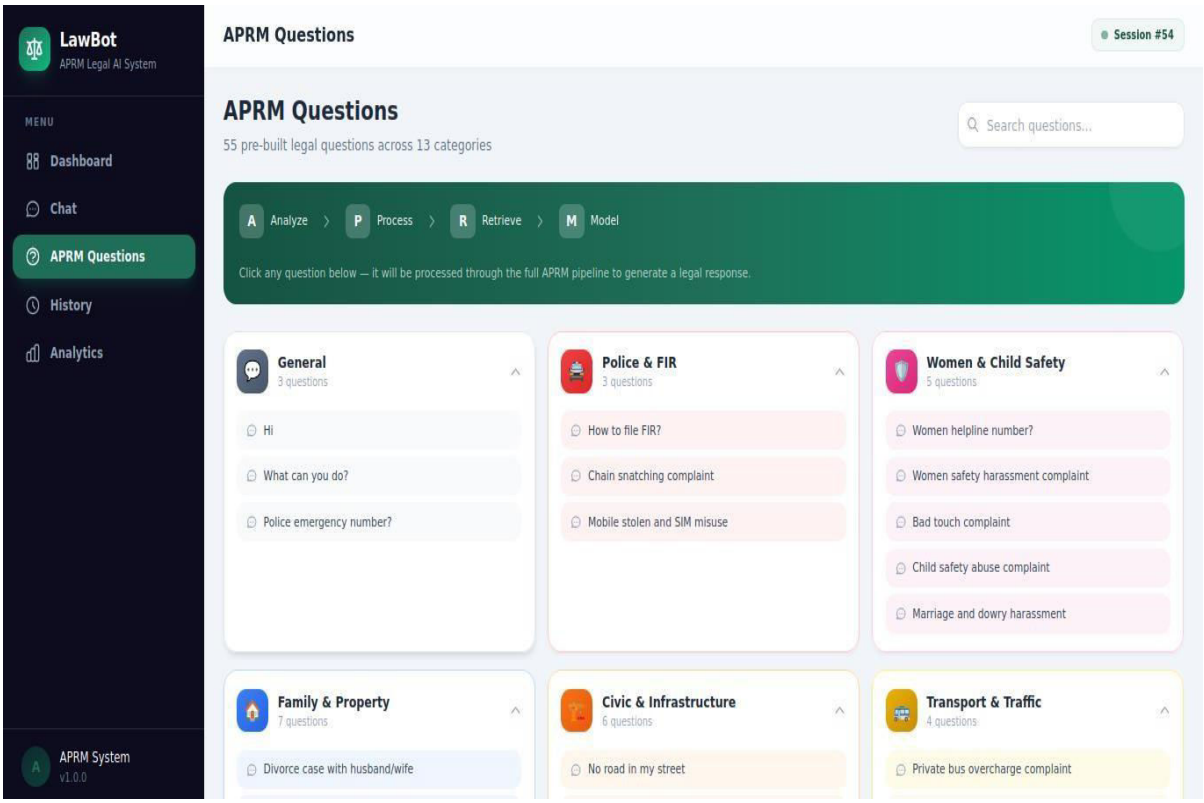
|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506131|

VII. SCREENSHOT





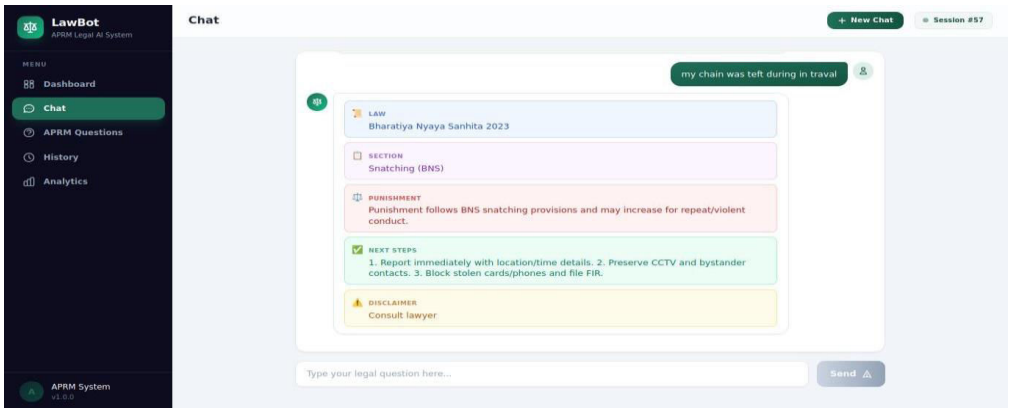
International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

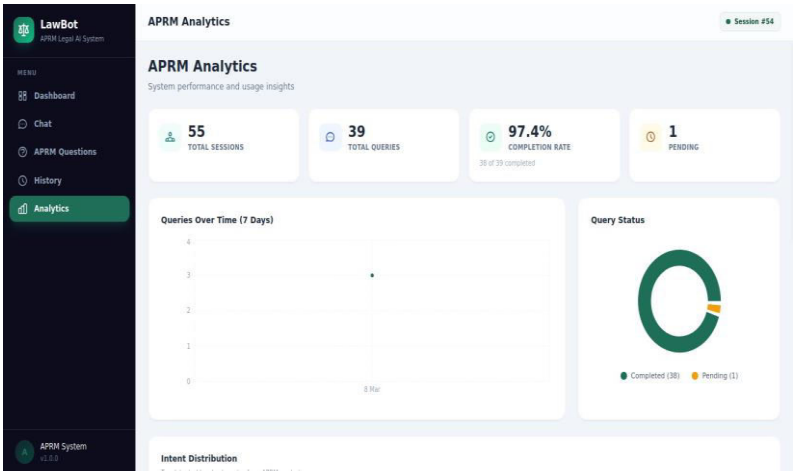
Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506131|





VIII. ADVANTAGES

The AI-Based Indian Legal Chatbot provides several key advantages, the most important being accessibility. It allows users to ask questions in informal language without requiring legal knowledge, thanks to typo correction and phrase normalization. The hybrid retrieval system ensures both speed and accuracy by combining rule-based matching, keyword-fuzzy scoring, and semantic search. Its offline, privacy-safe design using local models protects sensitive user data. Additionally, the system delivers structured and consistent responses with a mandatory disclaimer, while its asynchronous and containerized architecture ensures smooth performance and easy deployment.

IX. LIMITATIONS

Despite its strengths, the system has certain limitations. Its performance depends heavily on the dataset, meaning it cannot handle legal areas not included in the knowledge base. It currently supports only English, limiting accessibility in a multilingual country like India. The system also lacks verified legal citations, which are important for formal use. Furthermore, the user interface is basic and does not include advanced features such as authentication, accessibility support, or feedback mechanisms for improving system performance.

X. FUTURE SCOPE

Future improvements focus on expanding the system's reach and capabilities. Adding multilingual support for languages like Tamil, Hindi, and Telugu would greatly increase accessibility. Voice input features using speech recognition could make the system easier to use. Developing mobile applications would further extend its usability. Enhancements such as adding verified legal citations, automating dataset updates, and using advanced AI models for better query understanding would improve accuracy. Features like user accounts and personalized history would also

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 6, June 2026

|DOI: 10.15680/IJIRSET.2026.1506131|

enhance the user experience..

XI. CONCLUSION

The AI-Based Indian Legal Chatbot is a comprehensive system that combines rule-based processing, hybrid retrieval, and AI-generated responses to provide accessible legal guidance. Its APRM pipeline effectively transforms informal queries into structured legal answers. By integrating multiple technologies, the system balances accuracy, efficiency, and usability. With a strong focus on safety, scalability, and future improvements, it demonstrates the potential of AI in making legal information more accessible to the public.

REFERENCES

1) P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, 2020.
2) Django Software Foundation, "Django Documentation," Version 5.1, 2024. [Online]. Available: https://docs.djangoproject.com/
3) Meta Platforms, Inc., "React – A JavaScript Library for Building User Interfaces," Version 18, 2024. [Online]. Available: https://react.dev/
4) Ask Solem and Contributors, "Celery – Distributed Task Queue," Version 5, 2024. [Online]. Available: https://docs.celeryq.dev/
5) Redis Ltd., "Redis Documentation," Version 7, 2024. [Online]. Available: https://redis.io/docs/
6) Ollama, "Ollama – Run Large Language Models Locally," 2024. [Online]. Available: https://ollama.ai/
7) Chroma Research, "ChromaDB – The Open-Source Embedding Database," 2024. [Online]. Available: https://www.trychroma.com/
8) LangChain, Inc., "LangChain Documentation," 2024. [Online].Available: https://python.langchain.com/docs/
9) The Pandas Development Team, "pandas: Powerful Data Analysis Tools for Python," Version
10) 2.x, 2024. [Online]. Available: https://pandas.pydata.org/docs/
11) Python Software Foundation, "difflib — Helpers for Computing Deltas," Python 3 Standard Library Documentation, 2024. [Online]. Available: https://docs.python.org/3/library/difflib.html
12) Ministry of Law and Justice, Government of India, "India Code — Digital Repository of All Central and State Acts," 2024. [Online]. Available: https://www.indiacode.nic.in/
13) Government of India, "The Indian Penal Code, 1860 (Act No. 45 of 1860)," as amended by the Bharatiya Nyaya Sanhita, 2023. India Code, 2024.
14) Government of India, "The Information Technology Act, 2000 (Act No. 21 of 2000)," as amended by the Information Technology (Amendment) Act, 2008. India Code, 2024.
15) Government of India, "The Motor Vehicles Act, 1988 (Act No. 59 of 1988)," as amended by the Motor Vehicles (Amendment) Act, 2019. India Code, 2024.
16) T. Brown et al., "Language Models are Few-Shot Learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 1877–1901, 2020.
17) J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, pp. 4171–4186, 2019.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details