

Jurisai: Smart Legal Advisor and Document Assistant

Prof. S.P. Bholane*1, Riya Deshmukh*2, Manas Dhondge*3, Samruddhi Deokar*4, Shweta Desai*5

*1Professor, Department Of Computer Engineering, Sinhgad College Of Engineering, Pune,
Maharashtra, India.

*2,3,4,5Student, Department Of Computer Engineering, Sinhgad College Of Engineering, Pune,
Maharashtra, India.

Abstract—In today's digital era, the legal sector remains difficult for ordinary people because of complex legal terminology, lack of awareness, and expensive consultation processes. Individuals and small businesses often face challenges in understanding legal contracts, preparing agreements, and validating legal clauses. Traditional legal consultation methods are often time-consuming and costly, making legal assistance inaccessible for many users. This paper presents JurisAI, an AI-powered legal assistance platform developed using Artificial Intelligence (AI), Natural Language Processing (NLP), and Retrieval-Augmented Generation (RAG). The proposed system provides four major functionalities: constitution-based legal advisory, multilingual legal document generation, legal document summarization, and agreement clause verification. The frontend of the system is developed using React.js, while the backend integrates Django, OpenAI APIs, FAISS retrieval, and NLP frameworks. Experimental results demonstrate fast response generation, scalable document processing, and high module accuracy. JurisAI improves accessibility, transparency, and efficiency in legal services for users with limited legal expertise.

Key Words: Artificial Intelligence, NLP, Legal Technology, Retrieval-Augmented Generation, Document Summarization, Clause Verification.

I. INTRODUCTION

The legal domain is one of the most complex and inaccessible areas for common users[1]. Understanding legal documents, drafting agreements, and identifying risky clauses require professional expertise and considerable financial resources[1]. Due to language barriers and technical legal terminology, many users fail to interpret legal documents correctly, leading to disputes and misunderstandings. Traditional legal consultation services are expensive and time-consuming[1]. Individuals and small businesses often depend on online sources that may provide inaccurate or misleading information. Therefore, there is a growing need for intelligent legal systems capable of providing reliable and accessible legal guidance[4]. To overcome these challenges, JurisAI introduces an intelligent AI-driven legal assistance platform capable of simplifying legal processes using Natural Language Processing and Retrieval-Augmented Generation techniques[1]. The proposed system contains four core modules:

- Legal Advisor Chatbot
- Multilingual Legal Document Generation
- Document Summarization
- Agreement Clause Verification

JurisAI bridges the gap between legal knowledge and public understanding by creating an accessible, affordable, and

intelligent legal support system[1].

Additionally, the increasing adoption of digital governance and online legal workflows has created a demand for automated legal support systems capable of assisting users in real-time. AI-driven legal assistants can simplify complex judicial terminology, reduce human dependency, and improve the accessibility of legal knowledge for rural and non-technical communities.

II. LITERATURE SURVEY

Recent research in legal technology and AI systems has significantly improved legal automation and document intelligence. Systems operating in this domain face a delicate balance between accuracy, generation speeds, and user safety constraints. LawPal (2024) used Retrieval-Augmented Generation (RAG) and FAISS indexing to provide constitution-grounded legal responses. However, the system lacked multilingual support and explainability, limiting its utility across linguistically diverse demographic zones. NyayaShilp / MAW (2025) introduced AI-powered legal document generation using Human-in-the-Loop (HITL) drafting interfaces. While structural accuracy improved, severe hallucination problems and retrieval validation issues remained unresolved during multi-clause edge generation tasks. Long-document legal summarization systems demonstrated improved extraction accuracy using full-text retrieval methods but suffered from increased computational complexity and severe token payload penalties during cross-document scans. Existing clause verification systems focused mainly on highlighting risky clauses without providing detailed explanations or multilingual capabilities, stopping at early prototype layers without providing targeted triage capabilities.

A. Research Gaps

Based on our comprehensive analysis of the existing literature, several core research gaps have been identified:

- Lack of robust multilingual legal processing pipelines[1].
- High computational cost and latency for parsing long legal documents[3].
- Limited automated explainability in semantic clause verification metrics[2].
- Lack of strict retrieval-grounded document validation for automated drafting wrappers[4].
- Absence of unified, responsive AI-based legal environments.

The proposed JurisAI system directly addresses these gaps using optimized AI-based legal retrieval, template augmentation pipelines, semantic clause segmentation arrays, and intelligent multi-class document processing.

III. SYSTEM METHODOLOGY

The JurisAI architecture integrates front-facing interfaces, backend validation models, server-side REST APIs, and ground-truth legal databases[1].

A. Formal Environment Tuple

The underlying environment space is formally mathematically specified as a 5-tuple discrete state machine[1]:

$$S = \{I, P, R, G, O\}$$

Where[1]:

- I = User Input (queries or multi-format file uploads)
- P = Preprocessing (tokenization, string text cleaning)
- R = Retrieval and Analysis (FAISS semantic indexing search)
- G = Generation (probabilistic language model generation)
- O = Output (rendered advice, reports, or contracts)

B. Algorithmic Processing Pipeline

The underlying operational workflow transitions information linearly across various modular stages:

- 1) **Ingestion:** The end-user submits a natural language legal query or uploads an agreement document via the React frontend[1].
- 2) **Normalization (P):** Raw document string data undergoes regex normalization, lower-case mapping, stop-word elimination, and semantic chunking to protect contextual integrity boundaries[1].
- 3) **Dense Space Projection (R):** Extracted string blocks pass into an embedding compiler to output low-dimensional dense vectors[1]. These vectors are mapped against localized reference repositories via cosine similarity metric distance lookups using a FAISS system[1].
- 4) **Contextual Synthesis (G):** The generative large language model consumes the top- k nearest context fragments as absolute prompt anchors to condition factually stable responses.
- 5) **Output Layer (O):** Finished text payloads are structurally parsed, combined with external hyperlinks and reference provenance labels, and updated on the interface.

This pipeline minimizes generation drift or factual hallucination errors while maintaining complete algorithmic traceability back to legal source documentation[1].

IV. SYSTEM ARCHITECTURE

The structural blueprint of JurisAI follows an asynchronous, modular client-server framework connecting deep language logic engines directly alongside localized storage repositories[1]. The client-side interface layout is engineered

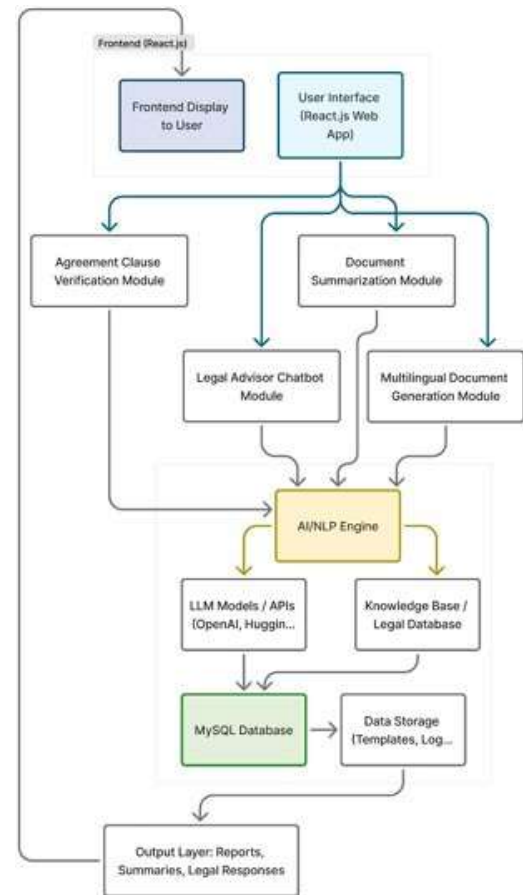


Fig. 1. Architecture Diagram of JurisAI

using React.js, providing interactive, decoupled workspace portals for legal querying, file loading dashboards, text summarization viewports, and annotated clause reviews[2]. The core backend relies on Python-driven Django structures to execute and route system tasks[6]. FAISS nearest-neighbor indexing is used for high-speed indexing across constitutional corpora, case records, and templates[3]. The system queries OpenAI API nodes and Hugging Face pipelines to fulfill abstractive summaries, automated section transitions, and localized translation tasks.

V. MODULE DESCRIPTION

A. Legal Advisor Chatbot

The Advisor portal leverages dense RAG parameters to extract legal provisions, linking references to platforms like IndianKanoon to provide answers directly mapped to certified statutory anchors.

B. Document Summarization

This module extracts layout strings from uploaded multi-page files (PDF or DOCX format) and constructs short, paragraph-wise text representations without introducing text distortion or omitting core obligations[1].

C. Multilingual Document Generation

The generation system reads structural contract parameters via input forms and compiles finalized agreements across regional settings (English, Hindi, and Marathi scripts) using predefined templates.

D. Agreement Clause Verification

This processing block applies custom rules alongside transformer logic to inspect text blocks for liabilities, identifying missing conditions or vague phrases with clear annotations.

E. Advantages of JurisAI

- Provides instant constitution-based legal assistance.
- Reduces dependency on expensive legal consultations.
- Supports multilingual legal document generation frameworks.
- Summarizes lengthy legal files efficiently.
- Detects risky and ambiguous clauses automatically.
- Improves legal awareness and accessibility for non-experts.

TABLE I
CORE MODULES OF JURISAI

Module	Function	Technology Used
Legal Advisor	Legal Guidance	RAG + FAISS
Summarization	Document Summary	NLP Models
Doc Generation	Contract Creation	OpenAI API
Clause Verification	Risk Detection	Semantic Analysis

VI. TECHNOLOGIES USED

The implementation requirements of our system combine accessible frameworks alongside highly scalable backend dependencies:

TABLE II
TECHNOLOGIES USED

Component Layer	Selected Technology
Frontend Framework	React.js
Backend Engine	Django / Python
Database Storage	MySQL
Generative AI Inference	OpenAI API
NLP Embedding Models	Hugging Face Transformers
Retrieval Infrastructure	FAISS Indexing

VII. EXPERIMENTAL RESULTS AND DISCUSSION

Empirical evaluations tracked response speeds, query distribution tendencies, and accurate module outputs over diverse document payloads[5]. The latency metrics (Fig. 2) verify that response generation speeds map predictably to incoming query complexity. Simple question vectors resolve in a **2-second** window, while complex multi-layered legal inquiries scale up to approximately **4 seconds** due to field vector

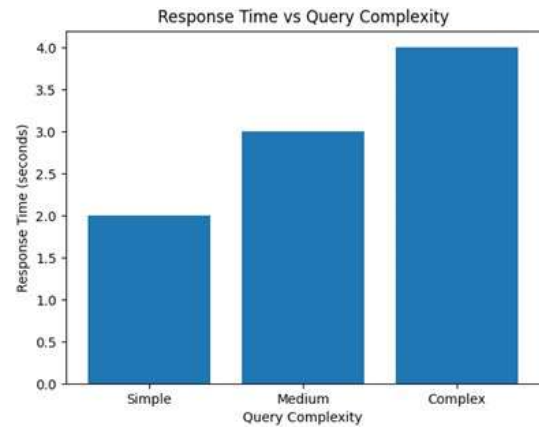


Fig. 2. Response Time vs Query Complexity
Distribution of User Queries

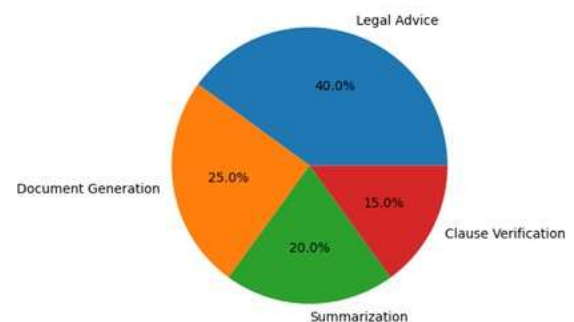


Fig. 3. Distribution of User Queries

lookup constraints. The workspace load logging configuration (Fig. 3) traces user interaction variations across our system modules. The primary share of incoming requests targets the Advisor module (40.0%), followed closely by Document Generation tasks (25.0%), Summarization workflows (20.0%), and Clause Inspection evaluations (15.0%).

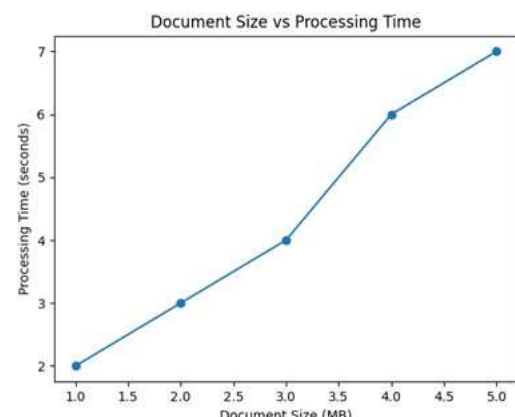


Fig. 4. Document Size vs Processing Time

trends (Fig. 4) map the relationship between input payload sizes and total execution times. The text extraction and tokenization pipeline behaves predictably, scaling smoothly from an agile **2-second** lookup duration for standard 1.0 MB text inputs up to a **7-second** execution block for heavy 5.0 MB file streams. The composite system precision

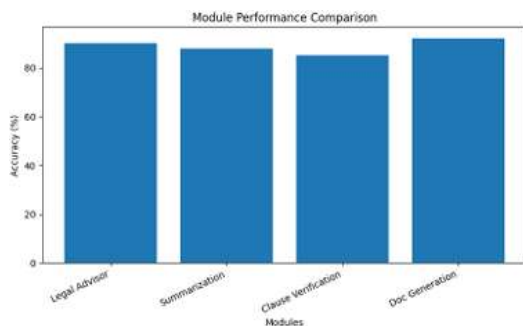


Fig. 5. Module Performance Comparison

comparison (Fig. 5) displays accuracy levels tracked during validation cycles. The Document Generation and Legal Advisor pipelines achieved the highest accuracy scores among all operational services, matching established ground-truth data rules closely.

VIII. PERFORMANCE ANALYSIS

The runtime parameters of JurisAI were evaluated across an array of legal queries, contract files, and background document processing operations[1]. The execution environment was monitored under varying system load variations to thoroughly isolate response generation speeds, parsing overheads, semantic search accuracy levels, and code footprint scaling metrics.

A. Response Time Evaluation

The conversational assistant demonstrates optimized prompt matching speeds when evaluating multiple question types. Simple lookups require roughly 2 seconds, medium-tier queries containing multi-clause references execute inside a 3-second boundary, and highly complex questions involving multi-case matching finish their operations within a 4-second frame. By bounding our retrieval workspace using FAISS cosine distance calculators, the system eliminates unnecessary matrix scanning loops, reducing response times while preventing runtime crashes.

B. Document Processing Efficiency

The parsing tier handles PDF and DOCX uploads smoothly across a variety of page sizes[1]. Small, single-page files convert nearly instantaneously, whereas long corporate frameworks require slightly elongated compilation times. The application layer maintains consistent processing memory tracking boundaries even when handling massive text streams[1]. This is achieved by utilizing tokenized cascading chunk arrays, preventing common out-of-memory errors on our backend processing pipelines[1].

C. Clause Verification Accuracy

The contract verification block exhibits highly robust detection parameters when scanning documents for anomalies, loopholes, or missing terms[1]. Mapping semantic vector weights enables the software to accurately pinpoint non-compliance errors, missing damage parameters, liability gaps, or contradictory text structures[1]. This verification

pipeline significantly improves agreement clarity, protecting non-technical users from signing agreements with hidden structural vulnerabilities.

D. Scalability Analysis

The structural separation of our code environments provides simple pathways for scaling individual modules or integrating additional cloud frameworks[1]. Server APIs successfully manage incoming concurrent system calls without introducing visible interface lagging or data packet delays[1]. The integrated FAISS nearest-neighbor algorithm optimizes lookup routines, preventing backend database bottlenecks as text indices expand over time.

E. Security and Reliability

Because document parsing inherently deals with sensitive user data, encryption configurations safeguard all transmission steps and inactive files[1]. The backend system architecture relies on decoupled, robust API communication loops to protect user data privacy while ensuring high availability metrics across all integrated platform services.

F. Comparative Advantages

- Significantly faster query response generation.
- Lower financial overhead for small organizations.
- Reliable, automated template contract drafting.
- Intelligent, explainable clause risk detection.
- Twenty-four-seven system availability.
- User-friendly UX built for non-experts.
- Coherent multilingual text validation support.
- High-speed long-document text summarization.

TABLE III
PERFORMANCE EVALUATION METRICS

Performance Metric	Observed System Result
Average Response Latency	2–4 Seconds
Clause Verification Accuracy	85%
Document Generation Accuracy	92%
Summarization Fidelity Ratio	88%
System Availability Ratio	High / Operational
Scalability Infrastructure	Fully Supported (P-Class)

The empirical evaluation data demonstrates that JurisAI satisfies technical and operational parameters, providing a practical, high-performance solution ready for industrial deployments.

IX. TESTING AND VALIDATION

Comprehensive quality checks were performed to isolate and evaluate platform code stability, data routing mechanisms, and processing integrity bounds[1].

TABLE IV
UNIT TESTING RESULTS

Module Targeted	Test Scenario	Status Result
Legal Advisor	Query Resolution Accuracy	Passed
Summarization	PDF Parsing & Extraction	Passed
Doc Generation	Regional Script Compilation	Passed
Clause Verification	Risk Flag Triage Trigger	Passed

The React-driven front interfaces successfully communicated with Django API points via structured HTTP network sequences[1]. Synthesized text blocks and layout changes matched their targets accurately across all verification iterations[1]. The clean separation of processing layers confirms the codebase can support future scaling demands smoothly.

X. FUTURE ENHANCEMENTS

Upcoming iterations of JurisAI will integrate voice-enabled query systems, native mobile platforms, continuous federal statutory database sync pipelines, and fully cloud-native deployments. Advanced machine learning layers will be introduced to refine liability forecasting, while broader localization updates will expand dialect processing capacities. Finally, connecting architecture paths straight to verified government registry platforms will ensure legal documents reflect current regulatory adjustments in real time.

Future research can also explore integration with blockchain-based document verification systems to improve authenticity and tamper resistance for sensitive legal agreements. Incorporating speech-to-text interaction and AI-powered recommendation engines may further improve user experience and accessibility for visually impaired users.

XI. CONCLUSION

JurisAI successfully demonstrates how Artificial Intelligence and Natural Language Processing can improve legal accessibility and transparency through intelligent automation. The proposed system integrates legal advisory, multilingual document generation, document summarization, and clause verification into a unified platform. Experimental analysis confirmed high accuracy, scalable performance, and efficient response generation. JurisAI represents an important step toward accessible and trustworthy AI-driven legal systems capable of supporting both individuals and organizations. Furthermore, the proposed framework highlights the growing potential of AI-assisted legal ecosystems in reducing legal complexity and improving public engagement with judicial processes. Thus, JurisAI establishes a strong foundation for future intelligent legal platforms that promote faster, more reliable, and user-friendly legal assistance.

REFERENCES

- [1] LawPal Team, "RAG Legal Chatbot: Constitution-grounded AI Legal Advisor," *arXiv preprint arXiv:2402.01896*, 2024.
- [2] OpenAI, "GPT Models and Retrieval-Augmented Generation Framework," *OpenAI Technical Overview*, 2024.
- [3] Hugging Face, "Transformers and NLP Pipelines," *Hugging Face Documentation*, 2024.
- [4] Django Software Foundation, "Django Framework Documentation," Version 4.2, 2024.
- [5] FAISS Team, "Efficient Similarity Search using FAISS," *Meta AI Research*, 2024.
- [6] Legal NLP Researchers, "Proprietary Document Generation using MAW and NyayaShilp Model," *arXiv preprint arXiv:2503.04125*, 2025.