International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

# Legal Connect: AI-Based Legal Guidance and Query Assistance Platform

**Prof. Nikita Nandapure, Tanisha Dalal[1], Shreya Sadawarti[2], Manasvi Aglawe[3], Shreyash Bawankar[4], Diya Mohapekar[5]**

Department of Computer Science and Engineering, Priyadarshani Bhagwati College of Engineering, Nagpur, Maharashtra, India

**Corresponding contact:** | Tanisha Dalal: tanishadalal208@gmail.com

Prof. N. Nandapure: nnandapure@gmail.com | S. Sadawarti: sadawartishreya@gmail.com | M. Aglawe: amanasvi05@gmail.com | S.Bawankar:shreyashbawankar64@gmail.com| D. Mohapekar: mohapekardiya@gmail.com

## Abstract

Legal information is dispersed across statutes, rules, court decisions and official notices, and it is usually written in language that ordinary citizens find difficult to follow. People who need a quick and dependable orientation before approaching an advocate therefore have few convenient options. This paper presents the design of Legal Connect, a web-based legal assistance system in which a plain-language question is validated, interpreted, matched with relevant legal information, and answered by an AI module in structured and accessible wording. The proposed architecture separates a static presentation tier hosted on Netlify from a versioned REST service hosted on Render, which exposes interactive OpenAPI documentation and a health endpoint, while the source code is maintained in a public GitHub repository. The design adds authenticated access, a safety stage that attaches disclaimers and discourages reliance on unverified output, and an evaluation plan covering correctness, hallucination, clarity, latency and privacy. Legal Connect is an information tool and is not a substitute for a licensed advocate. The supplied project material does not contain measured results, benchmark datasets or expert evaluation, so no accuracy or performance figures are reported and no claim of completed testing is made. The paper therefore serves as a design-and-methodology draft that should be updated with verified implementation details and test results before journal submission.

**Keywords—** legal technology, artificial intelligence, natural language processing, large language models, access to justice, REST API, cloud deployment

## I. INTRODUCTION

Legal questions arise in everyday life, whether in a rental agreement, a consumer dispute, a workplace issue or a family matter. The Constitution of India directs the State to secure equal justice and to provide free legal aid [1], and a statutory framework for legal services exists [2]. In practice, however, many people do not know where to start. The relevant provisions are scattered, phrased in formal vocabulary and spread across several sources, so a person may need to read many documents before understanding even a basic position.

Digital portals can narrow this gap, but a collection of documents still leaves the interpretation to the reader. Artificial Intelligence (AI), Natural Language Processing (NLP) and Large Language Models (LLMs) offer the possibility of reading a question written in ordinary words and answering it in an organised form. At the same time, accuracy, fabricated content and the sensitivity of personal circumstances remain serious concerns that require human oversight and careful design.

The project, titled Legal Connect, proposes a web platform that combines a simple interface, a documented service layer and an AI-based query module. The interface is intended to help a user orient themselves and decide whether to consult a qualified advocate. The design aims at clarity, controlled access and responsible presentation of AI output through a simple user interface.

The main contribution of this paper is a structured design of the proposed project, including its scope and assumptions, functional modules, query workflow, threat model and a reproducible evaluation plan. Since the supplied project material does not report measured outcomes, this paper distinguishes design expectations from demonstrated results.

### *A. Problem Statement*

Legal rules are accessible in principle but difficult to use in practice. A citizen who searches for guidance may meet outdated pages, technical language or conflicting summaries, and professional consultation may be unaffordable at the first stage. An AI system could help, but a model that answers without safeguards can state incorrect law with great confidence. A responsible workflow is therefore required in which a query is validated, interpreted, grounded in identifiable legal information, answered in plain language, and accompanied by a clear reminder that the output is general information.

### *B. Aim and Objectives*

The aim is to design a web-based legal assistance system that gives citizens understandable preliminary guidance while reducing the risk of misleading output. The objectives are:

- Accept legal queries in simple natural language through a web interface.
- Identify the intent and legal area of each query.
- Retrieve relevant legal information to support the answer.
- Generate an easy-to-follow AI-based explanation.
- Attach disclaimers and encourage verification by a licensed professional.
- Expose a versioned, documented and monitorable REST service.

- Deploy front end and back end independently on cloud platforms.
- Define tests that evaluate correctness, safety and usability without inventing performance outcomes.

### *C. Scope and Assumptions*

The proposed system covers authenticated query submission, query understanding, retrieval of legal information, AI-based response generation, and a safety stage before display. The language model and the knowledge sources are treated as components that may be wrong or out of date; they are not assumed to be fully reliable. The design assumes that the hosting platforms and cryptographic libraries are not already compromised. It does not claim to give formal legal advice, to represent a user before any authority, to guarantee that every answer is correct, or to cover every jurisdiction and every area of law.

Legal assistance can be considered as a layered problem rather than a single question-answering task. The presentation layer must be understandable, the application layer must enforce identity and protect data, and the intelligence layer must produce grounded content. A failure in any layer weakens the whole system. For example, a fluent answer is of little value if it rests on a superseded provision, and strong security does not help if the displayed guidance is misleading. This layered perspective informs the separation of modules in the proposed architecture.

An important distinction in reviewing such systems is the difference between legal information and legal advice. Information explains what a rule generally says; advice applies it to the particular facts of a person with professional responsibility. Legal Connect is designed to provide the former. Its output should therefore be phrased as general guidance, and any serious matter should be referred to a licensed advocate.

## II. LITERATURE REVIEW

Research on legal AI has grown quickly. Zhong et al. surveyed how NLP is applied within the legal system, covering tasks such as case retrieval, judgment prediction and legal question answering [3]. Their review maps the problems that a legal assistant may address and supports the choice of query understanding and retrieval as separate stages.

Chalkidis et al. showed that language models pretrained on legal corpora handle legal text more effectively than general-purpose models, because legal writing has its own vocabulary and structure [4]. This supports the view that the AI module should be treated as replaceable and, where possible, adapted to the legal domain.

Katz et al. reported strong results for a modern LLM on a standard professional legal examination [5]. The finding indicates that automated legal reasoning is technically feasible, although examination performance is not the same as dependable advice for a real person with a real problem.

Dahl et al. profiled legal hallucinations and found that general-purpose models frequently produce incorrect legal statements [6]. A later assessment by Magesh et al. observed that even tools built on retrieval can still make mistakes [7]. These studies explain why the present project includes a safety stage, disclaimers and a recommendation of professional verification instead of presenting output as authoritative.

Lewis et al. proposed combining a generative model with an external knowledge source so that answers can be linked to retrieved evidence [8]. Such grounding is attractive in law, where traceability to a source is valued, and it indicates a path for later versions of the platform.

These studies indicate that a legal assistance system should address several related but distinct concerns: understanding the query, grounding the answer, controlling hallucination, protecting user data, and presenting information responsibly. The proposed project brings the first, second and fifth concerns into a single workflow and plans for the others. Its current limitation is that the supplied material does not provide benchmark datasets, expert evaluation or measurements. The evaluation plan in this paper is intended to address these gaps during implementation.

Unlike the approaches discussed in the literature, which concentrate on models, benchmarks or professional research tools, the proposed system focuses on combining a citizen-facing web interface, a versioned and self-documenting REST service, cloud-based deployment and an AI query module within a single low-cost platform. It also defines an evaluation plan to assess correctness, safety, usability and availability after implementation.

## III. METHODOLOGY

### *A. System Architecture*

The proposed system contains a web interface, an authentication component, a query-processing module, a legal-information retrieval module, an AI response module, a safety module and a service interface, as shown in Fig. 1. The static interface is delivered from Netlify and communicates over HTTPS with the application service hosted on Render. Routes of the service sit under a version prefix, so later changes can be introduced without breaking existing clients. The service publishes interactive documentation and a health endpoint, and the complete source is kept in a public GitHub repository.

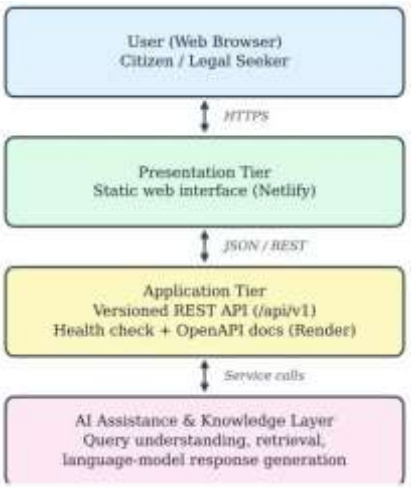



Fig. 1. Proposed logical architecture of the Legal Connect system.

### *B. Functional Modules*

1) User authentication: The application validates credentials before sensitive actions. Passwords should be stored as salted hashes produced by a dedicated password-hashing algorithm, sessions should expire appropriately, and repeated failed logins should be rate-limited.

2) Query input and validation: The user describes a situation in natural language. The service checks length and format, removes unsafe content and rejects malformed requests before any processing takes place.

3) Query understanding: NLP techniques identify the intent, the legal area involved and key details such as parties, dates and the nature of the dispute, so that retrieval can be targeted.

4) Legal information retrieval: Relevant material is gathered from the configured knowledge sources. Each source should be identifiable so that an answer can be traced back and checked. The currency of every source must be recorded because laws change.

5) AI response generation: A language-model module converts the understood query and the retrieved material into an organised explanation in accessible language. Because it sits behind the service boundary, the model can be replaced without changing the interface [9].

6) Safety, disclaimer and review: Before display, each answer is accompanied by a statement that it is general information and not legal advice. Out-of-scope or high-risk queries should be redirected, and flagged responses can be reviewed by an authorised person.

7) Service interface and monitoring: The service follows the representational state transfer style [10] and is described through an OpenAPI contract [11], a capability commonly provided by Python frameworks such as FastAPI [12]. Documentation and health endpoints are summarised in Table I.

**TABLE I. SERVICE INTERFACE ENTRY POINTS**

| Endpoint | Purpose | Used by |
|---|---|---|
| /api/v1/health | Liveness and status check | Operators |
| /docs | Interactive API documentation | Developers |
| /api/v1/* | Versioned functional routes | Front end |

### *C. Query Workflow*

Request workflow: (i) the user opens the interface and, where required, authenticates; (ii) the user submits a question; (iii) the service validates and sanitises the input; (iv) the query is interpreted to find its intent and legal area; (v) relevant legal information is retrieved; (vi) the AI module drafts an explanation grounded in that information; (vii) the safety stage adds disclaimers and checks scope; and (viii) the answer is returned over HTTPS and displayed in plain language.

Failure workflow: if validation, retrieval or generation fails, the system should fail safely by returning a clear message instead of an unsupported answer. Error messages must not expose internal paths, credentials or other users' data. The workflow is illustrated in Fig. 2.

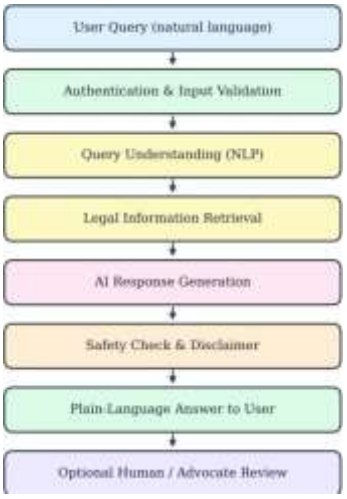



Fig. 2. Proposed query-processing workflow.

A language model must not be treated as an authority. Its role in this design is to explain retrieved material in plain wording, and the implementation must use maintained libraries and avoid unverified custom components. Prompt design, source selection and error handling are quality-critical parts of the implementation.

### *D. Security Requirements and Threat Model*

The main goals are privacy of user queries, integrity of the knowledge sources and controlled access. Privacy means that questions, which may reveal personal circumstances, are not exposed to other users or written carelessly to logs. Integrity means that the legal sources and the system configuration cannot be altered without authorisation. Access control means that protected operations are limited to authenticated and authorised users.

The threat model includes an attacker who attempts to read another user's data, submits crafted input to manipulate the AI module, abuses an exposed route, or obtains credentials from the repository. The design also considers accidental misuse by well-meaning users who treat output as professional advice. It does not assume protection against a fully compromised user device or a malicious platform administrator, and these limits should be stated in project documentation.

Important controls include HTTPS for data in transit, restriction of cross-origin access to the trusted front end, validation of all input, storage of secrets in environment variables outside the repository, least-privilege hosting permissions, logging that avoids sensitive content, and routine dependency updates. Authentication and safety solve different problems: a logged-in user may still receive an incorrect answer, and a correct answer may still be unsuitable without professional review.

### *E. Proposed Evaluation Plan*

Because no measurement logs or benchmark results were included in the supplied project material, evaluation is specified as a plan rather than reported as completed work. Tests should be performed only after the system is implemented, using non-sensitive and synthetic queries.

- Functional correctness: submission, processing and display flows behave as intended.

- Legal accuracy: answers are compared with reviewed legal sources by qualified reviewers.
- Hallucination check: unsupported or invented statements are counted per answer set.
- Safety behaviour: disclaimers appear and out-of-scope queries are redirected.
- Access-control test: protected routes reject unauthenticated requests.
- Clarity and usability: users complete tasks and rate how well they understand answers.
- Performance measurement: record response time under cold and warm conditions and service uptime using the health endpoint.

For performance measurements, the hosting tier, region, network conditions, software versions and query set should be documented. Each measurement should be repeated and summarised using a mean or median with the number of runs stated. No numerical result should be included unless it is produced by the implemented system and can be reproduced.

At least three classes of queries are recommended: simple informational questions, multi-issue scenarios, and out-of-scope or adversarial inputs. The test report should include negative cases and should be presented in a compact table with columns for test identifier, procedure, expected outcome, observed outcome and pass/fail status.

## IV. RESULTS

The supplied project material defines the intended components and workflow, together with the web resources that host the prototype, but it does not provide benchmark datasets, test logs, expert reviews or measured performance values. Consequently, this draft does not claim that the AI module, the retrieval stage or the safety stage has been experimentally validated. The current deliverable is a system design and evaluation specification.

At the design level, the workflow separates query understanding from retrieval and generation, and it includes a defined safety stage. These are properties of the proposed architecture, not proof that a completed application meets them. Implementation evidence should be added here after functional, safety and usability tests are performed. A final version should include a table of test cases with expected and observed outcomes and a performance table containing actual measurements.

**TABLE II. PLANNED VALIDATION TESTS**

| Test | Expected outcome | Status |
|---|---|---|
| Valid query and answer | Relevant, readable guidance is returned | Not tested |
| Out-of-scope query | System declines or redirects safely | Not tested |
| Unsupported claim check | Answer matches reviewed legal source | Not tested |
| Unauthorized access | Protected routes deny the request | Not tested |
| Service health probe | Health endpoint reports status | Not tested |
| Latency (cold / warm) | Response times are recorded | Not tested |

## V. DISCUSSION

The proposed architecture is a practical baseline for a student project because it separates the tasks of presentation, service logic, retrieval and AI generation. A static front end is inexpensive to host and can be published automatically from the repository, while the separate service can be upgraded, monitored and documented independently. The AI layer remains independent of the interface, provided the service returns structured responses.

Several design choices require care before the system can be described as dependable. First, a language model can produce fluent but unsupported statements [13], so answers should be tied to retrieved sources and displayed with a clear disclaimer. Second, law differs between jurisdictions and changes over time; each knowledge source needs a recorded date and a maintenance routine. Third, user questions can be sensitive, so storage should be minimised and protected. Fourth, authentication must be paired with authorisation and secure session handling.

Operational limits are also important. Free-tier hosting may suspend an idle service, so the first request after a quiet period can be slower, and a periodic probe of the health endpoint can both measure availability and keep the service responsive. Availability and recovery deserve separate treatment from correctness: a platform can be secure and still be unreachable if a hosting account is suspended or a configuration is lost. Backup of source code and configuration should therefore be documented.

Scalability depends on how the service handles concurrent users and long inputs. A production-oriented design may add request limits, caching of repeated queries and asynchronous processing of slow generation steps. The current project description does not state whether such measures have been implemented, so they remain design considerations.

The project lists lawyer assistance as an intended benefit, but a mechanism for connecting users to verified advocates is not specified. To support it responsibly, the system needs a verified directory, consent for sharing the user's question, and a clear separation between the AI answer and professional advice. Multilingual support, role-based access control, audit logging and voice input are reasonable extensions, but they should be reported as future work unless they are implemented and tested.

The main limitation of this draft is the absence of implementation evidence. The proposed design can guide development, but journal-ready claims about accuracy, usability or efficiency require repeatable tests and an explicit comparison with relevant baselines. A careful final paper should report both positive and negative findings, describe the test environment, and avoid presenting expected behaviour as an observed result.

## VI. CONCLUSION

This paper presents a design for Legal Connect, an AI-based platform for accessible legal guidance. The proposed system validates a plain-language query, interprets it, retrieves relevant legal information, generates an accessible explanation and applies a safety stage before display. The architecture separates a static front end from

a versioned, self-documenting REST service and relies on Git-based deployment on cloud platforms.

The design can provide a foundation for a useful legal-information application, but its real value depends on the quality of its sources, the correctness of generated answers, the protection of user data and the clarity of its limits. AI output must be paired with grounding and professional review to be meaningful. Future work includes implementation, expert-reviewed accuracy testing, measured performance evaluation, retrieval over verified statutes and judgments, multilingual support, advocate referral and stronger privacy controls. Actual results must be added before the manuscript is represented as an experimentally evaluated system.

## ACKNOWLEDGMENT

The authors acknowledge the guidance of Prof. Nikita Nandapure and the Department of Computer Science and Engineering, Priyadarshani Bhagwati College of Engineering, Nagpur, for support of the project.

## REFERENCES

[1] Constitution of India, 1950, art. 39A (inserted by the Constitution (Forty-second Amendment) Act, 1976).

[2] Legal Services Authorities Act, 1987, Act No. 39 of 1987, India.

[3] H. Zhong, C. Xiao, C. Tu, T. Zhang, Z. Liu, and M. Sun, "How does NLP benefit legal system: A summary of legal artificial intelligence," in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics (ACL), 2020, pp. 5218–5230.

[4] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "LEGAL-BERT: The muppets straight out of law school," in Findings Assoc. Comput. Linguistics: EMNLP, 2020, pp. 2898–2904.

[5] D. M. Katz, M. J. Bommarito, S. Gao, and P. Arredondo, "GPT-4 passes the bar exam," Phil. Trans. R. Soc. A, vol. 382, no. 2270, 2024, Art. no. 20230254.

[6] M. Dahl, V. Magesh, M. Suzgun, and D. E. Ho, "Large legal fictions: Profiling legal hallucinations in large language models," J. Legal Analysis, vol. 16, no. 1, pp. 64–93, 2024.

[7] V. Magesh, F. Surani, M. Dahl, M. Suzgun, C. D. Manning, and D. E. Ho, "Hallucination-free? Assessing the reliability of leading AI legal research tools," arXiv:2405.20362, 2024.

[8] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, 2020, pp. 9459–9474.

[9] A. Vaswani et al., "Attention is all you need," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 30, 2017, pp. 5998–6008.

[10] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2000.

[11] OpenAPI Initiative, "OpenAPI Specification," Linux Foundation. [Online]. Available: https://spec.openapis.org/oas/latest.html

[12] S. Ramírez, "FastAPI documentation." [Online]. Available: https://fastapi.tiangolo.com

[13] Z. Ji et al., "Survey of hallucination in natural language generation," ACM Comput. Surv., vol. 55, no. 12, Art. no. 248, 2023.