IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2026

DOI: 10.17148/IARJSET.2026.131060

MargVidhi: An AI-Assisted Prototype for Legal Education and Case Analysis in Indian Motor Vehicle Law

Sheeba S[1], Darshan H[2], Rakshan Appachu KB[3], Arnav Adithya S[4], Tarun S[5]

Assistant Professor, Dept. of CSE, Don Bosco Institute of Technology Bengaluru, India[1]

Computer Science and Engineering, Don Bosco Institute of Technology Bengaluru - 560074, India[2-5]

Abstract: People encountering a road accident describe what happened, not which section of the Motor Vehicles Act, 1988 might apply. MargVidhi is a browser-based prototype built around a short conversation to help with that lookup. The local corpus for this build holds 60 curated entries. Google Gemini can be used to talk through the user's scenario, but the prototype does not depend on it: a deterministic fallback covers a small set of common topics. In practice the controller asks a few clarifying questions, drafts a case summary for the user's approval, and only then starts retrieval. Fuse.js searches the local corpus across keywords, titles, and explanations, and the matches are shown as cards with the statutory wording kept separate from the plain-language explanation, penalty, suggested action, and forum. Users type their description at present; a reusable speech-recognition component and language settings for English, Hindi, Tamil, Telugu, and Bengali sit in the codebase but voice input is not wired to the current analysis screen. This paper describes how we built the prototype, what we checked functionally, and where it falls short. With no labelled benchmark or user study, we make no claim about legal accuracy.

Index Terms: legal technology, legal education, conversational AI, retrieval-grounded generation, Motor Vehicles Act, fuzzy search, multilingual interface, explainable output

I. INTRODUCTION

A road accident does not begin with a legal citation. Someone may mention "no helmet," "drunk driving," or "hit and run"; meanwhile, the Motor Vehicles Act, 1988, is organised into sections, duties, conditions, and penalties [1]. Connecting the two means turning a casual description into a legal one. Simply putting the Act online does not fix that for someone who has never worked with its internal organisation.

Language models can help with the conversational framing of that first query, but fluency does not equal reliability. Work on legal-domain models confirms that legal language has features that matter for downstream tasks [2]. Retrieval-augmented generation proposes a practical division of labour between models and repositories: a model may frame a query, while a separate knowledge source grounds the response [3]. MargVidhi follows that division at a much smaller scale. When Gemini is available we use it to carry the conversation and draft section summaries; the cards themselves, with citations, are always populated from a local structured repository.

This paper describes the implementation in the legal-ai-main project. MargVidhi is a learning assistant. It cannot act as a judicial officer and it does not replace an advocate. The interaction starts with a casual description, asks a few clarifying questions, and prepares a summary for the user's approval. Retrieval starts only after the user confirms that the summary reflects their situation. Our contributions are a staged interview for fact extraction, an offline fallback alongside the optional cloud model, deterministic fuzzy retrieval over a structured Act corpus, and a modular card layout showing statutory text, explanation, penalty, action, and forum separately. We limit validation claims to observable prototype behaviour, since the repository has no benchmark or user study to judge legal accuracy.

II. PROBLEM DEFINITION AND OBJECTIVES

A. Problem Description

The main engineering problem is reconciling a user's casual description with the indexed form of legal sections. The corpus is organised by act, chapter, section, keywords, and procedure. Search then becomes fragile: spelling slips, missing section numbers, irrelevant procedures, and other everyday usability issues get in the way. There is a second risk. A generated answer can sound convincing while leaving out a critical qualification, quoting an outdated penalty,

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2026

DOI: 10.17148/IARJSET.2026.131060

or naming the wrong section. Finally, the prototype has to keep working when no API key is set, when the provider call fails, or when cloud access is simply turned off.

B. Objectives

Our design goals are deliberately pragmatic: take casual descriptions, collect the facts that matter, and ground everything shown in a local legal reference. The interface should show its confidence score without pretending it is the truth, explain each result in plain language, and keep the Indian-language interface and speech settings usable. It must also remind the user that this is for education. That is what makes MargVidhi a learning aid rather than an autonomous legal assistant.

III. LITERATURE SURVEY

A. Legal Language Considerations

Legal text is not generic prose. It carries unusual terms, long explanations, cross-references, and meanings that shift by jurisdiction. Prior work on legal encodings, including LEGAL-BERT's adaptation of BERT, explores exactly this gap [2]. MargVidhi does not train its own domain model. Instead we pair a general conversational model with a curated structured repository, so the explanations are readable but the deterministic retrieval stays separate. That separation keeps model wording apart from the repository texts used in the cards.

B. Retrieval and Grounding

Lewis et al. describe retrieval-augmented generation as parametric generation combined with an explicit non-parametric memory [3]. We implement the same split in a lightweight way: the optional model handles stage detection and summary drafting, while the cards come from deterministic local search. Relevance ranking cannot stand in for a full legal review, but it does mean every record shown can be inspected by hand and updated when needed.

C. Conversational Safety and Accessibility

Public-facing models have to handle conversational safety: incomplete context, bias, and answers that sound more confident than the evidence allows [4]. MargVidhi answers with clarification questions, asks the user to confirm the generated summary, shows an educational disclaimer, and keeps model-generated text visually apart from repository records. For input, the Web Speech API defines browser speech-recognition and synthesis interfaces, though what actually works depends on browser and OS [5]. We ship a reusable voice-input component, but it is not wired to the current analysis view, so typed input remains the dependable fallback.

IV. SYSTEM ARCHITECTURE AND DESIGN

MargVidhi is a client-side React application with two routed views: a landing page and a case-analysis workspace. Fig. 1 shows the processing pipeline for a sample incident description. The interface takes typed input and includes a language switcher. A reusable voice-recognition component sits in the codebase but is not mounted in the current view. The controller either asks clarifying questions, shows a summary, or answers a general chat query. Once the user confirms the summary, it turns that summary into search terms, runs a local repository search, and shows the matches as expandable cards.

A. Input and Multilingual Interface

The landing page, also known as the e-Vakil interface in code, includes the language switcher and the route into analysis. On large screens the analysis view keeps the interview sidebar next to the legal-output panel; on small screens the user flips between them with tabs. LanguageContext provides interface strings and speech locales for English, Hindi, Tamil, Telugu, and Bengali. The reusable VoiceInput component looks for SpeechRecognition or webkitSpeechRecognition, sets the right BCP-47 locale, and returns transcripts through a callback. It is imported but not mounted by the current CaseAnalysis JSX, so the live view is text-based.

B. Interview Controller and Summary

The case-analysis controller moves through four states: INTAKE, CLARIFYING, SYNTHESIZED, and FINAL. Every incoming message is appended to the history before analyzeLegalCase handles it. When Gemini is enabled, the prompt asks for a JSON object with type, content, and suggestedKeywords. A QUESTION response keeps the interview open, SUMMARY produces a summary the user can review, and CHAT answers general queries. That confirmation step matters: the user can catch a misleading summary before any search runs.

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2026

DOI: 10.17148/IARJSET.2026.131060

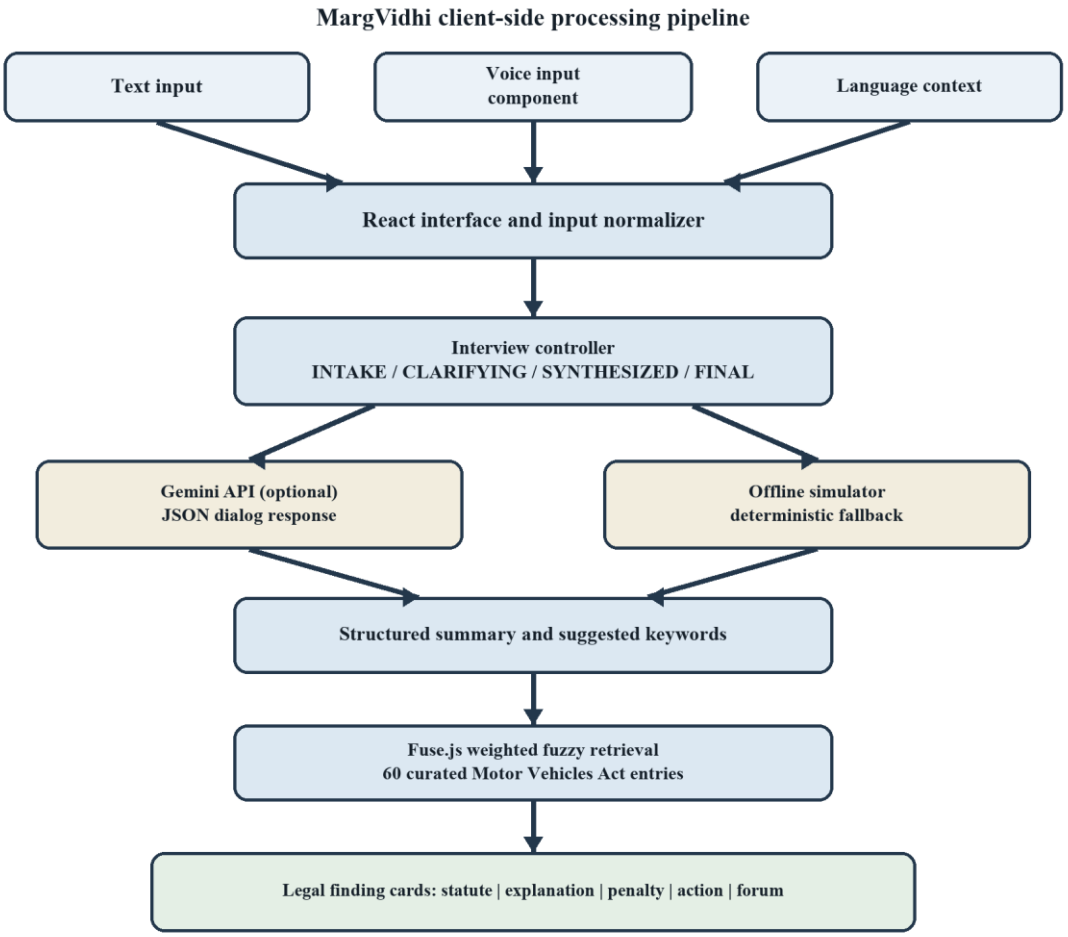


Fig. 1. Processing pipeline used by the MargVidhi prototype.

C. Optional Model and Offline Fallback

Gemini is optional here. It helps decide the conversation stage and draft the summary. The app reads VITE_GEMINI_API_KEY from the Vite environment and uses the Google Generative AI client when a key exists [6]. If the key is missing or the request fails, the app drops into simulation mode. The local rules recognise greetings, capabilities, penalties, procedures, accidents, licences, helmets, and drunk driving, and answer those directly. That keeps the prototype easy to demo, though it is clearly not a general offline legal assistant.

TABLE I
FUNCTIONAL MODULES IN THE MARGVIDHI PROTOTYPE

Module	Responsibility	Project evidence
Interface	Landing page, routed analysis workspace, and mobile tabs	LandingPage.tsx, CaseAnalysis.tsx
Input	Typed incident descriptions, reusable speech input, and five language settings	VoiceInput.tsx, LanguageContext.tsx
Dialogue	Clarification, summary, chat, and final retrieval stages	analyzeLegalCase, FlowStage
Retrieval	Weighted fuzzy search, thresholding, ranking, and top-six selection	Fuse.js, findRelevantSections
Explanation	Statutory text, plain-language meaning, punishment, action, and forum	LegalResponseCard.tsx

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2026

DOI: 10.17148/IARJSET.2026.131060

D. Search and Relevance Ranking

The controller merges the summary's suggested keywords with extra terms pulled from the summary text after the user confirms it. Fuse.js then runs a fuzzy search in the browser, giving keywords more weight than titles, numbers, and simplified explanations [7]. The code tries each keyword plus the combined string, keeps the best (lowest) Fuse score per section, and maps it to a bounded confidence value: confidence = round((1 −score) × 100). Results are sorted by confidence and the top six become cards. This is relevance ranking only: a higher number means a better text match, not that the section definitely applies.

E. Legal Text Explanations

Each LegalSection record is shown as a LegalResponseCard with the section name, relevance score, and a short excerpt of the statutory text in the header. The body holds the explanation, penalties, actions, and forums. Laying them out separately helps the reader tell retrieved text apart from generated explanation and guidance.

V. IMPLEMENTATION

A. Technology Stack and Modules

The prototype is written in React 18 with TypeScript and bundled with Vite. React Router handles the landing and analysis routes, Tailwind CSS handles responsive styling, Lucide supplies the icons, Fuse.js runs the fuzzy search, and Google's Generative AI client connects to Gemini when needed. The build is deliberately client-side: there is no server, database, or authentication layer in the repository.

B. Structured Legal Corpus

The current acts.json file holds 60 Motor Vehicles Act, 1988 records. Each record stores act and chapter, domain, section number and title, official text, simplified explanation, punishment, offence type, cognizable and bailable flags, trial forum, procedural guidance, related sections, and keywords. That layout maps directly to the cards: every field ends up somewhere on screen. At the same time we do not treat the file as ground truth. The repository has no line-by-line comparison against an authoritative source, so completeness and currency still have to be checked manually.

TABLE II
REPRESENTATIVE VALIDATION SCENARIOS FOR THE IMPLEMENTED PROTOTYPE

Scenario	Prototype behavior	Scope of check
First incident description	Asks whether the incident occurred on a public road and whether witnesses were present	The stage transition is implemented; exact wording varies by model
Drunk-driving query	Local responses and corpus keywords can surface Section 185	The curated entry exists, but there is no accuracy benchmark
Missing licence query	Fuzzy search links licence-related terms to Sections 3 and 181	Matching may improve recall; the confidence value is heuristic
Accident or hit and run	Summary terms can return accident-duty or compensation entries	Coverage depends on the 60-record corpus
No Gemini key	Deterministic simulation remains available for supported topics	Offline coverage is intentionally limited
Voice component	Unsupported browsers hide the control; typed input remains available	The component exists but is not yet mounted in CaseAnalysis

C. End-to-End Processing

Each turn appends the new message to the conversation history. The model or the local rules then decide whether to ask for more detail, prepare a summary, or give a chat reply. Once the user confirms the summary, the app searches the local corpus with the suggested keywords and summary terms. The ranked hits are mapped to LegalSection objects and shown with the short disclaimer from LanguageContext. Because the search is deterministic and local for a fixed query, it can be tested without touching the cloud model.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2026

DOI: 10.17148/IARJSET.2026.131060

D. Deployment and Configuration

For development, Vite reads the optional provider key from VITE_GEMINI_API_KEY at build time. That is handy for a prototype, but shipping a secret in browser code would not suit a public release. A real deployment would need a server-side proxy, rate limiting, and privacy-preserving redaction. Conversation history is also transient: it lives only for the current session, with no server-side persistence or database behind it.

VI. FUNCTIONAL VALIDATION

Validation at this stage is limited to demonstrating the prototype. There is no labelled legal-query benchmark, retrieval corpus, or user study behind it, so we cannot support claims about legal accuracy, precision, recall, or latency. Table II lists the scenarios we walked through to exercise the controller, local rules, corpus, and output components. They show what the prototype does; they do not prove it handles every case correctly.

Automated coverage is intentionally thin: the repository holds only a placeholder assertion. Changing the legal-search routine or the controller stages therefore breaks no unit or integration tests. The build works as a proof of concept, but proper validation would need unit tests, integration tests, and a benchmark query-to-section evaluation.

VII. LIMITATIONS, SAFETY, AND ETHICAL CONSIDERATIONS

A. Coverage and Legal Currency

Our local reference covers only the Motor Vehicles Act. The landing page also mentions the Information Technology Act and Indian Penal Code, but neither domain is present in acts.json. That mismatch should be fixed by removing those labels or by marking them clearly as planned extensions. The code also ignores amendments, state-specific sections, court interpretations, and current penalty schedules. Every entry would need version control and regular upkeep before this could be treated as a public service.

B. Model, Retrieval, and Language Limitations

The Gemini path pulls a JSON object out of generated text and falls back to local rules when parsing or the API call fails. Even a successful call can return a weak summary or poor keywords. Local records reduce how much model wording reaches the final display, but they cannot fix a wrong summary. The confidence score is likewise a heuristic tied to ranking, not the probability that a record actually applies. The interface itself supports several languages, but the legal records and explanations are mostly in English. Translations would need separate review rather than automatic generation.

C. Privacy and Responsible Use

An incident description can include names, addresses, medical details, and evidence. When Gemini is turned on, the client library sends that text to the provider. The current repository has no consent flow, redaction, retention control, or access management. A responsible public version would keep stored personal data to a minimum, explain external processing plainly, keep model secrets on the server, and let users delete their data. The on-screen disclaimer is only a start. Users with urgent or disputed matters should be told to consult an advocate, not rely on an unreviewed model. The tool must not drive enforcement, bail, or judicial decisions without human review and official sources.

VIII. CONCLUSION AND FUTURE WORK

MargVidhi shows a workable division of labour for a small legal-education prototype. Conversational models handle the interview and summary, while the structured repository and deterministic search supply the legal sections on screen. The confirmation step, offline fallback, expandable explanations, reusable voice component, and disclaimer together make for a responsible demo that is useful in class and as a base for further work. Above all, retrieved records and model-generated text stay visibly separate.

We do not present the current build as a full legal adviser or a quantitatively validated classifier. The short-term list is clear: add unit and integration tests, build a benchmark query-to-section set, measure precision, recall, latency, usability, and legal accuracy, and put every legal entry under version control with its source. Model calls should move behind a privacy-preserving server before any production use. New legal domains can wait until their corpora get the same curation. Embedding or hybrid retrieval, multilingual explanations, audit trails, and user feedback all make sense later, but they should not blur the present emphasis on keeping model claims apart from explicitly cited texts.

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 10, October 2026

DOI: 10.17148/IARJSET.2026.131060

REFERENCES

[1]. The Motor Vehicles Act, 1988, Act No. 59 of 1988, Government of India, India Code. [Online]. Available: https://indiacode.gov.in/. Accessed: Sep. 2, 2026.

[2]. I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "LEGAL-BERT: The Muppets straight out of Law School," in Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 2898–2904, 2020, doi: 10.18653/v1/2020.findings-emnlp.261.

[3]. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Advances in Neural Information Processing Systems 33, pp. 9459–9474, 2020.

[4]. E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in Proc. 2021 ACM Conference on Fairness, Accountability, and Transparency, pp. 610–623, 2021, doi: 10.1145/3442188.3445922.

[5]. MDN Web Docs, "Web Speech API," Mozilla. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API. Accessed: Sep. 2, 2026.

[6]. Google, "Gemini API documentation," Google AI for Developers. [Online]. Available: https://ai.google.dev/gemini-api/docs. Accessed: Sep. 2, 2026.

[7]. Fuse.js, "Fuse.js documentation: Lightweight fuzzy-search library." [Online]. Available: https://www.fusejs.io/. Accessed: Sep. 2, 2026.

© IARJSET This work is licensed under a Creative Commons Attribution 4.0 International License