CaseBench: A GraphQL Benchmark for Indian Legal Text Analytics

Anonymous ACL submission

Abstract

Despite the integration of Large Language Models (LLMs) into legal workflows, several fundamental challenges remain in Legal Text Analytics (LTA). Many downstream tasks, such as determining case similarity or drafting complex legal documents, involve reasoning over large and heterogeneous data sources. Current models often struggle with factual consistency, hallucinations, and handling large contexts that integrate structured and unstructured data. To address these challenges, we introduce **CaseBench**, a new benchmark and resource that uses *GraphQL* as a retrieval mechanism for multi-modal legal data, enabling complex queries over relational tables, knowledge graphs, and vector databases. CaseBench provides data samples, query templates, and evaluation tasks designed to test the ability of LLMs to leverage GraphQL-based retrieval-augmented generation in legal contexts.

1 Introduction

Large Language Models (LLMs) have shown remarkable capabilities in style generation, short-form writing, and code generation. However, tasks in Legal Text Analytics (LTA) frequently demand more than stylistic fluency. They often require reasoning over large corpora of legal precedents, factual consistency, and precise citation of relevant documents. For example, drafting a legal petition, identifying case similarity, or answering domain-specific legal questions requires not just language proficiency but also the ability to retrieve and integrate factual information from external data sources.

LLMs, even with increased context windows, often struggle in these settings due to their tendencies to hallucinate or produce imprecise reasoning steps. Recent agentic frameworks and retrieval-augmented generation (RAG) solutions attempt to mitigate these issues by externalizing memory and retrieval functions. Instead of storing all knowledge in the model's parameters, these approaches rely on structured databases and retrieval mechanisms (e.g., SQL, vector searches, or knowledge graphs) to fetch relevant facts. This reduces hallucinations and improves factual correctness.

A key challenge, however, is integrating heterogeneous sources consistently. Legal documents may be spread across relational databases (case filings, metadata), knowledge graphs (case law citations, semantic relations), and vector databases (semantic embeddings for efficient similarity search). Orchestrating queries across these modalities can be complex, error-prone, and difficult to scale.

In this paper, we propose using *GraphQL* as a unified retrieval mechanism for LTA. GraphQL provides a single, flexible interface to query diverse backends — relational, graph, and vector databases — by integrating them behind a GraphQL schema. This can simplify the retrieval layer in agentic systems, reducing complexity and making it easier to build RAG pipelines that LLMs can leverage.

We introduce **CaseBench**, a new dataset and benchmark for evaluating LLMs on LTA tasks using GraphQL-mediated retrieval. CaseBench includes:

- Multi-modal Data Sources: We provide legal documents from Indian courts (2,286 case judgments), stored across relational tables, a Neo4j graph database, and a Milvus vector store. Each modality enriches the representation of legal knowledge, from metadata and events (relational), to citation and similarity relations (graph), to dense embeddings for semantic search (vector DB).

- GraphQL Query Samples: We show how to write GraphQL queries to retrieve structured attributes from relational DBs (e.g., PostgreSQL), graph relations from Neo4j, and vector embeddings from Milvus. These queries

can be integrated into LLM prompts or agentic frameworks to produce factually grounded answers.

- Tasks and Baselines: We propose three evaluated tasks: *Case Similarity*, *Question Answering (QA)*, and *Automatic Answer Validation*. Additionally, we discuss *Petition Drafting* as a complex, real-world application, though we do not currently provide quantitative evaluation for it.

By building on GraphQL-based retrieval, we hope CaseBench will spur research into more trustworthy and controllable LLM-based legal assistants. Our initial experiments indicate that integrating factual retrieval into generation pipelines can improve performance on tasks requiring factual precision and legal reasoning.

2 Related Work

Legal Text Analytics (LTA) has benefited from large, annotated corpora and specialized models. Benchmarks like LegalBench (Guha et al., 2023) and the ILDC corpus (Malik et al., 2021) have spurred research on legal reasoning and judgment prediction. Domain-specific models such as LegalBERT (Chalkidis et al., 2020), InLegalBERT (Paul et al., 2022), NyayaAnumana (Nigam et al., 2024), and InLegalLLaMA (Ghosh et al., 2024) highlight the utility of leveraging pre-trained language models tailored to legal corpora. Further, knowledge-based enhancements like legal knowledge graphs (Dhani et al., 2021) have supported tasks including question answering and similarity detection, enriching the legal NLP ecosystem.

While GraphQL has been widely used in the industry, publicly available GraphQL datasets have been relatively few. Recently however, there is increasing interest in generating GraphQL using large language models (Ganesan et al., 2024; Kesarwani et al., 2024; Saha et al., 2024).

While substantial progress has been made in assembling legal datasets and building GraphQL resources, these two areas have not been integrated for retrieval-augmented generation (RAG) in the legal domain. Similarly, while large-scale LTA datasets provide ample text and structured information, there is currently no GraphQL-based resource specifically designed to facilitate multi-modal retrieval—across relational tables, vector databases, and knowledge graphs—for legal text analytics tasks.

A GraphQL-based dataset tailored for legal RAG would unify access to legal documents, metadata, embeddings, and networked relations via a single schema. Such a resource could enable more controllable and verifiable queries, reducing hallucination and improving the reliability of LLM-based legal applications. We address this need by presenting a new benchmark and dataset that couples GraphQL with legal text analytics resources.

3 CaseBench Dataset

We introduce a new legal dataset consisting of 2,286 case judgments across multiple modalities, namely relational tables, a graph database, and a vector database.

Case Documents in Vector DB

We adopt the Milvus database for storing judgement's text in vector storage. Milvus is specifically designed for handling large-scale data and excels at representing unstructured documents using vector representation, which capture semantic of the documents. By converting case documents into high-dimensional vectors, we enable efficient similarity searches and retrievals. Additionally, Milvus allows us to store rich metadata associated with each document, such as case ID, date of judgment, and involved parties, ensuring that both the vectorized content and the structured metadata are accessible for advanced queries. This method provides a powerful foundation for legal data analysis, as it combines the benefits of semantic search with the flexibility of metadata-based retrieval. For facilitating semantic search, We choose to employ the Sentence Transformer, specifically utilizing the all-MiniLM-L6-v2 model with dimension 384.

Case Graph in Graph DB

We move to case graphs, which offer an optimal way to transform unstructured legal documents into structured representations. Graphs provide a more comprehensive and practical depiction of unstructured data, making them particularly suited for capturing the intracies of legal texts. In our case graphs, the nodes represent individual judgments, each associated with attributes such as case metadata, involved parties, and judgment details, effectively defining the node's characteristics. To enable relationships between these nodes, we leverage two primary sources. First, we use citations from IndianKanoon (Sinha, 2008), which connect judgments based on referenced cases, which supported using

directed edge. Second, we use recommendations of similar cases from Casemine (Yadav, 2013), enacting links based on legal relevance. These two types of relationships—citations and similarity-based edges—form the edges of our graph, creating a rich network of interconnected legal judgments that enhances the ability to analyze case dependencies and legal precedents. For managing and efficiently querying the case graph we stored them into Neo4j database, which is a graph database that provides flexibility with a schema-less design and supports real-time data processing. This constructed case graph consists of 2,286 nodes and 4,766 edges.

Case Details in Relational DB

We populate the Postgres tables to store the data in structured relations. For that, we create the schema for the various tables, and then to fill these tables we opt for two methods, manual and automated extraction. We manually extract the required values from the judgements and store them into relations. Due to the tedious nature of this task, we further populate our relations by employing an open-source large language model, LLAMA 3.1, given its efficient data extraction capabilities. It is important to note that the extracted data is reliable, as it is not generated but rather accurately sourced from the original documents. In addition, we enhance the LLM by providing few-shot examples of the extraction procedure. Afterward, we carry out post-processing to amend the extracted values, making them suitable for storage in structured relations.

Table Name	Records	Columns
casedetail	2286	5
court	951	2
event	11567	4
gpe	951	3
ground	18061	3
person	6596	7

Table 1: Relational Tables Statistics

Such queries can be extended to retrieve related cases from the graph database or fetch top-k similar embeddings from the vector database. We also provide GraphQL samples for complex retrieval, for example:

- Relational + Graph: Retrieve a case's ID, then fetch its citing cases from the Neo4j backend.

Property	Value
# Nodes	2286
# Edges	4766
# Edges (CITATION)	1159
# Edges (SIMILAR TO)	3607
# Node's Properties	32
# Node's Properties (quant)	19
# Node's Properties (categ)	9

Table 2: Case Graph Statistics

Property	Value
# Documents	2284
# Embedding Dim	384

Table 3: Vector Database Statistics

- Vector Search: Query Milvus via a GraphQL endpoint that takes a query embedding, returning semantically similar paragraphs.

This schema can be exposed to LLM-based agents. Instead of ad-hoc retrieval calls, the LLM can produce a GraphQL query string that the agent executes, returning structured JSON results to the model.

4 CaseBench Tasks

We present three core tasks—Case Similarity, Petition Drafting, and Case Brief Evaluation—that demonstrate how querying multi-modal legal data with GraphQL can support complex reasoning and text generation. All three tasks can be reframed as question answering (QA) problems where an LLM, augmented by GraphQL-based retrieval, accesses relational tables, graph databases, and vector stores to gather necessary facts before producing final answers. Through this unified QA-based evaluation, we can measure the effectiveness of retrieval-augmented generation (RAG) in solving practical legal tasks.

4.1 Case Similarity

Determining if two legal judgments are similar is essential for precedent analysis. Prior work (Dhani et al., 2021) relied on graph-based methods and domain-tuned models like LegalBERT. In our framework, an LLM can first pose GraphQL queries to fetch relevant case metadata, citations, or semantic embeddings from the vector database. By integrating these facts, the model can answer a QA-style prompt: "Are these two cases similar?" This

reduces hallucination and increases factual grounding, improving performance beyond closed-book baselines.

4.2 Petition Drafting

Drafting a petition often requires filling in missing details, citing appropriate precedents, and adhering to procedural requirements. Instead of directly generating the entire petition, we treat the task as iterative QA. The model asks targeted questions—e.g., "Which prior cases support this ground?" or "What is the correct jurisdiction?"—and issues GraphQL queries to retrieve answers. By extracting structured information and then integrating it into the petition draft, the system can incrementally produce a factual, contextually accurate petition.

4.3 Case Brief Evaluation

In evaluating a student's case brief, we must confirm factual accuracy, identify missing elements, and assess whether the brief aligns with the source judgments. Using QA prompts such as "Is the summary of the defendant's argument correct?" the model can query the graph DB for citations, retrieve semantic vectors from Milvus, and verify metadata from relational tables. This ensures that the LLM's evaluation of the brief is grounded in actual case content rather than relying on memory alone.

By formulating all three tasks as QA challenges enhanced by GraphQL-based retrieval, we unify the evaluation paradigm. The final performance metric for each task—be it identifying similarity, completing a petition draft, or evaluating a brief—boils down to the correctness and completeness of the model's answers to factual queries. This approach enables an end-to-end RAG solution that leverages multi-modal data to improve factual accuracy and reliability in legal text analytics.

5 Experiments and Results

Case Similarity

We use Dhani et al. (2021) as our GNN baseline. It leverages handcrafted features and citations, while our LLaMA-3 variants use different inputs: one model receives feature-based inputs, and another uses full document excerpts. Table 4 shows the accuracy scores. The GNN baseline achieves an accuracy of 0.536. The LLaMA-3 model conditioned on extracted features alone performs at 0.454, which is lower than the baseline. However, when provided with entire document excerpts, LLaMA-3 obtains an accuracy of 0.548, surpassing the GNN baseline. This suggests that providing richer textual context to a large language model can improve performance on the case similarity task.

Model	Accuracy
GNN	0.536
LLaMA-3 (features)	0.454
LLaMA-3 (docs)	**0.548**

Table 4: Case Similarity results. The LLaMA-3 model leveraging entire documents outperforms the GNN baseline.

GraphQL Query Generation

We conducted an additional experiment to evaluate how smaller models might reproduce the GraphQL queries generated for a subset of our questions. From our collection of 100 QA-query samples, we randomly selected 20 samples and asked three smaller models—codellama, ibm-granite, and deepseek coder—to independently generate queries for each question. We then compared their generated queries against the original queries for exact string matches. As shown in Table 5, codellama models performs the best in our expeiments.

Model	Correct Queries	Percentage
codellama 34b	19	95
ibm-granite-code 34b	14	70
deepseek coder 33b	15	75

Table 5: Accuracy of smaller models in reproducing the original GraphQL queries on 20 random samples.

6 Conclusion

We introduce CaseBench, a novel benchmark and dataset that integrates GraphQL-based retrieval with legal text analytics tasks. By providing a unified retrieval interface over structured and unstructured data, we facilitate more trustworthy and controllable LLM-based solutions in the legal domain. In the future, we plan on exploring agentic LLM frameworks that dynamically select retrieval strategies, conducting human evaluation for petition drafting, and experimenting with additional data sources.

Limitations

While we have introduced datasets for all the three legal text analytics tasks, we have compared the performance of our solution only with the case similarity baselines. The two other tasks, petition drafting and case brief evaluation do not have relevant baselines, especially in the context of legal text analytics. We hope to produce supervised methods for these tasks and compare the results in future works.

Ethics Statement

We are aware of ethical concerns in using AI systems in the legal domain. In this work, we do not propose any solutions where decision impacting people will be made by AI systems. On the contrary, we have discussed tasks that can enable people to more easily approach the judicial system. None of the datasets we have generated can be used to specifically identify any individual or organisation, even though the original documents from which we have generated such data are public domain documents released by Indian courts.

References

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Jaspreet Singh Dhani, Ruchika Bhatt, Balaji Ganesan, Parikshet Sirohi, and Vasudha Bhatnagar. 2021. Similar cases recommendation using legal knowledge graphs. *Preprint*, arXiv:2107.04771.

Balaji Ganesan, Sambit Ghosh, Nitin Gupta, Manish Kesarwani, Sameep Mehta, and Renuka Sindhgatta. 2024. Llm-powered graphql generator for data retrieval. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8657–8660. International Joint Conferences on Artificial Intelligence Organization. Demo Track.

Sudipto Ghosh, Devanshu Verma, Balaji Ganesan, Purnima Bindal, Vikas Kumar, and Vasudha Bhatnagar. 2024. Inlegalllama: Indian legal knowledge enhanced large language model. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence*.

Neel Guha, Julian Nyarko, Daniel Ho, Christopher Ré, Adam Chilton, Aditya K, et al. 2023. LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 36, pages 44123–44279.

Manish Kesarwani, Sambit Ghosh, Nitin Gupta, Shramona Chakraborty, Renuka Sindhgatta, Sameep Mehta, Carlos Eberhardt, and Dan Debrunner. 2024. Graphql query generation: A large training and benchmarking dataset. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1595–1607.

Vijit Malik, Rishabh Sanjay, Shubham Kumar Nigam, Kripabandhu Ghosh, Shouvik Kumar Guha, Arnab Bhattacharya, and Ashutosh Modi. 2021. ILDC for CJPE: Indian Legal Documents Corpus for Court Judgment Prediction and Explanation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, pages 4046–4062.

Shubham Kumar Nigam, Balaramamahanthi Deepak Patnaik, Shivam Mishra, Noel Shallum, Kripabandhu Ghosh, and Arnab Bhattacharya. 2024. Nyayaanumana & inlegalllama: The largest indian legal judgment prediction dataset and specialized language model for enhanced decision analysis. *arXiv preprint arXiv:2412.08385*.

Shounak Paul, Arpan Mandal, Pawan Goyal, and Saptarshi Ghosh. 2022. Pre-training transformers on indian legal text. *arXiv preprint arXiv:2209.06049*.

Avirup Saha, Lakshmi Mandal, Balaji Ganesan, Sambit Ghosh, Renuka Sindhgatta, Carlos Eberhardt, Dan Debrunner, and Sameep Mehta. 2024. Sequential api function calling using graphql schema. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19452–19458.

Sushant Sinha. 2008. IndianKanoon: Search Engine for Indian Law.

Aniruddha Yadav. 2013. Casemine: A granular mapping of indian case law.

A Appendix

This appendix presents additional details on the CaseBench tool, data schemas, prompts, and examples used in our experiments. We first describe the tools and interfaces that enable unified retrieval of legal data, followed by schemas and query examples. We then provide sample prompts and code snippets illustrating how we structure queries and instructions for the LLM.

A.1 CaseBench Tool and Interfaces

In this section, we provide screenshots and descriptions of the interfaces and backend systems used for creating and interacting with the CaseBench dataset. These tools facilitate the integration of relational data, knowledge graphs, and vector databases, accessible via GraphQL queries.

A.1.1 GraphQL Interface for Relational Data

Figure 1 shows the GraphQL interface for accessing relational data stored in PostgreSQL. This interface enables querying case details, courts, events, and other structured attributes using a unified schema.

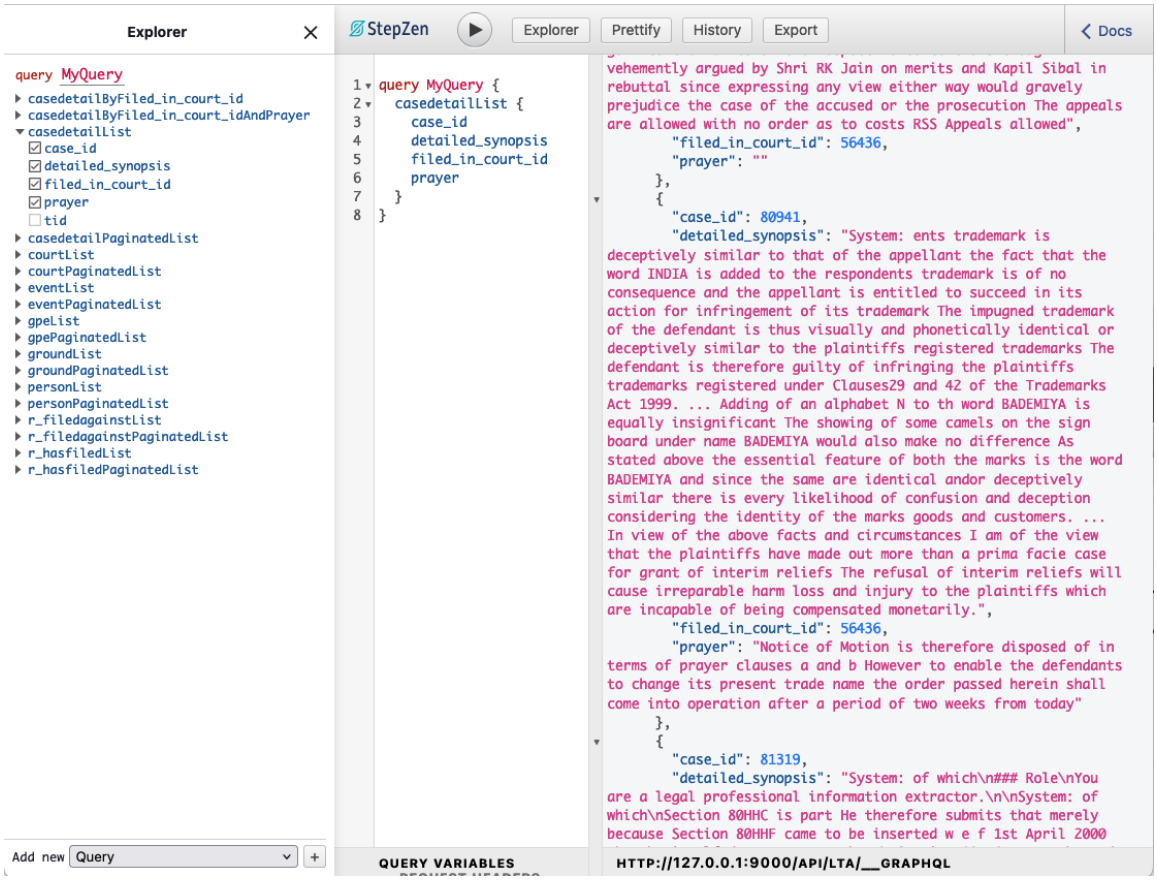

Figure 1: GraphQL interface for Relational Data

A.1.2 Case Graph in Neo4j

Figure 2 illustrates a portion of the legal knowledge graph stored in Neo4j. Nodes represent cases, and edges capture citation and similarity relationships, enabling graph-based queries to find relevant precedents and related cases.

A.1.3 Petition Drafting Application Interface

Figures 3, 4, and 5 show the user interfaces of our Petition Drafting Application. The tool provides different views for advocates and clients, guiding them through the drafting process by retrieving necessary legal facts via GraphQL queries.

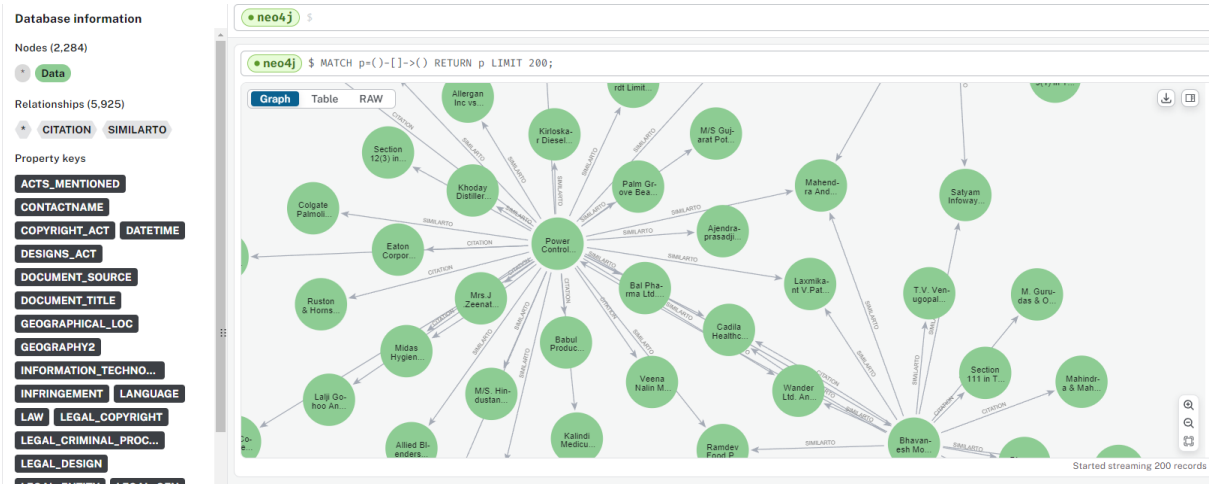

Figure 2: CaseGraph stored in Neo4j database

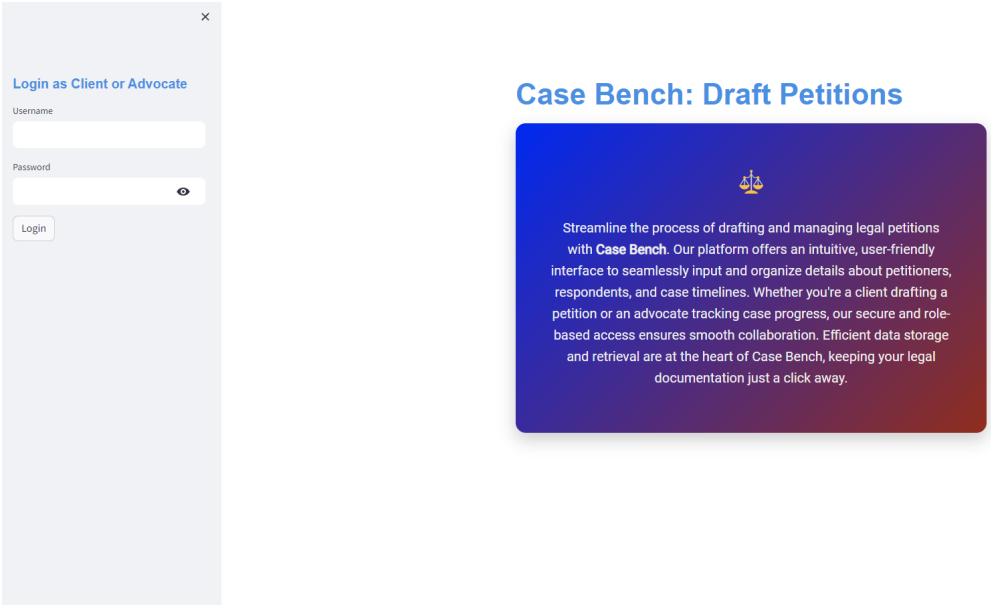

Figure 3: Login Page of the Petition Drafting Application

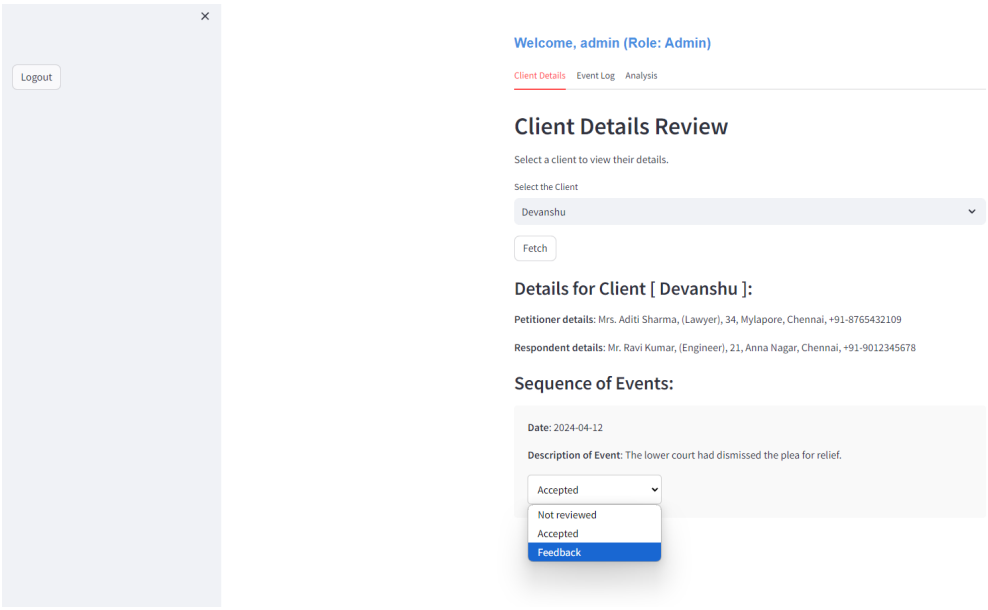

Figure 4: Advocate View for Petition Drafting

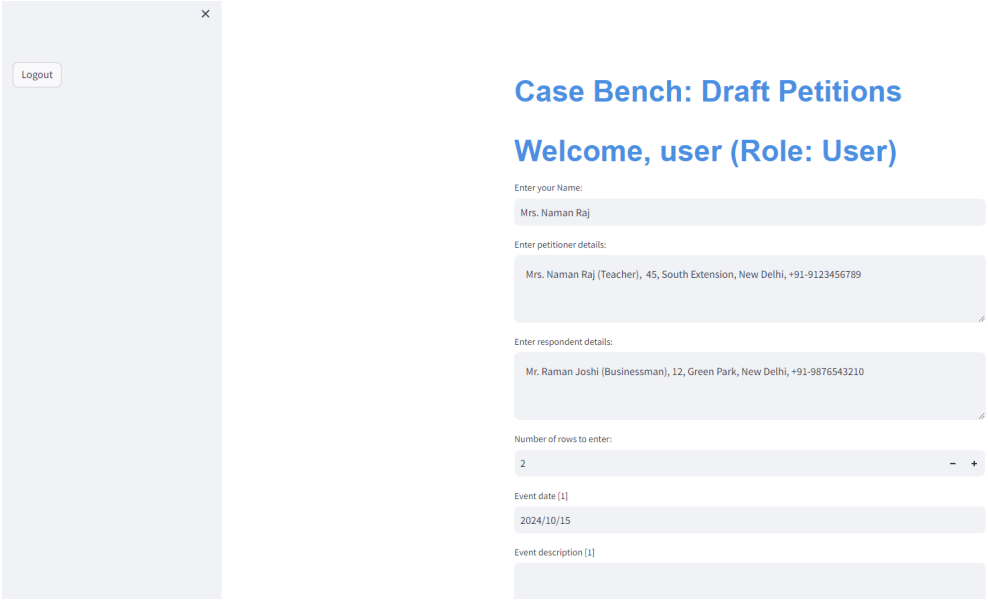

Figure 5: Client View for Petition Drafting

A.1.4 Case Brief Evaluation Interface

Figure 6 shows an example interface for evaluating a student's case brief. Here, the system can generate factual questions and retrieve authoritative information to check the brief's accuracy.

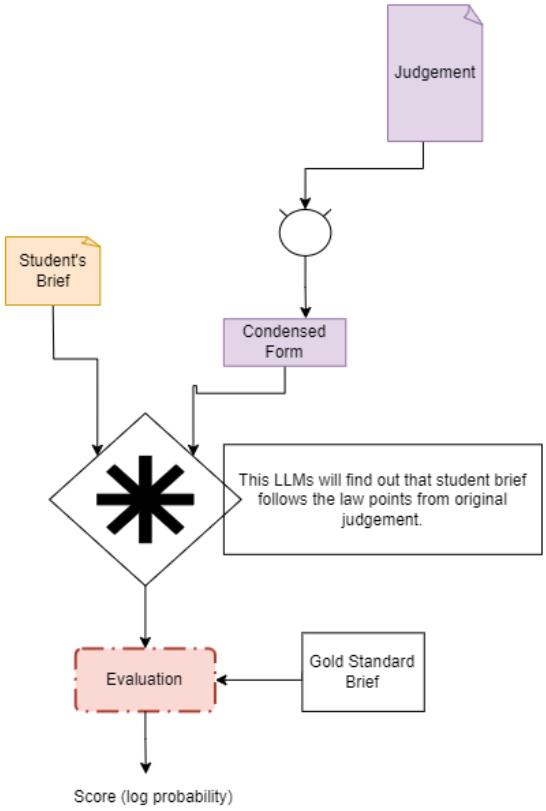

Figure 6: Case Brief Evaluation Interface

A.2 GraphQL Schemas and Queries

Below, we present an example GraphQL schema that unifies the PostgreSQL tables and describes the queries supported. This schema enables seamless retrieval of data such as case details, events, grounds, and related entities.

A.2.1 GraphQL Example Query

The following code snippet demonstrates how to query the `casedetailList` to retrieve case IDs, synopses, and prayers. More complex queries can combine relational, graph, and vector data.

```
query {
    casedetailList {
        case_id
        detailed_synopsis
        prayer
        filed_in_court_id
    }
}
```

A.2.2 GraphQL Schema Example

The schema below shows the types and queries for interacting with relational data. Similar schemas integrate graph and vector databases to achieve a unified retrieval interface.

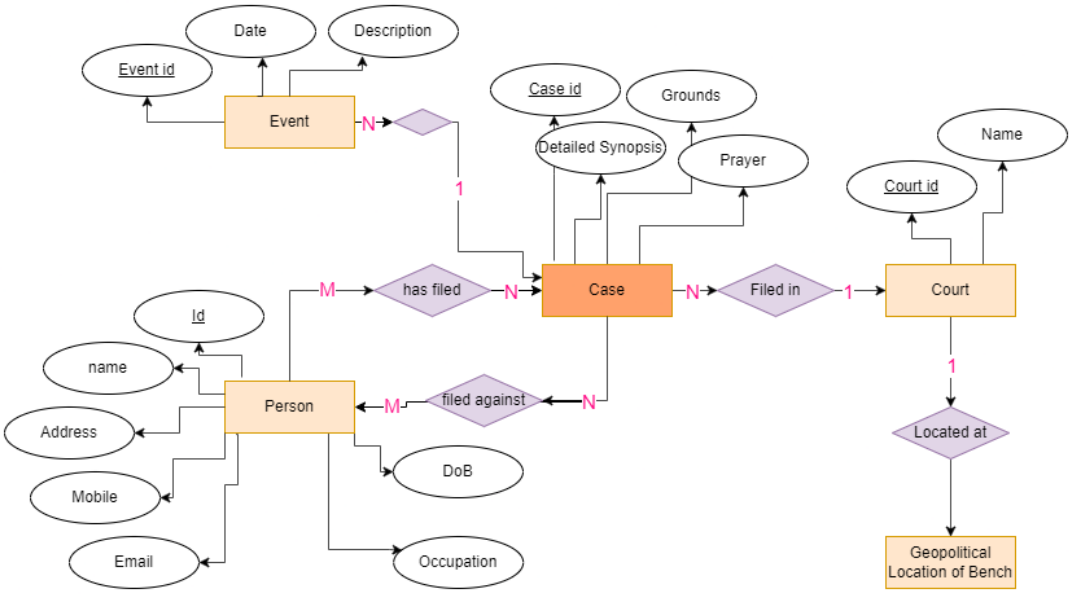

Figure 7: Schema for Petition Drafting Data

```
type Casedetail {
case_id: Int!
detailed_synopsis: String
filed_in_court_id: Int
prayer: String
tid: Int
}

type Court {
court_id: Int!
name: String
}

type Event {
date: Date
description: String
event_id: Int!
related_to_case_id: Int
}
. . .
type Query {
casedetailList: [Casedetail]
courtList: [Court]
eventList: [Event]
. . .
}
```

A.3 LLM Prompts and Examples

This section provides examples of the prompts used to guide the LLM in various tasks such as case similarity prediction and question-answer generation.

A.3.1 Case Similarity Prompts

The prompt below shows how we present pairs of legal documents along with known similarity examples to the model. After showing a few labeled examples, the model is asked to classify a new pair.

Listing 1: Sample LLaMA-2 Prompt

```
[INST]Given below are four pairs of legal documents. You have been presented with
    some parts of these documents and whether they are similar or not. Try to answer
     for Document 9 and Document 10.

### Document 1: 22. In the light of the foregoing discussion, we allow the appeals,
    set aside the impugned ...
### Document 2: the suit or application the Court should accept that the statements
    made in the plaint/application are ...
### Similarity: Yes

### Document 3: filed, unacceptable. Learned counsel for the respondent submitted
    that the Court could not have granted ...
### Document 4: right. By adopting the latter course indicated by us, the defendants
     first set would have got a fair ...
### Similarity: No

### Document 5: iv) "Hits of Salman Khan" v) "Hum Aapke Hai Kaun". However, I may
    clarify that it shall be open to ...
### Document 6: flights can not be a ground to prevent the passengers on board from
    returning to the airport lounge ...
### Similarity: No

### Document 7: An advocate abusing the process of court is guilty of misconduct.
    When witnesses are present in the ...
### Document 8: complying with the legal provisions contained in Section 309 of the
    Code. Of course, the High Court ...
### Similarity: Yes

### Document 9: the order passed by the learned Trial Judge we wish to make it clear
     that our aforesaid conclusion ...
### Document 10: ... Even in the light of the principles highlighted above when the
    evidence is tested, the inevitable ...
### Similarity: ?

Similarly, are the given parts of Legal Document 9 and Legal Document 10 similar
    apart from the fact that they contain discussions of legal terms? Give a one
    word response: Yes or No.[/INST]
```

A.3.2 Question-Answer Generation Prompts

For generating QA pairs, we provide examples and a target context. The model generates questions and answers relevant to the provided legal context.

Listing 2: Sample LLaMA-2 Prompt

```
These are the few examples of questions pertaining to the Indian Constitution,
    judiciary, legislative, and various socio-political issues in India.
<Examples>

Context:The 'Doctrine of Basic Structure' was propounded by the Indian Supreme Court
     to limit the amendment power of the Parliament. It holds ...
Question: Can you interpret the implications of the 'Doctrine of Basic Structure' in
     the Indian Constitution?

Context:To: The Hon'ble District and Sessions Judge, [Location]. Subject:
    Application for Anticipatory Bail under Section ...
Question: How would you draft an anticipatory bail application under Section 438 of
    the Code of Criminal Procedure, 1973?

Context:Honourable court, my client has been unjustly accused of defamation. However
    , as the evidence will show, my ...
Question: Can you write an opening statement for a defense attorney in a defamation
    case under Indian law?
```

A.4 Few-Shot examples for information retrieval from legal documents

Figure 8 and Figure 9 show how we provide few-shot examples to the LLM for tasks like extracting court information or parsing event descriptions. These help the model learn the format and style of the output required.

```
[
    {
    "judgement": """
        This appeal came up for hearing before the Hon'ble Supreme Court of
India.
        The judgment was delivered in open court on 15th June 2023. The Supreme
Court,
        located in New Delhi, Delhi, ruled in favor of the appellants.
        .
        .
    """,
    "extracted_values": """
        "court":
            "name": "Supreme Court of India",
            "city": "New Delhi",
            "state": "Delhi"
    """
    },
    {
    "judgement": """
        This matter was brought before the Hon'ble High Court of Karnataka.
        The judgment was pronounced on 10th March 2022 in open court.
        The High Court of Karnataka, located in Bengaluru, Karnataka, ruled
against the appellants.
        .
        .
    """,
    "extracted_values": """
        "court":
            "name": "High Court of Karnataka",
            "city": "Bengaluru",
            "state": "Karnataka",
    """
    },
    {
    "judgement": """
        The appeal was heard by the Hon'ble High Court of Kerala.
        Judgment was passed in open court on 20th August 2021.
        The High Court of Kerala, located in Ernakulam, Kerala, issued a ruling
in favor of the respondent.
        .
        .
    """,
    "extracted_values":
        "court":
            "name": "High Court of Kerala",
            "city": "Ernakulam",
            "state": "Kerala",
    """
    },
```

Figure 8: Few-shot examples to instruct the language model (court queries)

```
    "judgement": """
        On 15th June 2023, the Hon'ble Supreme Court of India delivered its
judgment in the matter of Mr. ABC vs. Mrs. XYZ. The court found in favor of the
appellant.
        On 10th March 2023, the preliminary hearing took place, where both
parties presented their arguments.
        Additionally, on 5th February 2023, the lower court had dismissed the
plea for relief.
         .
         .
    """,
    "extracted_values": """
        "events":
            "event 1":
                "date": "15th June 2023",
                "event": "The Hon'ble Supreme Court of India delivered its
judgment in the matter of Mr. ABC vs. Mrs. XYZ.",
            "event 2":
                "date": "10th March 2023",
                "event": "The preliminary hearing took place, where both parties
presented their arguments.",
            "event 3":
                "date": "5th February 2023",
                "event": "The lower court had dismissed the plea for relief."
    """
    },
    {
    "judgement": """
        On 20th April 2022, the Hon'ble High Court of Delhi delivered its
judgment in favor of the respondent in the case of Mr. Rajesh Kumar vs. State of
Delhi.
        On 15th March 2022, the court held a final hearing where both parties
presented their concluding arguments.
        Earlier, on 25th January 2022, the trial court had issued an interim
order allowing the respondent to retain possession of the disputed property.
         .
         .
    """,
    "extracted_values": """
        "events":
            "event 1":
                "date": "20th April 2022",
                "event": "The Hon'ble High Court of Delhi delivered its judgment
in favor of the respondent in the case of Mr. Rajesh Kumar vs. State of Delhi."
            "event 2":
                "date": "15th March 2022",
                "event": "The court held a final hearing where both parties
presented their concluding arguments."
            "event 3":
                "date": "25th January 2022",
                "event": "The trial court issued an interim order allowing the
```

Figure 9: Few-shot examples to instruct the language model (events queries)