

Sukkur IBA University

Computer Science Department

(2025)

AI Based Smart Lawyer

**Batch 2022**

**Moiz Ahmed (CMS-ID#023-22-0067)**

**Haseebuddin Qureshi (CMS-ID#053-22-0010)**

**M. Mubashir Hussain Shah (CMS-ID#023-22-0071)**

**Supervisor: Dr. Muhammad Hussain Mughal**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Supervisor Signature**

Table of Contents

[Table of Contents 2](#_Toc207790340)

[Abstract 3](#_Toc207790341)

[1.0 Introduction 4](#_Toc207790342)

[2.0 Background and Problem Identification 4](#_Toc207790343)

[3.0 Proposed Work and Methodology 5](#_Toc207790344)

[4.0 Project Management 12](#_Toc207790345)

[5.0 Conclusion 16](#_Toc207790346)

[References 17](#_Toc207790347)

# Abstract

Accessing and understanding Pakistani civil law is often difficult due to fragmented legal documents, scattered sources, and technical jargon. Smart Lawyer addresses this challenge by providing a bilingual (English–Urdu) intelligent assistant focused on the Civil Law domain. Built on the MERN stack, it uses a Retrieval-Augmented Generation (RAG)[1] pipeline powered by a large language model (LLM). The system ingests key civil statutes [2](e.g. Civil Procedure Code, Contract Law) and relevant Supreme and High Court judgments[3], then processes and indexes them in a vector database. Users submit queries in English or Urdu; the system retrieves relevant text fragments and feeds them into the LLM along with contextual citations to generate grounded, accurate responses. Features include advanced filtering (court, section, year), automatic summaries, and transparent citation display. Designed for both legal professionals and lay users, Smart Civil Lawyer demonstrates effective integration of modern AI techniques to improve legal accessibility in a specialized domain.

# 1.0 Introduction

Legal research is the foundation of justice, policymaking, and education. In Pakistan, however, laws are scattered across different books, websites, and court records, making them difficult to find and understand. A law student may spend hours searching through long documents for a single reference, while lawyers often face delays in locating relevant cases. This not only wastes time but also slows down the delivery of justice.

Around the world, technology has already made legal information easier to access, with platforms that quickly provide lawyers and researchers the answers they need. Unfortunately, Pakistan still lacks such modern solutions, leaving a gap between people’s needs and the resources available.

The Smart Lawyer project aims to bridge this gap by offering a simple and user-friendly platform. It will allow anyone (e.g. students, professionals, or ordinary citizens) to ask questions in plain language and instantly receive clear, reliable, and well-referenced answers. This will save time, improve understanding, and make legal knowledge more accessible to everyone.

# 2.0 Background and Problem Identification

Pakistan’s legal framework is governed by a mix of federal and provincial laws. These are documented in government gazettes, Pakistan Code[2], and judicial rulings from the Supreme Court and High Courts. However, these sources are scattered and not easily searchable. The average citizen does not have the time or expertise to explore multiple platforms to find answers.

**Key challenges include:**

1. Fragmentation of data: Legal resources are distributed across disconnected websites and archives.
2. Absence of natural language search: Users must know the exact section number or keywords to retrieve relevant information.
3. Complexity of legal language: Terms such as 'plaintiff,' 'jurisdiction,' or 'writ petition' are not easily understood by non-lawyers.
4. Lack of free centralized access: While some paid databases exist, they are unaffordable for students and the general public.

**Problem Statement:**

There is no centralized, intelligent, and user-friendly system in Pakistan that enables natural language queries, summarization of judgments, and accurate legal references with proper citations. Smart Lawyer aims to bridge this gap by developing an AI-powered assistant tailored to the Pakistani legal context.

# 3.0 Proposed Work and Methodology

**3.1 Data Collection**

Data will be curated from authentic and official sources such as the Pakistan Code, official provincial gazettes, and judgment portals of the Supreme Court and High Courts. In addition to PDFs and scanned documents, many judgments and notifications are available as HTML pages on court websites. These will be processed using structured web-scraping tools (Cheerio[4]/Puppeteer[5]). To address scanned gazettes, OCR tools such as Tesseract[6] will be used.

**3.2 Preprocessing**

After data collection, the next step is preprocessing. Raw legal data is often unstructured and contains unnecessary formatting, symbols, and noise. Preprocessing will involve **cleaning the text, removing errors, and standardizing the format** so that it can be effectively used in further stages.

For example, judgments and gazette documents often include repetitive headers, page numbers, or formatting artifacts. These will be removed. In addition, the data will be divided into **logical sections**, such as chapters, sections of law, or individual paragraphs of judgments. This ensures that the data becomes more structured and organized, making it suitable for embedding and retrieval in later stages.

**3.3 Chunking and Embeddings**

Once the data has been preprocessed, it will be broken down into **smaller, meaningful segments (chunks).** For instance, a long court judgment will be split into multiple paragraphs or clauses. This process allows the system to focus on smaller units of information, improving both accuracy and efficiency.

After chunking, embeddings will be generated for each chunk. Embeddings are mathematical vector representations of text that capture the meaning and context of words. In *Smart Lawyer*, Embeddings will be generated using Hugging Face Inference API[7] (e.g. paraphrase-multilingual-MiniLM-L12-v2 [8]) to capture semantic meaning of both Urdu and English legal texts.

**3.4 Database**

The embeddings generated from the previous step need to be stored and queried efficiently. For this purpose, Smart Lawyer will integrate a vector database along with a metadata store. Vector databases are specifically designed to handle similarity searches on high-dimensional vectors, which is essential for semantic search in legal texts.

**Qdrant (Vector Database):**

Qdrant[9] will serve as the primary vector storage engine. It can be hosted on QdrantCloud or deployed locally, making it flexible and cost-effective. Smart Lawyer will use Qdrant to store embeddings of legal text along with metadata payloads such as court name, section number, and year of judgment. This metadata allows advanced filtering of results (e.g., retrieving only Supreme Court cases or laws from a specific year). Integration will be done through LangChain’s[10]Qdrantmodule, enabling smooth orchestration of query-to-retrieval workflows.

**MongoDB Atlas (Metadata & Logs):**

In addition to vector storage, MongoDB Atlas[11] will act as a persistent database for storing non-vector data. This includes user profiles, query logs, and uploaded files.

**3.5 Retrieval-Augmented Generation (RAG)**

At the core of the Smart Lawyer system is the **Retrieval-Augmented Generation (RAG) pipeline**[1]. RAG is an advanced AI technique that combines information retrieval with large language models (LLMs). The process works as follows:

1. A user submits a query in plain English, such as “What is the procedure for filing a civil suit under Pakistani law?”.
2. The query is converted into embeddings and matched against the stored embeddings in the vector database.
3. The most relevant legal sections (e.g., provisions from the **Civil Procedure Code, 1908**)are retrieved.
4. These retrieved sections are then passed to the **language model (e.g.** Google Gemini 2.5 Flash[12]**) through API**, which generates a clear and structured response. Importantly, the response includes **citations from authentic sources** such as the Pakistan Code or Supreme Court judgments.

This approach ensures that Smart Lawyer is not just a “chatbot” but a **credible legal research assistant**, as every response is grounded in actual legal texts.

**3.6 System Development**

The technical implementation of Smart Lawyer will be done using the **MERN stack**:

* **Backend (Node.js + Express.js):** The backend **(Node.js** [13]**+ Express.js**[14]**)** will handle the processing of user queries, communication with the vector database, and interaction with the language model. It will also provide RESTful APIs to connect the frontend and backend components.
* **Frontend (React.js):** The frontend **(React.js**[15]**)** will be designed to be user-friendly and intuitive. Users will be able to type their legal queries in a search bar, view results with citations, and access case summaries in an organized format.
* **Database (MongoDB Atlas):** MongoDB[11] will be used to store metadata, user queries, and system logs. Since it is a schema-less database, it is highly flexible for managing semi-structured legal text.
* **Hosting (Vercel & Render):** The frontend will be hosted on **Vercel**[16], ensuring fast and secure content delivery. The backend APIs will be hosted on **Render**[17], which provides scalability and reliability at low cost.
* **Machine Learning Models:**  
  Smart Lawyer will employ pre-trained transformer-based models via the Hugging Face (embeddings) and Google Gemini (LLM generation) using APIs:
  + Embeddings (Hugging Face[7]) Generates semantic vector representations for Urdu & English legal texts.
  + LLM Generation (Google Gemini 2.5 Flash[12]) Receives retrieved chunks and produces natural-language, citation-backed responses.

**3.7 Datasets:**

The primary datasets for Smart Civil Lawyer will be official and authoritative Pakistani legal resources, focusing exclusively on the civil law domain. These include:

* **Pakistan Code (Federal Laws):** Official repository of statutes from the Ministry of Law and Justice[2]. Focus will be given to key civil statutes such as:
  + Civil Procedure Code (1908)
  + Contract Act (1872)
  + Limitation Act (1908)
  + Specific Relief Act (1877)
* **Provincial Law Gazettes:** Notifications, amendments, and civil statutes published at the provincial level (Punjab Gazette[7], Sindh Gazette[19], Khyber Pakhtunkhwa Gazette[20], Balochistan Gazette[21]).
* **Judgment Databases:** Supreme Court and High Court civil judgments obtained directly from official portals (Supreme Court of Pakistan website[3], Sindh High Court Case Law portal[22], Lahore High Court database[23], Islamabad High Court[24], and Peshawar High Court[25]).
* **Custom-Built Corpus:** Older scanned gazettes and judgments digitized using OCR tools (e.g., Tesseract for Urdu script). This ensures inclusion of non-digitized but relevant legal material.
* **Pakistan Laws Dataset (Hugging Face):** JSON version of 969 laws/acts from the Ministry of Law and Justice, including the Constitution of Pakistan and the Pakistan Penal Code[26].
* **Web & HTML Sources:** Certain judgments and legal notifications are published as HTML pages. These will be extracted using structured web-scraping tools (Cheerio/Puppeteer) and cleaned into a machine-readable corpus.

**3.8 Testing and Evaluation**

Testing will be carried out in multiple stages to ensure the reliability and accuracy of the system:

* **Functional Testing:** Verifying that each component (frontend, backend, database, embedding, RAG pipeline) works as expected.
* **Accuracy Testing:** Measuring how accurately the system retrieves and summarizes relevant legal information. Evaluation will use metrics such as **Precision, Recall, and F1-score**, with a target accuracy of ≥85% when benchmarked against expert-labeled queries.
* **Performance Testing:** Evaluating the system’s response time under multiple simultaneous queries.
* **User Testing:** Conducting trials with law students, teachers, and practicing lawyers to gather feedback about usability, clarity of results, and citation accuracy.

By combining technical and user-centered testing, Smart Lawyer will achieve both **reliability** and **practical usability**.

**3.9 Documentation and Reporting**

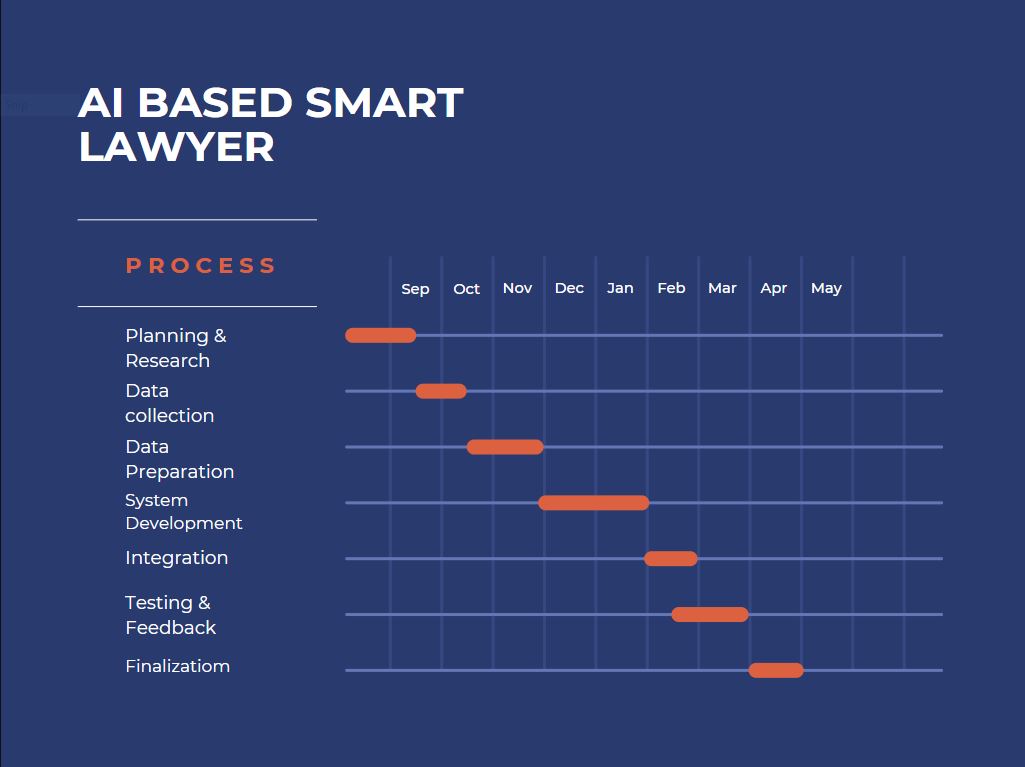
A key part of the methodology is comprehensive documentation. So, we will maintain:

* **Development Logs:** Recording data preprocessing steps, embedding generation methods, and API workflows.
* **System Design Documentation:** Explaining the architecture of Smart Lawyer, including backend, frontend, and database integration.
* **User Guide:** Providing instructions for law students and professionals to use the system effectively.
* **Final Report:** Compiling the objectives, methodology, implementation details, results, and recommendations.

This ensures the project is transparent, extendable, and ready for academic or practical use.

# 4.0 Project Management

**4.1 Workflow Gantt Chart**

****

**4.2 Project Breakdown**

|  |  |
| --- | --- |
| **Task** | **Group Member** |
| Data Collection (laws, gazettes, judgments) | All members |
| Preprocessing & Chunking | Moiz Ahmed |
| Embeddings & Vector DB Integration | Mubashir |
| Backend Development (APIs, RAG pipeline) | Haseebuddin Qureshi |
| Testing & Evaluation | All Members |
| Documentation & Final Report | All Members |
| Site Visits (If needed) | This will be done by all the group members collectively. |

|  |  |
| --- | --- |
| **Design & Implementation** | |
| Frontend UI Design (React) | Moiz Ahmed |
| Backend API Design & Database Schema Design | Haseebuddin Qureshi |
| AI/NLP Integration Design | Mubashir Hussain |
| Final Integration | All Members |

**4.3 Literature Review**

Over the past decade, researchers have explored how AI and NLP can transform the legal domain. Several notable works include:

* Legal-BERT developed by Chalkidis et al. (2020), a transformer-based model trained on European legal documents, which significantly improved legal text classification and retrieval tasks[28].
* M-RAG introduced by Wang et al. (2024), a partition-based RAG framework that improves retrieval accuracy through reinforcement learning, achieving significant gains in summarization, translation, and dialogue tasks, and demonstrating its potential for legal research applications[1].

In Pakistan, limited research exists on applying AI to the legal system. However, the increasing digitization of laws through the Pakistan Code [2] and provincial law websites provides an opportunity to create AI-powered tools. The Smart Lawyer project builds on these global examples while tailoring the solution to Pakistan’s unique challenges.

**4.4 Tools and Technologies**

 **MERN Stack (MongoDB, Express.js, React.js, Node.js):**  
Full-stack framework for developing the system. MongoDB stores metadata and user queries; Express.js manages backend APIs; React.js builds the frontend interface; Node.js handles server-side processing.

 **LangChain / RAG Framework:**  
Orchestrates the retrieval and generation pipeline, ensuring retrieved legal text is passed correctly to the LLM for context-aware responses.

 **Embeddings (Hugging Face Inference API):**  
Converts legal documents into multilingual vector embeddings for semantic search. Handles both English and Urdu legal text efficiently.

 **Vector Database (Qdrant):**  
Stores embeddings and metadata, enabling fast similarity search and advanced filtering based on court, section, or year.

 **LLM Generation (Google Gemini 2.5 Flash API):**  
Cloud-based LLM responsible for generating natural-language, citation-backed responses using retrieved embeddings. Supports bilingual output (English–Urdu) for accessible legal assistance.

 **OCR & PDF Parsing:**  
Converts scanned PDFs and digital documents into machine-readable text using (Tesseract[6], PyMuPDF[29], pdfminer[30]), ensuring inclusion of non-digitized legal material.

 **Hosting & Deployment (Vercel, Render, MongoDB Atlas):**

* **Vercel:** Hosts React frontend.
* **Render:** Hosts Node.js backend APIs.
* **MongoDB Atlas:** Cloud database for persistent storage.

 **Additional Tools:**

* **Cheerio / Puppeteer:** Structured web scraping of HTML-based judgments and gazettes.
* **Version Control & Documentation:** Git [31] for code management and project documentation for transparency and reproducibility.

**4.5 Project Feasibility**

Smart Lawyer is highly feasible due to its reliance on **open-source models, free-tier cloud services, and publicly available legal data**. The use of **Sentence-Transformers with Qdrant and LangChain** avoids the high costs of training large models while ensuring accurate semantic retrieval.

Financially, the system remains low-cost with free tiers of **MongoDB Atlas, Vercel, Render, and Qdrant Cloud**. Risks such as incomplete data or OCR errors can be managed through redundancy and manual checks.

# 5.0 Conclusion

The **Smart Lawyer** initiative represents a pivotal shift in how legal research can be conducted in Pakistan. By integrating **NLP**, **Retrieval-Augmented Generation (RAG)**, and a robust **MERN-based system**, the platform not only promises quicker access to legal sources but also ensures that responses are concise, accurately cited, and easily understandable. This tool has the potential to significantly reduce research time—saving students, professionals, and the general public countless hours each week.

Moreover, by breaking down legal jargon into plain English and fetching relevant laws immediately, Smart Lawyer bridges a long-standing gap between legal expertise and general accessibility. Over time, this platform could evolve into a nationwide legal resource, democratizing justice and encouraging informed citizen engagement. The project’s success could also inspire further AI-powered enhancements in legal technologies in Pakistan.

# References

[1] Z. Wang, S. X. Teo, J. Ouyang, Y. Xu, and W. Shi, “M-RAG: Reinforcing Large Language Model Performance through Retrieval-Augmented Generation with Multiple Partitions,” May 26, 2024, *arXiv*: arXiv:2405.16420. doi: 10.48550/arXiv.2405.16420.

[2] “Pakistan Code.” Accessed: Sept. 03, 2025. [Online]. Available: https://www.pakistancode.gov.pk/english/index.php

[3] “Supreme Court of Pakistan – Supreme Court of Pakistan.” Accessed: Sept. 03, 2025. [Online]. Available: https://www.supremecourt.gov.pk/

[4] “The industry standard for working with HTML in JavaScript | cheerio.” Accessed: Sept. 03, 2025. [Online]. Available: https://cheerio.js.org/

[5] “Puppeteer | Puppeteer.” Accessed: Sept. 03, 2025. [Online]. Available: https://pptr.dev/

[6] *tesseract-ocr/tesseract*. (Sept. 03, 2025). C++. tesseract-ocr. Accessed: Sept. 03, 2025. [Online]. Available: https://github.com/tesseract-ocr/tesseract

[7] “Access the Inference API.” Accessed: Sept. 03, 2025. [Online]. Available: https://huggingface.co/docs/huggingface\_hub/v0.13.2/en/guides/inference

[8] “sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2 · Hugging Face.” Accessed: Sept. 03, 2025. [Online]. Available: https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2

[9] “Qdrant - Vector Database.” Accessed: Sept. 03, 2025. [Online]. Available: https://qdrant.tech/

[10] “LangChain.” Accessed: Sept. 03, 2025. [Online]. Available: https://www.langchain.com/

[11] “MongoDB: The World’s Leading Modern Database | MongoDB.” Accessed: Sept. 03, 2025. [Online]. Available: https://www.mongodb.com/

[12] “Gemini Developer API Pricing | Gemini API | Google AI for Developers.” Accessed: Sept. 03, 2025. [Online]. Available: https://ai.google.dev/gemini-api/docs/pricing

[13] “Node.js — Run JavaScript Everywhere.” Accessed: Sept. 03, 2025. [Online]. Available: https://nodejs.org/en

[14] “Express - Node.js web application framework.” Accessed: Sept. 03, 2025. [Online]. Available: https://expressjs.com/

[15] “React.” Accessed: Sept. 03, 2025. [Online]. Available: https://react.dev/

[16] “Vercel: Build and deploy the best web experiences with the AI Cloud,” Vercel. Accessed: Sept. 03, 2025. [Online]. Available: https://vercel.com/home

[17] “Cloud Application Platform,” Render. Accessed: Sept. 03, 2025. [Online]. Available: https://render.com

[18] “Welcome to Punjab Laws Online.” Accessed: Sept. 03, 2025. [Online]. Available: http://www.punjablaws.gov.pk/

[19] “Law, Parliamentary Affairs & Criminal Prosecution Department, Government of Sindh.” Accessed: Sept. 03, 2025. [Online]. Available: https://www.sindhlaws.gov.pk/

[20] “List of KP Government Gazette.” Accessed: Sept. 03, 2025. [Online]. Available: https://stationery.kp.gov.pk/page/list\_of\_kp\_government\_gazette

[21] “Balochistan Code.” Accessed: Sept. 03, 2025. [Online]. Available: https://balochistancode.gob.pk/Home.aspx

[22] “Welcome to High Court of Sindh.” Accessed: Sept. 03, 2025. [Online]. Available: https://sindhhighcourt.gov.pk/

[23] “Lahore High Court.” Accessed: Sept. 03, 2025. [Online]. Available: https://lhc.gov.pk/

[24] “ISLAMABAD HIGH COURT.” Accessed: Sept. 03, 2025. [Online]. Available: https://ihc.gov.pk/

[25] “Peshawar High Court, Peshawar.” Accessed: Sept. 03, 2025. [Online]. Available: https://www.peshawarhighcourt.gov.pk/app/site/

[26] “AyeshaJadoon/Pakistan\_Laws\_Dataset · Datasets at Hugging Face.” Accessed: Sept. 03, 2025. [Online]. Available: https://huggingface.co/datasets/AyeshaJadoon/Pakistan\_Laws\_Dataset

[27] “Minimalist Blue Orange Timeline Process Gantt Chart Graph - Templates by Canva,” Canva. Accessed: Sept. 04, 2025. [Online]. Available: https://www.canva.com/templates/EAFTft0IqUc-minimalist-blue-orange-timeline-process-gantt-chart-graph/

[28] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, “LEGAL-BERT: The Muppets straight out of Law School,” Oct. 06, 2020, *arXiv*: arXiv:2010.02559. doi: 10.48550/arXiv.2010.02559.

[29] “GitHub - pymupdf/PyMuPDF: PyMuPDF is a high performance Python library for data extraction, analysis, conversion & manipulation of PDF (and other) documents.” Accessed: Sept. 03, 2025. [Online]. Available: https://github.com/pymupdf/PyMuPDF

[30] “pdfminer · PyPI.” Accessed: Sept. 03, 2025. [Online]. Available: https://pypi.org/project/pdfminer/

[31] “Git.” Accessed: Sept. 03, 2025. [Online]. Available: https://git-scm.com/