Project Report: Legal Search Engine

Abstract

Finding relevant legal cases is an important but time-consuming task for lawyers and legal researchers. This project builds an AI-powered search engine to simplify the process of discovering and analyzing Supreme Court judgments from India. The system allows users to search using natural language, view summaries, and interact with the cases using an AI chatbot. It uses tools like Google Gemini for AI, Qdrant and FAISS for fast search, and React for the frontend interface. The goal is to help lawyers and citizens save time and better understand legal documents.

Introduction

Problem Domain

Legal professionals and law students often need to refer to past judgments for legal arguments, but finding the correct case from thousands of court documents is time-consuming and inefficient.

Problem Area

- No efficient way to search legal documents using plain English.
- Existing tools lack AI-powered summarization and analysis.
- Many tools are subscription-based, limiting access.

Challenges

- Scraping dynamic and structured court data from PDFs and websites.
- Cleaning and structuring legal text from various formats.
- Making AI chatbot responses accurate and case-specific
- Ensuring fast search and retrieval.

Existing Approaches

- Keyword-based PDF search tools (e.g., CTRL+F).
- Paid platforms like SCC Online and Indian Kanoon.
- Manual research using books or online portals with basic filters.

Proposed System

An AI-based legal search engine that:

- Allows **semantic** search of judgments.
- Displays **case summaries**, metadata, and downloadable PDFs.
- Offers a **case-specific AI chatbot**.
- Uses vector search technology to retrieve relevant documents faster and more accurately.

Literature Survey

Problem Area

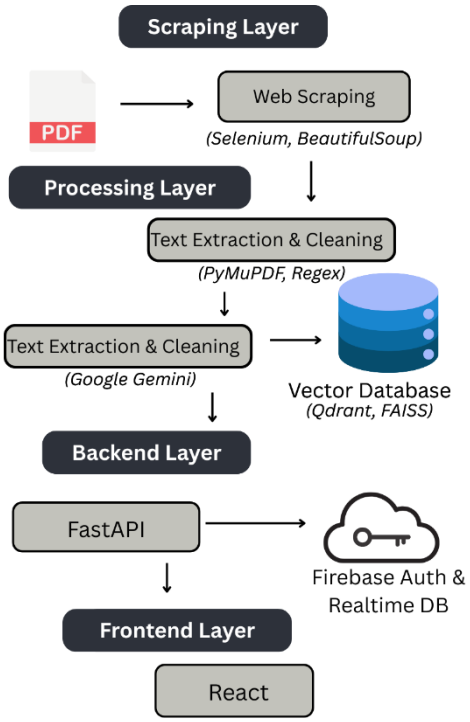
Studies show that lawyers spend over 30% of their time on legal research. This time is spent mostly reading through large volumes of case law.

Problem

Existing tools often fail to understand natural language queries like “environmental law cases after 2020” or “property disputes in Delhi”. Also, many tools don’t offer AI summaries or chat support.

NLP and LLM-based legal search systems in countries like the US are gaining popularity, proving this approach’s potential in India as well.

Proposed Solution



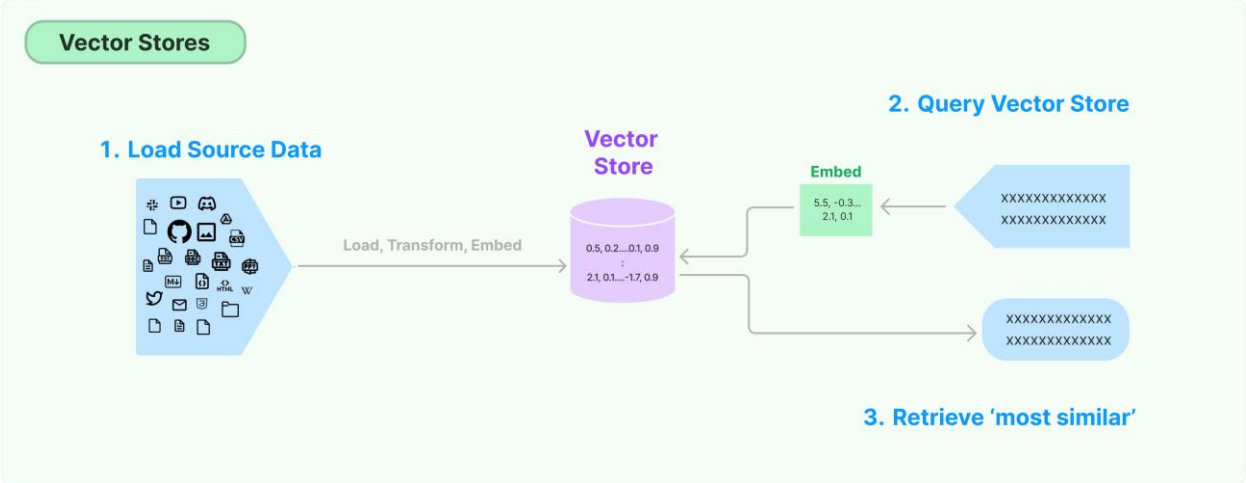
1. Scraping Layer:
   - Downloads PDFs from court websites using Selenium + BeautifulSoup.

2. Processing Layer:
   - Extracts and cleans text using PyMuPDF and Regex.
   - Chunks documents and generates vector embeddings with Google Gemini.
   - Stores vectors in Qdrant or FAISS for fast retrieval.

3. Backend Layer:
   - FastAPI server handles search, summaries, and chat endpoints.
   - Firebase handles user authentication and chat history.

4. Frontend Layer:
   - React + TailwindCSS provides an intuitive UI.
   - Supports login, search, PDF viewing, summaries, and AI chat interface.

Architectural Diagram



Implementation

Dataset

- Collected over 924 Supreme Court judgments from 2024 using automated scraping (Selenium + BeautifulSoup).
- Downloaded and stored all PDF files from LIIofIndia.
- Extracted raw text using PyMuPDF (Fitz) and cleaned using custom regex rules.

Technologies Used

Area	Tools & Libraries
Scraping	Selenium, BeautifulSoup
PDF Processing	PyMuPDF (Fitz), Regular Expressions
Text Embedding	Google Generative AI (Gemini 2.0)
Vector Database	Qdrant, FAISS
Backend	FastAPI, LangChain
Frontend	React.js, TailwindCSS, Vite
Authentication	Firebase Auth (Email/Password, Google OAuth)
Storage	Firebase Realtime Database
Search Pipeline	LangChain's Retrieval-Augmented Generation (RAG) model

Key Features

- **Smart Search:** Search using plain English like "land dispute cases after 2021".
- **AI Chatbot:** Users can chat with a judgment like "What did the court decide?".

- **Case Summary:** AI-generated summaries and key points.
- **PDF Viewer:** In-app viewing of full court documents.
- **Authentication:** Secure user login with session-based chat history.

Project Modules

1. Data Collection Module

- Scrapes Supreme Court judgments in PDF form.
- Handles dynamic content and saves PDFs locally.

2. Text Extraction & Cleaning Module

- Converts PDFs into clean, structured text.
- Removes headers, footers, and formatting issues using regex.

3. Vectorization Module

- Uses LangChain and Gemini Pro to embed documents into vector format.
- Stores embeddings in Qdrant/FAISS for fast semantic search.

4. Backend API Module

- Built with FastAPI to serve routes for:
    - Search
    - Summary generation
    - Chat interaction
    - Auth-protected endpoints

5. Frontend UI Module

- Login and Signup pages using Firebase Auth.
- Search Page with Google-like interface.
- Case Page with:
    - Summary section
    - Chatbot
    - PDF Viewer
- Chat History view using Firebase Realtime DB.

6. Chatbot Module

- LangChain + Gemini-powered RAG model.
- Responds to questions based on the selected case.

Challenges Faced

- **Web Scraping Challenges:**
    - Handling dynamically loaded elements on LII of India.
    - Implementing efficient delays to avoid IP bans.
    - Difficulty in structuring the extracted web-scraped data properly.

Next Steps

- Conduct more structured usability testing with legal professionals and law students.
- Deploy the backend (FastAPI) and frontend (React) on reliable cloud platforms such as Render, Vercel, or Firebase Hosting.
- To make the system more accessible, a **text-to-speech (TTS)** feature will be added to the chatbot.
- To generate case summaries using pre-trained models which helps users to understand key points.
- Store chat history in real time database.

Conclusion

This project successfully shows how AI can make legal research faster and easier. By combining scraping, text processing, AI models, and a responsive UI, we built a powerful legal research tool. It can help lawyers, students, and even regular people understand court judgments better.