

**AMITY SCHOOL OF ENGINEERING & TECHNOLOGY**

**Project Synopsis**

**B. Tech(CSE)**

Group No.: 169

Project Title: Smart Legal Doc Analyzer & Question Answering Tool (with RAG & Automatic Updates)

Area: Artificial Intelligence, Natural Language Processing, Machine Learning, Full-Stack Web Development, Information Retrieval

Academic Session: 2025-26

Project Guide: Dr. Geetika

**Details of Project Team:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Programme:- B.Tech CSE | | Year/Semester:- 07 | | |
| Enrollment No. | Name | | Signature |
| A2305222790 | Garv Behl | |  |
| A2305222549 | Parth Bhatia | |  |

**Abstract/Project Summary**

This project proposes the development of a "Smart Legal Doc Analyzer & Question Answering Tool" – a sophisticated AI-powered web application designed to revolutionize how individuals and legal professionals interact with legal documents and evolving laws. The primary objective is to create a system where users can securely upload their private legal documents (e.g., contracts, agreements, policies) and receive precise, contextually relevant answers to natural language questions posed about their content. What fundamentally distinguishes this solution is its integration with a dynamic, automatically updated knowledge base of public laws and regulations. The system will continuously fetch new legislation, amendments, and relevant case law summaries from official Indian government legal sources, ensuring that all information provided is current and accurate.

At its core, the system will leverage a robust Retrieval Augmented Generation (RAG) architecture. When a user queries, the system will intelligently retrieve relevant passages from *both* the user's uploaded documents *and* the up-to-date public legal knowledge base. These retrieved passages will then serve as direct context for a fine-tuned, open-source Large Language Model (LLM) to synthesize a coherent and accurate answer. This RAG approach is critical for mitigating LLM hallucinations, ensuring that responses are strictly grounded in verifiable legal text. Key functionalities will include multi-format document ingestion with OCR for scanned documents, intelligent text chunking, custom embedding generation using fine-tuned open-source models, and efficient storage and retrieval via a local vector database. Crucially, every answer provided will include granular citations, linking back to the exact sections or articles in the source documents (public or private), thereby enhancing transparency and user trust. The project aims to deliver a full-stack web application, providing a seamless user experience while demonstrating deep technical proficiency in AI, NLP, and web development. This tool promises to significantly reduce the time and complexity associated with legal research and document comprehension in an ever-changing legal landscape.

**Methodology to be adopted:-**

The project will follow an iterative and agile development methodology, structured into distinct phases, ensuring continuous integration and testing.

1. **Phase 1: Foundation & Data Ingestion Pipeline**
   * **Environment Setup:** Establish Python development environment, necessary libraries (PyTorch/TensorFlow, Hugging Face Transformers, FastAPI/Flask, React/Vue).
   * **Frontend Design (Basic Layout):** Develop initial UI/UX for document upload and Q&A interface.
   * **User Document Ingestion:** Implement robust parsers for PDF, DOCX, TXT. Integrate open-source OCR (Tesseract/PaddleOCR) for image-based PDFs.
   * **Public Data Source Identification & Initial Scrapers:** Research and identify reliable, official Indian government legal websites (e.g., India Code, Ministry of Law & Justice). Develop initial web scrapers (using BeautifulSoup, Playwright/Selenium) to extract raw legal text.
   * **Text Pre-processing:** Implement cleaning, normalization, and intelligent chunking algorithms for both user and public legal documents.
2. **Phase 2: RAG Core - Retrieval Component**
   * **Embedding Model Selection & Fine-tuning (if needed):** Choose and potentially fine-tune an open-source sentence embedding model (e.g., Sentence-BERT variants, BGE-small from Hugging Face) on a general legal corpus to enhance semantic understanding.
   * **Vector Database Implementation:** Set up and integrate a local vector database (e.g., ChromaDB, FAISS) to store document chunks' embeddings. Develop indexing routines for both user-uploaded and public legal documents.
   * **Query Embedding & Retrieval Logic:** Implement the mechanism to embed user questions and perform efficient similarity search in the vector database to retrieve relevant document chunks.
3. **Phase 3: RAG Core - Generation Component & Integration**
   * **Generative LLM Selection & Fine-tuning:** Choose a suitable open-source LLM (e.g., Flan-T5 (small variants), Mistral-7B, Llama 3 8B) and fine-tune it on a curated dataset of legal Q&A pairs.
   * **RAG Orchestration:** Integrate the retrieval component with the fine-tuned generative LLM, feeding the question and retrieved context for answer synthesis.
   * **Citation Generation:** Implement the mechanism to identify and link the LLM's answer back to the exact source chunks and their originating document/law, including version/date.
   * **Backend API Development:** Build FastAPI/Flask endpoints to handle document uploads, Q&A queries, and serve responses.
4. **Phase 4: Automatic Update System & Refinement**
   * **Scheduled Scraping:** Implement a scheduler (e.g., APScheduler) for automated, periodic execution of web scrapers for public legal data sources.
   * **Intelligent Change Detection & Incremental Indexing Logic:** Develop robust algorithms to detect new laws or amendments and trigger incremental updates to the vector database.
   * **Frontend Polish & Advanced Features:** Refine the user interface and implement advanced features.
   * **Comprehensive Testing:** Conduct thorough testing (unit, integration, end-to-end) for all features, focusing on accuracy, response time, and robustness.
   * **Deployment Preparation:** Containerize the application (Docker) for easier deployment.

**Project Management & Quality Assurance:** Iterative development with regular sprints and review meetings. Version control using Git and GitHub. Emphasis on modular code for reusability and maintainability. Thorough testing at each stage to ensure accuracy and reliability.

**Resource requirement (Hardware & software etc):-**

**Hardware:**

* **Development Machine(s):** High-performance laptops/desktops with at least 16GB RAM (32GB recommended).
* **GPU (Crucial for AI/ML):** Access to dedicated GPU(s) (e.g., NVIDIA RTX 3060/4060 or higher with 8GB+ VRAM) for LLM fine-tuning and faster inference. Cloud GPU instances (e.g., from AWS EC2, GCP Compute Engine, Azure ML) should be considered for more resource-intensive training or if local resources are insufficient.
* **Storage:** Ample SSD storage (500GB+ recommended) for datasets, models, and vector database indices.

**Software:**

* **Operating System:** Linux (Ubuntu recommended), Windows, or macOS.
* **Programming Language:** Python 3.9+
* **Deep Learning Frameworks:** PyTorch or TensorFlow (with Keras).
* **NLP/LLM Libraries:** Hugging Face Transformers, Sentence-Transformers, accelerate, bitsandbytes.
* **Vector Database/Search:** FAISS, ChromaDB, Weaviate (local instance).
* **Web Framework (Backend):** FastAPI or Flask.
* **Web Framework (Frontend):** React.js or Vue.js.
* **Database (for user data, metadata):** PostgreSQL or SQLite.
* **OCR Libraries:** Tesseract (with pytesseract wrapper) or PaddleOCR.
* **Document Parsers:** PyPDF2, python-docx, unstructured-io.
* **Web Scraping:** BeautifulSoup, Requests, Playwright or Selenium.
* **Task Scheduling:** APScheduler or Celery.
* **Version Control:** Git, GitHub.
* **Containerization:** Docker.
* **IDE:** VS Code or PyCharm.

**Justification of the project:-**

In an increasingly complex and dynamic legal environment, the need for efficient and accurate legal information retrieval is paramount. Professionals (lawyers, compliance officers) and individuals alike often struggle with comprehending intricate legal documents and staying abreast of rapidly changing laws and regulations. Existing solutions are often either manual (time-consuming research), expensive proprietary services, or AI tools that lack transparency or real-time adaptability to new legislation.

This "Smart Legal Doc Analyzer" directly addresses these pain points by offering an accessible, transparent, and intelligent platform. By enabling users to query their specific documents and simultaneously grounding answers in an automatically updated public legal knowledge base, the project provides a unique value proposition. It mitigates the risk of relying on outdated information, a critical concern in legal matters. The use of Retrieval Augmented Generation (RAG) ensures factual accuracy and provides verifiable citations, building trust in the AI's output, which is essential for legal applications. Furthermore, developing this project without reliance on commercial AI APIs serves as a robust educational endeavor, demonstrating mastery of full-stack AI development, from data ingestion and model fine-tuning to sophisticated RAG pipeline construction and full-stack web deployment. This project has the potential to significantly enhance legal literacy, streamline legal research processes, and serve as a foundation for a scalable, impactful legal tech product.

**PERT Chart/Schedule of project completion:-**

**Project Duration: Major Project (Flexible Timeline)**

**(High-Level PERT/Gantt Chart - Illustrative Milestones)**

| **Phase** | **Key Activities** |
| --- | --- |
| **Week 1-2** | Project environment setup, initial frontend design (UI layout), and tool/library installation. |
| **Week 3-4** | Implement document parsers (PDF, DOCX, OCR integration), begin public data source scrapers. |
| **Week 5-6** | Text preprocessing, normalization, and intelligent chunking of legal documents. |
| **Week 7-8** | Select and fine-tune sentence embedding model; begin setting up vector database. |
| **Week 9-10** | Complete indexing of user and public legal documents; start semantic search integration. |
| **Week 11-12** | Implement query embedding and retrieval logic; semantic search functional. |
| **Week 13-14** | Select and fine-tune generative LLM for legal Q&A; start RAG pipeline integration. |
| **Week 15-16** | Integrate retrieval with generation component; implement citation tracing logic. |
| **Week 17-18** | Backend API development for document upload and Q&A functionality. |
| **Week 19-20** | Implement automated web scraping scheduler and intelligent update logic. |
| **Week 21-22** | Refine frontend UI; enhance user experience and add advanced features. |
| **Week 23-24** | Conduct thorough testing (unit + integration), deploy system, finalize documentation. |

**References: Research papers / books / websites etc.:-**

* **Research Papers on Retrieval Augmented Generation (RAG):**
  + Lewis, P., et al. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *Advances in Neural Information Processing Systems*.
  + Guu, K., et al. (2020). "REALM: Retrieval-Augmented Language Model Pre-training." *Proceedings of the 37th International Conference on Machine Learning*.
  + Various papers on efficient vector search and indexing (e.g., FAISS, HNSW).
* **Large Language Models (LLMs) & Fine-tuning:**
  + Original Transformer paper: Vaswani, A., et al. (2017). "Attention Is All You Need." *Advances in Neural Information Processing Systems*.
  + Hugging Face Transformers library documentation and examples (huggingface.co/docs/transformers/index).
  + Papers related to specific open-source LLMs (e.g., Llama 2, Mistral, Flan-T5) and their fine-tuning methodologies.
* **Sentence Embeddings:**
  + Reimers, N., & Gurevych, I. (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.
  + Hugging Face's sentence-transformers library documentation.
* **OCR & Document Processing:**
  + Tesseract OCR documentation (tesseract-ocr.github.io/tessdoc/).
  + PaddleOCR GitHub repository ([github.com/PaddlePaddle/PaddleOCR](https://github.com/PaddlePaddle/PaddleOCR)).
  + PyPDF2 and python-docx documentation.
* **Web Scraping:**
  + BeautifulSoup documentation ([www.crummy.com/software/BeautifulSoup/bs4/doc/](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)).
  + Playwright/Selenium documentation for dynamic content scraping.
* **Vector Databases:**
  + ChromaDB documentation ([www.trychroma.com/](https://www.trychroma.com/)).
  + FAISS GitHub repository ([github.com/facebookresearch/faiss](https://github.com/facebookresearch/faiss)).
* **Legal Data Sources (Examples - Actual URLs depend on specific Indian laws/ministries):**
  + India Code (indiacode.nic.in) - for central laws of India.
  + Official websites of the Ministry of Law and Justice, Government of India.
  + The Gazette of India (egazette.nic.in).
  + Websites of specific High Courts and the Supreme Court of India for case law.

**Signature(s) of project team Name and Signature of project guide**

Date: July 17, 2025