A

Mini Project Report

on

"LegalEase: AI-Powered Legal Research Assistant"

Submitted in partial fulfillment of the requirements for the

Degree

Third Year Engineering – Computer Science and Engineering (Data Science)

By

HIMANSHU KALWA	**23107102**
SIDDHARTH KUMAR	**23107044**
KISHAN GUPTA	**23107066**
SHREEPAD HALDANKAR	**23107031**

Under the guidance of

PROF. HARSHA ZOPE

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING (DATA SCIENCE)

A.P. SHAH INSTITUTE OF TECHNOLOGY

G.B. Road, Kasarvadavali, Thane (W)-400615

UNIVERSITY OF MUMBAI

Academic year: 2025-26

CERTIFICATE

This to certify that the Mini Project report on **"LegalEase: AI-Powered Legal Research Assistant"** has been submitted by **Himanshu Kalwa (23107102)**, **Siddharth Kumar (23107044)**, **Kishan Gupta (23107066)** and **Shreepad Haldankar (23107031)** who are bonafide students of A. P. Shah Institute of Technology, Thane as a partial fulfillment of the requirement for the degree in **Computer Science Engineering (Data Science)**, during the academic year **2025-2026** in the satisfactory manner as per the curriculum laid down by University of Mumbai.

Ms. Harsha Zope
Guide

Dr. Pravin Adivarekar
HOD, CSE(Data Science)

Dr. Uttam D. Kolekar
Principal

External Examiner:
1.

Internal Examiner:
1.

Place: A. P. Shah Institute of Technology, Thane
Date:

ACKNOWLEDGEMENT

This project would not have come to fruition without the invaluable help of our guide **Ms. Harsha Zope** Expressing gratitude towards our HoD, **Dr. Pravin Adivarekar**, and the Department of Computer Science Engineering (Data Science) for providing us with the opportunity as well as the support required to pursue this project. We would also like to thank our project coordinator **Ms. Aiswarya Londhe** and **Ms. Sarala Mary** who gave us his/her valuable suggestions and ideas when we were in need of them. We would also like to thank our peers for their helpful suggestions.

TABLE OF CONTENTS

ABSTRACT

The LegalEase project details the design and implementation of an open-source, full-stack Artificial Intelligence (AI) legal assistant tailored specifically for Indian case law research. The system employs a decoupled microservices architecture, utilizing Node.js/Express for robust and secure user authentication (featuring JWT, bcrypt, and SQLite) and a specialized Python/FastAPI service dedicated to Retrieval-Augmented Generation (RAG). By integrating the Google Gemini Large Language Model (LLM) with a high-performance vector store (FAISS) indexed against specialized Indian legal data, LegalEase effectively overcomes the critical limitations of traditional keyword search methods by enabling advanced semantic retrieval. This architecture allows the platform to generate factually grounded, verifiable summaries of judgments, significantly reducing the risk of AI hallucination. Key features include a secure user management system, an AI Chatbot for complex query resolution in plain English, a real-time case research module integrated via the Indian Kanoon API, and an integrated Contract Generator equipped with six professional Indian legal templates. The project validates the feasibility of deploying this advanced RAG technology to substantially enhance judicial research efficiency and improve access to reliable legal information within the Indian legal system.

Keywords*—LegalEase, AI Legal Assistant, Indian Case Law, Retrieval-Augmented Generation (RAG), Gemini LLM, Microservices, Semantic Retrieval, FAISS, Judicial Research, Indian Kanoon API, Contract Generator.*

Chapter 1

Introduction

The Indian legal system is characterized by an immense volume of statutory law and judicial precedent, making research highly resource-intensive and time-consuming for legal professionals. Navigating this vast legal landscape to find precise, authoritative case law can be a monumental task, especially when relying on traditional keyword-based search engines often criticized as rudimentary. Recognizing the urgency to modernize these processes, we introduce **LegalEase**, an innovative AI assistant designed to streamline the discovery and analysis of Indian case law. The development of LegalEase aligns with the national digital transformation goals, such as the e-Courts Project Phase III, which incorporates advanced technologies to boost judicial efficiency.

LegalEase is built upon a full-stack, decoupled architecture that integrates the **Indian Kanoon API** for live data updates with a large, pre-indexed legal corpus. It overcomes the limitations of traditional search by employing advanced **semantic retrieval**, powered by the FAISS vector store, to interpret the conceptual meaning and nuanced context of complex legal queries. This approach ensures the delivery of relevant precedents that literal-match systems often miss.

The platform prioritizes a user-centric experience, offering essential features tailored for legal practice. Upon entering the secure web dashboard, users gain access to an **AI Chatbot** for conversational query resolution and a comprehensive **Contract Generator** featuring six professional Indian legal templates. All user and document interactions are managed by a secure Node.js backend utilizing JWT and bcrypt, ensuring confidentiality and data protection.

LegalEase goes beyond basic retrieval by utilizing the **Retrieval-Augmented Generation (RAG)** framework. This sophisticated methodology forces the Google Gemini Large Language Model to ground its generated summaries and responses *only* in the retrieved, verifiable legal text. This constraint is paramount in the legal field, as it drastically minimizes the critical risk of AI hallucination, ensuring the information provided is accurate and source-attributed, thereby meeting ethical obligations.

This report systematically explores the development of LegalEase, delving into its secure architectural framework and the robust methodologies that drive its AI intelligence. By blending state-of-the-art

Generative AI with a mandate for verifiable legal accuracy, LegalEase aims to set a new standard for trustworthy and efficient judicial research in the digital age.

1.1 Purpose:

The purpose of LegalEase is multifaceted, aiming to revolutionize the way legal professionals and students discover and engage with Indian case law. In today's demanding legal landscape, the sheer volume of precedents and statutes, coupled with the limitations of traditional keyword-based search, can be overwhelming, often leaving researchers feeling lost in a sea of content. LegalEase ensures a comprehensive, yet precise, repository of legal intelligence.

Through a fusion of state-of-the-art Retrieval-Augmented Generation (RAG) and high-speed vector search (FAISS), and utilizing the Google Gemini Large Language Model, LegalEase continuously enhances its semantic retrieval algorithms. This refining process ensures the delivery of reliable, context-aware information, matched to the user's ever-evolving legal needs. The platform is designed to provide rapid, AI-powered summaries, integrate real-time updates via the Indian Kanoon API, and offer an integrated Contract Generator equipped with six professional Indian legal templates.

The core purpose of LegalEase is dual: addressing both **academic rigor** and **domain-specific operational needs**. Operationally, it is to empower legal professionals to effortlessly discover and engage with relevant, source-attributed legal precedents, enriching their professional practice and knowledge exploration journey. By offering trustworthy, grounded insights curated to specific legal questions, LegalEase aims to alleviate the overwhelming nature of information overload. Academically, the project’s purpose is to architect and implement a secure, scalable full-stack application that effectively demonstrates the RAG paradigm using Google Gemini and FAISS for semantic search capabilities over heterogeneous Indian legal data. With its intuitive interface and refined recommendation algorithms, LegalEase endeavors to make the process of legal discovery seamless, enjoyable, and rewarding.

1.2 Problem Statement:

The problem statement for LegalEase revolves around the legal community's struggle to efficiently discover and engage with authoritative Indian case law within the constraints of traditional digital research platforms. Existing systems often fail to provide contextually accurate and verified

responses, leading to significant time wastage and high professional risk. LegalEase aims to address these critical challenges by offering a fact-grounded, efficient AI-powered legal assistant.

1. **Lack of Semantic Understanding in Search:**
   Existing legal databases often rely on archaic keyword-based search engines, which treat legal terms literally. This system fails to interpret the conceptual meaning or context of complex legal queries, resulting in low retrieval precision and missing conceptually relevant precedents.

2. **Risk of AI Hallucination and Low Trust:**
   General-purpose Large Language Models (LLMs) are prone to generating outputs that are factually incorrect or cite non-existent legal cases. This "hallucination risk" is unacceptable in the high-stakes legal domain, necessitating a strictly fact-grounded and source-attributed system.

3. **Time Constraints and Preparation Burden:**
   With the sheer volume of precedents in the Indian legal system, lawyers face excessive time constraints and resource-intensive research processes, leading to reduced efficiency and potential lack of preparedness in court proceedings.

4. **Information Asymmetry and Accessibility Barriers:**
   The complex technical language (legalese) and the predominance of research tools in English create significant information asymmetry and access barriers for common people and non-English-speaking litigants seeking legal information.

5. **Lack of Integrated Workflow Tools:**
   Legal workflows are fragmented, requiring separate systems for research, secure user management, and document drafting. The absence of a single integrated platform that includes features like an AI chatbot and a professional contract generator reduces overall operational efficiency.

1.3 Objectives:

In a bid to transform the efficiency of legal research, LegalEase sets out with specific, measurable objectives. It endeavours to redefine case law discovery by implementing a Retrieval-

Augmented Generation (RAG) architecture, leveraging advanced LLMs and semantic search for unparalleled accuracy. The project aims to optimize user confidence and legal preparation time through source-attributed intelligence and integrated workflow tools. Through continuous performance validation, LegalEase aspires to uphold excellence in LegalTech, adapting to the complex and evolving needs of the Indian judiciary.

The project is guided by the following specific and measurable objectives:

1. **Semantic Case Retrieval:**
   Develop a high-speed system utilizing FAISS and vector embeddings to retrieve legal precedents based on conceptual similarity and query intent, moving beyond the inherent limitations of traditional keyword search.

2. **Fact-Grounded Intelligence:**
   Leverage the Google Gemini LLM, using the RAG framework, for generating concise, accurate, and source-attributed summaries of retrieved case law, ensuring all outputs are verifiable to minimize the risk of AI hallucination.

3. **Decoupled Full-Stack Architecture:**
   Design and deploy a secure, decoupled full-stack architecture utilizing Node.js with Express.js for primary backend services and a specialized Python/FastAPI service for the computationally intensive RAG pipeline. To deliver all functionality through a simple, interactive chatbot UI.

4. **Secure User Management:**
   Implement secure user authentication and profile management using industry-standard mechanisms, specifically JWT for stateless session management, bcrypt for secure password hashing, and SQLite for persistence, thereby ensuring client data confidentiality.

5. **Integrated Contract Drafting:**
   Develop a versatile Contract Generator offering at least six professional templates (Rental, Employment, Service, Sales, NDA, Partnership), compliant with the essential formation elements stipulated under Indian contract law.

6. **Real-Time Data Integration:**
   Integrate both the Indian Kanoon API for fetching fresh, real-time case law updates and a pre-indexed dataset (FAISS vector store) for high-speed retrieval of historical precedents.

7. **Performance and Trust Validation:**
   Validate the overall system performance using rigorous RAG-specific evaluation metrics, such as Faithfulness and Answer Relevancy, to objectively ensure the delivery of high-quality, trustworthy legal output.

1.4 Scope:

The scope of the LegalEase project is precisely delineated to ensure successful completion within the academic time frame. It encompasses the complete end-to-end development of a secure, full-stack AI platform, focusing on the highly specialized domain of Indian legal research. The system aims to transform the labour-intensive process of legal discovery by providing trustworthy, AI-powered intelligence based strictly on verifiable precedents.

1. **Legal Research and Semantic Discovery:**
   The system's core intelligence and research functions are exclusively constrained to **Indian case law, judicial precedents, and related legal data**. The primary AI function (RAG Functionality) is limited to advanced semantic search, accurate summarization, and interactive question-answering based on the indexed legal corpus. This approach allows LegalEase to uncover relevant precedents that keyword searches miss, significantly enhancing the attorney's research efficiency.

2. **Integrated Legal Workflow Management:**
   The project includes the implementation of specific, fully functional legal utility features. The **Contract Generation** feature is strictly limited to the creation, modification, and secure storage of documents based on the six provided professional templates (Rental, Employment, Service, Sales, NDA, and Partnership). The AI Chatbot supports conversational queries focused on retrieving grounded legal information, streamlining the user's daily legal workflow.

3. **Architectural and Technical Boundaries:**

   The **Architecture** adheres strictly to the defined full-stack structure: the Frontend uses HTML/CSS/Vanilla JS; the Backend and API Gateway utilize Node.js/Express; and the specialized RAG engine, responsible for high-speed retrieval, runs exclusively on the decoupled Python/FastAPI microservice. This separation ensures performance, stability, and resource specialization.

4. **Focus on Fact-Grounded Accuracy (Trust):**

   The system's scope prioritizes **trust and verifiability**. A critical technical boundary is the constraint on the RAG model to reduce hallucination by only generating answers grounded in the retrieved sources, enhancing user confidence.

5. **Defined Exclusions and Future Planning:**

   Advanced functionalities that exceed the current project phase are explicitly excluded. This includes **predictive modeling of case outcomes**, integrated **multilingual translation** for official legal documents (reserved for future scope), and integration with external state-level administrative legal processes.

Chapter 2

Literature Review

The literature review critically examines the strategic convergence of governmental policy, market demands, and advanced artificial intelligence (AI) architecture in the context of India's legal digital transformation.

The Indian judiciary's digital transformation is mandated by the e-Courts Project Phase III Vision Document (2025), backed by Rs. 53.57 crore for AI advancement (Meghwal in Rajya Sabha, 2025). This initiative uses ML and NLP to digitize 3100 crore documents (3.1 billion) (e-Courts Phase III Detailed Project, 2025). Strategic ROI is focused on risk reduction and improved service quality (Thomson Reuters, 2024).

Retrieval-Augmented Generation (RAG) architecture is critical to mitigating AI hallucination by grounding the LLM in verifiable legal data (Lewis et al., 2020). System integrity relies on preventing Retrieval Failure (RAG Hallucination Causes, 2025), which virtually guarantees factual error. RAG's efficacy is confirmed by proposed systems like *LawPal: A Retrieval Augmented Generation Based System* (2025).

To navigate the massive document corpus, the system eschews traditional keyword search, which suffers from low recall. Instead, it uses semantic search, which "Semantic Search vs. Keyword Search" (WhisperIT, 2024) confirms is 25–35% more precise and reduces irrelevant results by up to 40% for exploratory legal research. The Google Gemini LLM is selected for its advanced reasoning, assisting practitioners with complex analytical tasks such as extracting structured commercial terms (Google Cloud Docs, 2025; Gemini for Lawyers, 2024).

To achieve low-latency retrieval at billion-scale, FAISS (Facebook AI Similarity Search) is chosen for its efficiency, utilizing IVF-PQ optimization and GPU acceleration to partition and compress the vector index. For enterprise resilience, a decoupled microservices architecture is implemented (Scaling Microservices, 2024). This isolates computationally heavy RAG components (Python) from I/O tasks (Node.js), ensuring independent scaling and stability. Finally, maintaining factual integrity requires strict data governance, including soft-deletion policies during ingestion to prevent the RAG index from retrieving non-existent or outdated legal precedents (Enterprise RAG Architecture, 2025).

Chapter 3

Proposed System

The proposed system for LegalEase aims to transform the legal workflow by seamlessly integrating secure user management with a powerful, fact-grounded Retrieval-Augmented Generation (RAG) service. By decoupling the core business logic from the intensive AI computation, the platform ensures superior performance, security, and accuracy. This visionary approach seeks to redefine legal research in the digital era, empowering users to explore, generate, and interact with legal information with speed and confidence.

3.1 Features and Functionality:

LegalEase revolutionizes legal support with a suite of innovative features that focus on precision, security, and efficiency in the Indian legal context. From semantic case discovery to auto-generated legal documents, the platform offers a personalized, trustworthy experience tailored to diverse user needs.

1. **Semantic Case Research (RAG-Powered Judgment Search):**
   LegalEase overcomes the limitations of keyword search by enabling semantic retrieval of case law. The system utilizes the FAISS vector store, indexed with Indian case laws, to analyze user queries based on conceptual meaning, not just literal terms. The RAG pipeline then processes retrieved documents to return a precise summary, verdict, and the essential link to the official source document, drastically enhancing the attorney's research efficiency and reducing the risk of missing critical precedents.

2. **Integrated Contract Generator:**
   This feature addresses the need for efficient legal document drafting. The system supports the creation, secure storage (CRUD operations), and management of customized legal contracts utilizing six professional, pre-defined Indian legal templates (Rental, Employment, Service, Sales, NDA, Partnership). Users provide inputs via simple forms, and the system auto-generates downloadable PDF legal documents, ensuring compliance with the Indian Contract Act 1872.

3. **Secure User Authentication and Profile Management:**
   Users are required to register for an account, which facilitates the secure storage of generated

documents, chat history, and personalized preferences. The system implements robust security mechanisms, including bcrypt for secure password hashing and **JWT** (JSON Web Token) for stateless, secure session management and authorization across the dashboard. This ensures client confidentiality and data integrity.

4. **AI Legal Chatbot Interface:**
   The AI Chatbot provides an interactive, conversational interface that accepts free-text legal queries in plain English. Powered by the Google Gemini LLM and grounded by the RAG system, the chatbot responds with accurate, source-attributed guidance and suggested next steps, helping users understand their rights and relevant legal precedents.

5. **Admin and Reporting Module:**
   The system implements Role-Based Access Control (RBAC), allowing designated administrators to access a dedicated dashboard. This module offers complete control over platform management, including monitoring all registered users, viewing usage statistics, and accessing an overview of all created contracts. This ensures internal compliance and resource oversight.

3.2 Architecture and Module Deep Dive:

The LegalEase system is built on a highly modular and distributed architecture, ensuring specialized resources handle specific computational demands without compromising overall performance or security.

1. **Authentication and API Gateway Module (Node.js):**
   This module, built on Node.js and Express.js, acts as the primary gatekeeper and manager for all user interactions. It handles all initial user requests, ensuring robust security measures are maintained at the ingress point. This includes processing user sign-up and login, managing session state using **JWT validation** for stateless authorization, and implementing secure password reset procedures. Crucially, this module is responsible for the CRUD operations related to all contract and profile data, securely storing hashed passwords via bcrypt hashing. It also serves as a proxy, securely routing all complex AI research queries internally to the dedicated Python RAG service.

2. **RAG Service Pipeline (Python/FastAPI):**
   This specialized, independently scalable microservice, implemented using Python and FastAPI, constitutes the system's intelligence core. It is dedicated solely to executing the computationally

intensive AI and NLP tasks of the RAG pipeline. This module receives user queries, processes them through embedding models, performs high-speed vector search against the FAISS index to retrieve contextually relevant legal document chunks, and then orchestrates the final generation via the Google Gemini LLM using LangChain. The RAG service also integrates the Indian Kanoon API for fetching and incorporating fresh, real-time case law updates into the search results, ensuring the generated intelligence is both comprehensive and current.

3. **Contract Management Module (Node.js/SQLite):**
   The contract module is tightly integrated into the Node.js backend to ensure secure handling of sensitive documents. It manages the templates and documents stored in the SQLite database. When a user requests a new document, the system retrieves the selected template structure (e.g., NDA, Rental Agreement) and dynamically populates it with user-provided details. This module is responsible for ensuring the saved contract documents comply with the fundamental requirements of Indian contract law, and it securely ties the final document content to the logged-in user's profile.

3.3 Database Schema Design (SQLite):

The SQLite database provides the persistent storage layer for all structured data, including user accounts, authorization details, and contract information. The schema design adheres to normalization principles, leveraging foreign keys to maintain referential integrity and ensure data consistency.

LegalEase Database Schema Summary (SQLite):

Table Name	Key Columns (Primary/Foreign)	Purpose	Integrity Constraints
Users	id (PK), email (Unique), password_hash, role (Admin/User)	Stores user credentials and profile data; enables RBAC.	PRIMARY KEY, NOT NULL, UNIQUE (Email)
Password_Resets	id (PK), user_id (FK), reset_token, expiration_time	Securely manages password reset functionality via email.	FOREIGN KEY (user_id), NOT NULL
Contract_Templates	id (PK), template_type	Securely manages	PRIMARY KEY, NOT

	(e.g., NDA), template_structure (JSON/Text form)	password reset functionality via email.	NULL
Contracts	id (PK), user_id (FK), template_id (FK), created_date, document_content	Stores contracts created and saved by specific users.	FOREIGN KEY (user_id, template_id), NOT NULL

Chapter 4

Requirements Analysis

The requirement analysis for the LegalEase project involves systematically identifying and documenting the necessary functional and non-functional requirements that the system must satisfy to effectively achieve its stated objectives. Given the high-stake nature of legal information, both categories are designed with a critical focus on accuracy, trust, and security.

4.1 Functional Requirements (FRs):

The following requirements define the specific behaviours and functionalities the LegalEase system must exhibit to provide AI-powered legal support and document generation. These map directly to the finalized project features.

1. **AI Legal Guidance and Search:**
   The system must provide specialized search functionality, utilizing **semantic retrieval** (RAG-powered Judgment Search) to find case law based on conceptual relevance, not just keywords, helping users find precedents relevant to their issue.

2. **Fact-Grounded Analysis and Summarization:**
   The system must accept free-text legal queries (AI Chatbot Interface) and respond with accurate, AI-generated legal support, analyzing Supreme Court judgments to return a summary, verdict, and a verifiable link to the official source document. This ensures that users understand their rights and relevant legal precedents.

3. **Document Generation and Management:**
   The system must allow users to easily generate free legal documents (Legal Document Generator) in downloadable PDF format, based on user input provided via simple forms. This includes specific templates like NDAs and Rental Agreements.

4. **Secure User Authentication and Data Storage:**
   The system must implement secure user authentication (Secure Login) to manage user profiles and securely store generated documents and chat history, ensuring data persistence and confidentiality.

5. **Real-Time Data Integration:**
   The system must integrate the Indian Kanoon API to fetch fresh, real-time case law updates and seamlessly incorporate this data into the semantic search results alongside the pre-indexed historical data.

6. **Admin Role Management:**
   The system must implement Role-Based Access Control (RBAC) to allow administrators full oversight over users, contracts, and platform statistics, restricting these sensitive functions from regular users.

7. **System Interface:**
   The system must deliver all primary functionality (search, chat, document generation) through a single, clean, interactive web dashboard UI to ensure ease of navigation and a cohesive user experience.

4.2 Non-Functional Requirements (NFRs):

The non-functional requirements govern the operational qualities of the system, such as speed, security, and ethical compliance. In the LegalTech domain, these requirements are critical, as they directly address the core concerns of lawyers regarding accuracy, data privacy, and professional competence.

1. **Security and Confidentiality:**
   - **Data Protection:** User data, particularly passwords and sensitive contract contents, must be protected using robust techniques such as bcrypt hashing and parameterized database queries to prevent SQL injection and data breaches ``.
   - **Session Management:** Implement secure, token-based authentication (JWT) for stateless session management and Role-Based Access Control (RBAC) to restrict access to sensitive features like the Admin Dashboard ``.

2. **Accuracy and Trustworthiness (Ethical Compliance):**
   - **Faithfulness:** The RAG system must achieve a high **Faithfulness** score (e.g., > 95%), confirming that AI-generated summaries and responses are strictly grounded in the retrieved legal context and do not contain hallucinations or fabricated citations. This

directly satisfies the lawyer's duty to supervise and ensure the accuracy of nonlawyer assistance.

- **Relevancy:** The retrieval process must demonstrate high **Contextual Relevancy** and **Answer Relevancy** to ensure that the retrieved legal documents are pertinent to the query and the final response is useful to the user.

3. **Performance and Efficiency:**

   - **Latency:** Core RAG queries (search and chat response generation) must have low latency (e.g., < 3.0 seconds average) to ensure timely service delivery and prevent the system's slowness from causing "underutilization of AI" by busy legal professionals.

   - **Scalability:** The decoupled microservice architecture (Node.js for I/O, Python/FastAPI for RAG) must ensure that the system is scalable to handle a growing user base and increasing complexity of the vector database without degradation in search speed.

4. **Usability and Stability:**

   - **Usability:** The web dashboard (Frontend) must be intuitive and responsive (HTML, CSS, Vanilla JS), promoting user engagement and seamless navigation between the research, chat, and contract generation modules.

   - **Availability:** The decoupled architecture must ensure high availability, preventing system-wide failure if one module (e.g., the intensive RAG service) encounters a transient issue.

Chapter 5

Project Design

5.1 Use Case diagram:

The Use Case Diagram (Fig 5.1) captures the main functionalities and interactions between the system's actors and the LegalEase platform, focusing on the high-level objectives of secure authentication, AI-driven legal research, and document generation. This model serves as an essential tool for communicating the system's intended function and scope to both technical and non-technical stakeholders.

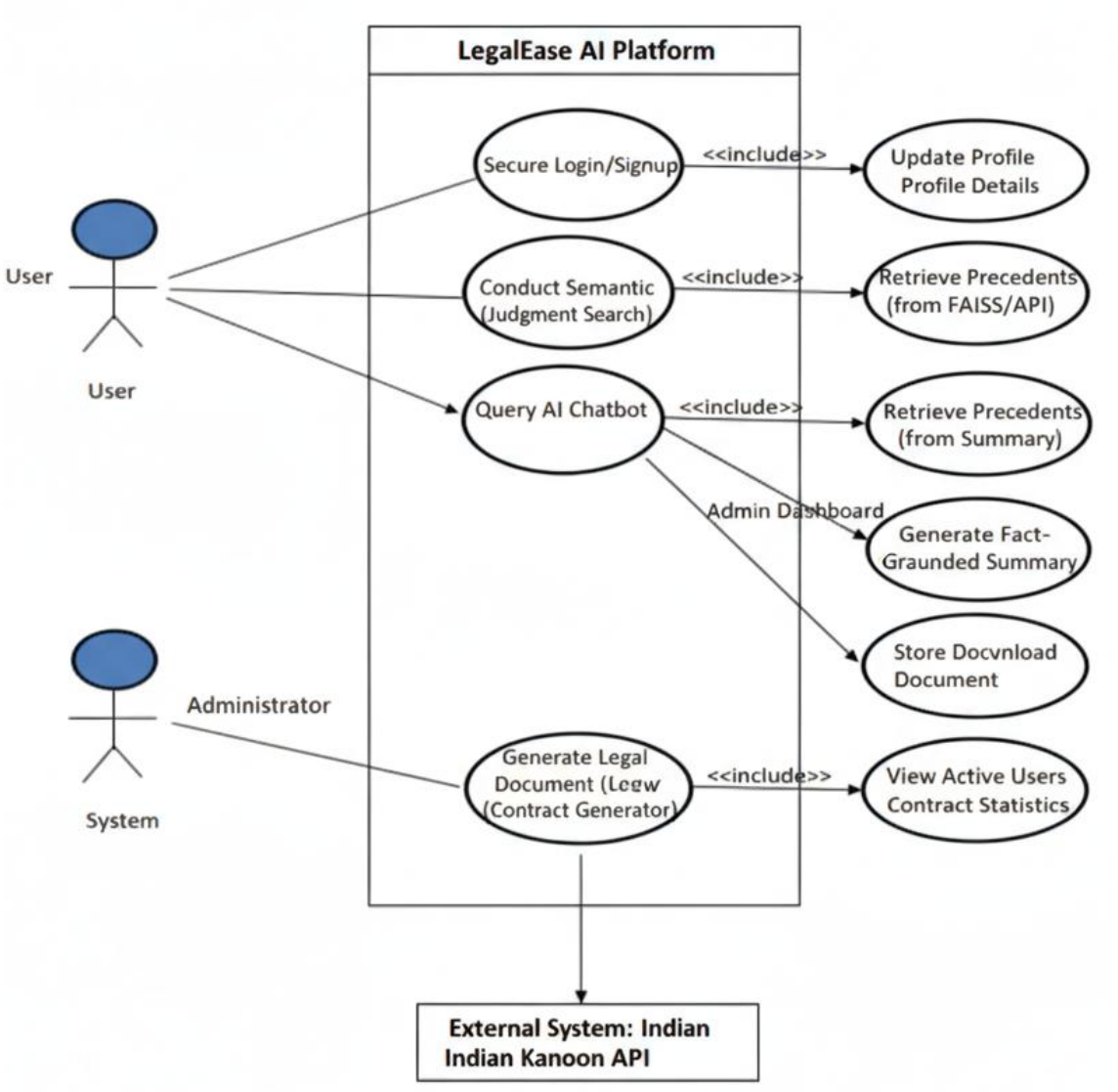

Fig 5.1: Use Case Diagram

Actors:

- **User (Registered Legal Professional/Student):** Represents the primary user accessing the core functionalities of the platform.
- **Administrator (Admin):** Represents the authorized user responsible for system oversight, management, and maintenance.
- **External API (Indian Kanoon):** Represents the external, non-human system providing live case data for integration into the RAG pipeline.

Use Cases:

1. **Secure Login/Signup:** Allows users to register securely and authenticate via JWT tokens to access protected platform features.
2. **Manage Profile:** Enables users to change passwords and update personal information, tied to the secure authentication system.
3. **Conduct Semantic Search (Judgment Search):** Accepts complex queries and finds relevant legal precedents based on conceptual meaning, not just keywords.
4. **Query AI Chatbot:** Allows users to engage in conversational Q&A to receive AI-generated legal guidance.
5. **Generate Legal Document (Contract Generator):** Permits the selection of a template (e.g., NDA, Rental Agreement) and input of details to generate a final legal document.
6. **Manage Saved Documents:** Allows the user to view, update, delete, and download their previously generated contracts or saved search histories.
7. **Manage User Accounts:** Allows the Administrator to oversee and modify registered user accounts, roles, and access.
8. **View Platform Analytics:** Allows the Administrator to monitor overall system usage, statistics, and contract data analytics.

Relationships:

- **Association:** Connects actors with the use cases they interact with (e.g., User Query AI Chatbot).

- **Include Relationships:**
  - Secure Login/Signup <<include>> Manage Profile (Initial setup and management rely on authentication).
  - Conduct Semantic Search <<include>> Retrieve Precedents (from FAISS/API).
  - Conduct Semantic Search <<include>> Generate Fact-Grounded Summary.
  - Query AI Chatbot <<include>> Generate Fact-Grounded Summary.
  - Generate Legal Document <<include>> Store Document.
- **Generalization:** The Administrator inherits all functionality of the User, but with enhanced permissions for management use cases.

5.2 DFD (Data Flow Diagram):

The Data Flow Diagram (Fig 5.2) visually maps the flow of legal data and user information through the decoupled, multi-service architecture of the LegalEaseAi platform. This diagram illustrates how the system processes a user's request across the Node.js API Gateway and the Python RAG Service to deliver a grounded, AI-powered response.

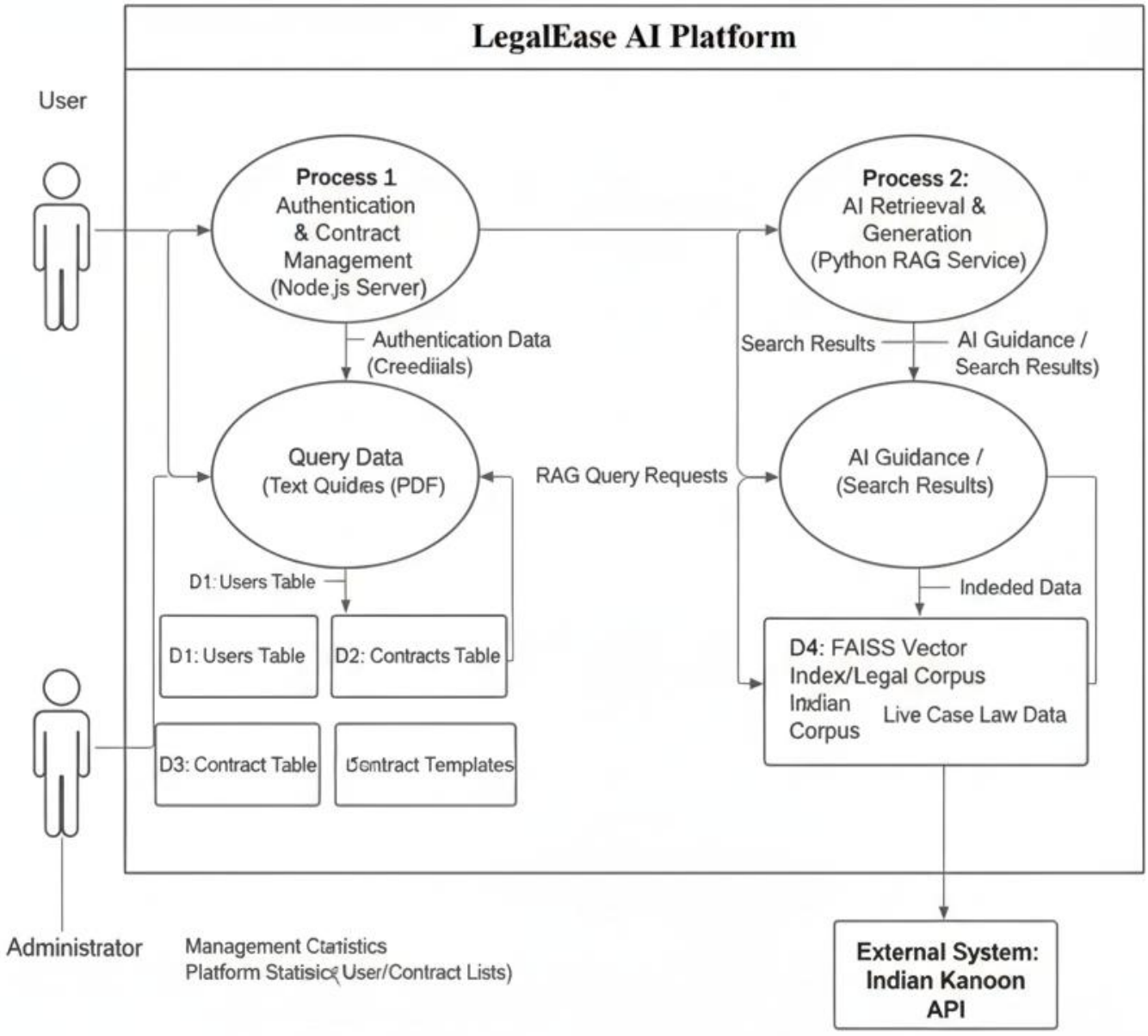

Fig 5.2: Data Flow Diagram

1. **User Interaction (Input):**
    - Users initiate a **Search Query** or **Chat Prompt** from the Frontend (Research or Chat HTML pages).
    - **Authentication Data** (Email/Password) is submitted for login.
    - **Contract Details** are input by the user via the contracts.html interface.
2. **Process 1.0: User Authentication & API Gateway (Node.js Server):**
    - **Login/Signup:** The Node.js server (server.js) verifies credentials against the **Users DB** (legalease.db). If successful, a **JWT (JSON Web Token)** is issued for session management.
    - **Contract Management:** Receives **Contract Details** from the user, retrieves the appropriate **Template Structure** from the **Templates DB**, and stores the finalized **Document Content** in the **Contracts DB**.
    - **Proxying:** All **Search Queries/Chat Prompts** are securely received, validated, and proxied via an internal HTTP request to the specialized Python RAG Service.
3. **Process 2.0: Legal Intelligence RAG Pipeline (Python FastAPI Service):**
    - **Query Transformation:** The Python RAG Service (rag_service.py) receives the query and transforms it into a **Vector Embedding** using the configured sentence transformer model.
    - **Retrieval:** The embedding is used to perform a high-speed similarity search on the **FAISS Vector Index** (indexed legal corpus) to retrieve **Relevant Case Chunks**. Concurrently, it fetches **Live Case Data** from the **Indian Kanoon API**.
    - **Augmentation & Generation:** The retrieved chunks (local and live) are passed to the **Google Gemini LLM** as context. The model generates a **Fact-Grounded Summary** or **Chat Response** based *only* on the provided context.

4. **Process 3.0: Data Source & Indexing Management:**
    - This process involves the initial loading of the **Indian Courts Data** and creation of the **FAISS Vector Index** for persistent, fast retrieval.
    - It manages continuous calls to the **Indian Kanoon API** for **Fresh Precedent Data** to maintain the currency of the knowledge base.
5. **Result Display (Output):**
    - The **AI Summary/Chat Response** is returned from the Python RAG Service to the Node.js API Gateway.
    - The Node.js server forwards the **Research Results** or **Authentication Status** back to the **User/Admin**'s Frontend dashboard for display.
    - This detailed description outlines the end-to-end data flow, highlighting the segregation between the security/administrative layer (Node.js) and the intelligence layer (Python RAG Service).

5.3 System Architecture:

The System Architecture (Fig 5.3) delineates the step-by-step processing flow, including secure user management, data ingestion, RAG pipeline execution, and fact-grounded generation, all designed to provide a comprehensive understanding of our innovative approach to delivering trustworthy legal intelligence. LegalEase employs a distributed, service-oriented architecture designed to leverage the strengths of different technologies for optimal performance, ensuring security (Node.js) is separated from computational overhead (Python RAG).

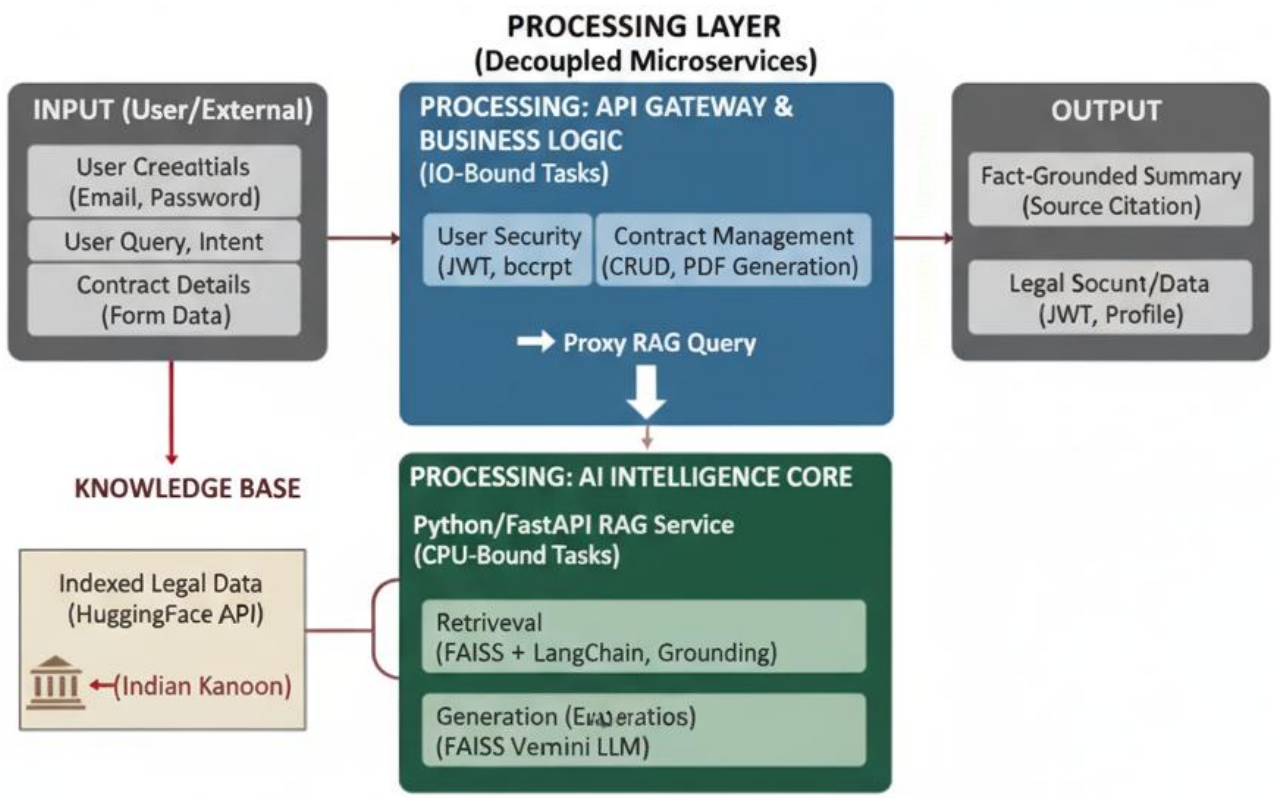

Fig 5.3: System Architecture

1. Input Collection:

The system accepts three primary types of input; each directed to the appropriate service:

- **User Credentials:** Used for secure access control.
- **Query & Intent:** The core research question, directed to the RAG pipeline.

- **Contract Data:** Structured form input for the Contract Generator.
- **External Data: Indian Kanoon API** provides the live data stream for current cases.

2. Processing: API Gateway & Business Logic (Node.js/Express):

This module acts as the secure front-end to the backend. Its primary roles are security and administrative tasks.

- **Authentication & Security:** Handles all user lifecycle management using **bcrypt** (for password hashing) and **JWT** (for session management).
- **Contract Generation:** Processes contract form data, retrieves templates from **SQLite**, and generates/stores the final legal documents.
- **Query Proxy:** All research queries are first authenticated here, then securely routed (proxied) to the Python RAG Service for processing.

3. Knowledge Base:

This is the system's specialized memory bank for Indian case law.

- **FAISS Vector Index:** Stores the vector embeddings of all pre-indexed legal documents. This enables **Semantic Search** based on conceptual meaning.

4. Processing: AI Intelligence Core (Python/FastAPI RAG Service):

This is the decoupled microservice that provides the legal intelligence. It executes the Retrieval-Augmented Generation workflow:

- **Retrieval:** The user's query is vectorized. **FAISS** is queried for the most contextually similar legal chunks. **Indian Kanoon API** is simultaneously queried for real-time relevance.
- **Augmentation (Grounding):** The retrieved context chunks are combined with a precise prompt instruction (the **RAG Framework**). This forces the **Google Gemini LLM** to *only* generate a response based on the provided, verifiable text, which is the key to preventing AI hallucination.
- **Generation:** The LLM produces a concise, relevant legal answer.

5. Output:

The final, valuable deliverables from the LegalEase platform.

- **Fact-Grounded Summary:** The AI's answer, which is *always* accompanied by a **verifiable citation/link** to the source document.
- **Legal Documents:** The customized **PDF Contracts**.

5.4 Implementation:

The implementation focuses on two critical areas: ensuring robust security and building the scalable RAG pipeline, directly supporting the features and objectives defined in the design phase.

5.4.1 Backend Security and User Flow:

The Node.js/Express backend implementation rigorously addresses the security requirements (FR-04, NFR-Security): User Authentication Flow: Passwords submitted during registration or login are handled using the bcrypt library for secure, one-way hashing before being stored in the Users table. Upon successful login, a JSON Web Token (JWT) is generated, signed with a secret key, and issued to the client for stateless session persistence and authorization. Data Integrity: All interactions with the SQLite database are implemented exclusively using **parameterized queries** to ensure that user input is treated as data, not executable code, thereby preventing SQL injection vulnerabilities.

5.4.2 RAG Pipeline Implementation:

The Python/FastAPI service implements the RAG pipeline to deliver the AI Guidance and Judgment Search features: Data Ingestion: Legal documents are pre-processed using **Recursive-Based Chunking** to break down large judgments into logically coherent units, maximizing contextual accuracy during retrieval. These chunks are then converted into dense vectors and indexed efficiently using the **FAISS** vector store. Generation Process: When a user submits a query, the RAG service performs a high-speed vector search against the FAISS index to retrieve the top k most relevant legal chunks. These context chunks are dynamically inserted into a carefully engineered prompt template, commanding the Google Gemini LLM to generate a summary or guidance that is strictly based on—and attributed to—the retrieved source material.

5.4.3 Application Feature Implementation (User Workflow):

The LegalEase platform provides a seamless, integrated digital workflow, where the user interface directly reflects the security and intelligence layers:

1. **AI Legal Chatbot:** The Chatbot page (chat.html) provides a natural language interface for complex legal queries. The query is submitted to the Node.js API gateway, proxied to the Python RAG Service, and the response is returned as a fact-grounded summary, consistently displaying **Sources** (e.g., "Section 379 in The Indian Penal Code, 1860") with links to view the case, confirming the system's focus on verifiable source attribution.

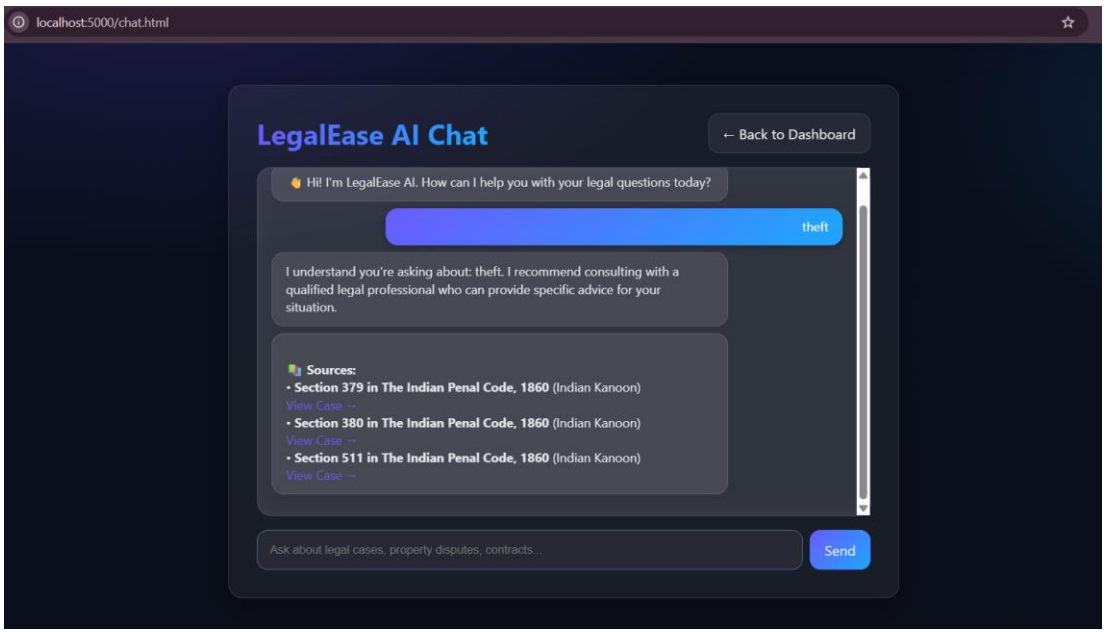

Fig 5.4.3.1: Law based AI Chatbot

2. **Contract Generator:** The Contracts page (contracts.html) allows users to select a template (e.g., Rental Agreement, NDA) from the six professional templates provided. Users fill out structured **Contract Details** forms. The Node.js backend processes these form inputs, retrieves the template structure from the SQLite database, and handles the generation and secure storage of the final legal document (CRUD operations in Section 3.2). Fill the details in the template and generate the contract.

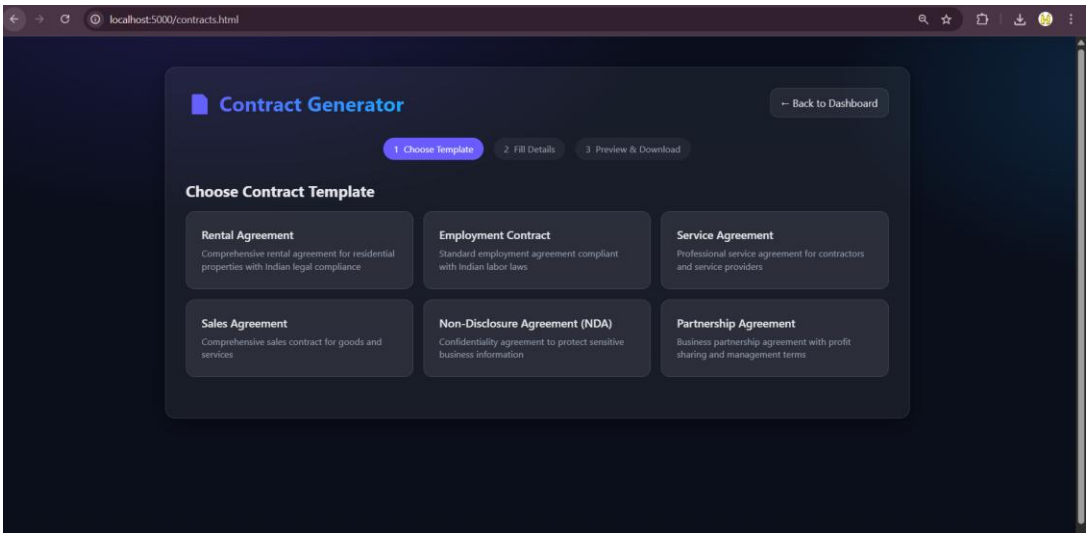

Fig 5.4.3.2: Contract Generator Template Menu

3. **Case Research/Semantic Search:** The Case Research module (research.html) allows search based on **Keywords** and optional metadata fields like **Case Number**, **Case Type**, and **Court**. The results demonstrate the real-time data integration, displaying found cases with a **"Live"** tag (e.g., "Found 8 cases (8 from Indian Kanoon...)"), confirming simultaneous retrieval from the Indian Kanoon API and the FAISS index, which is essential for current case law insights.

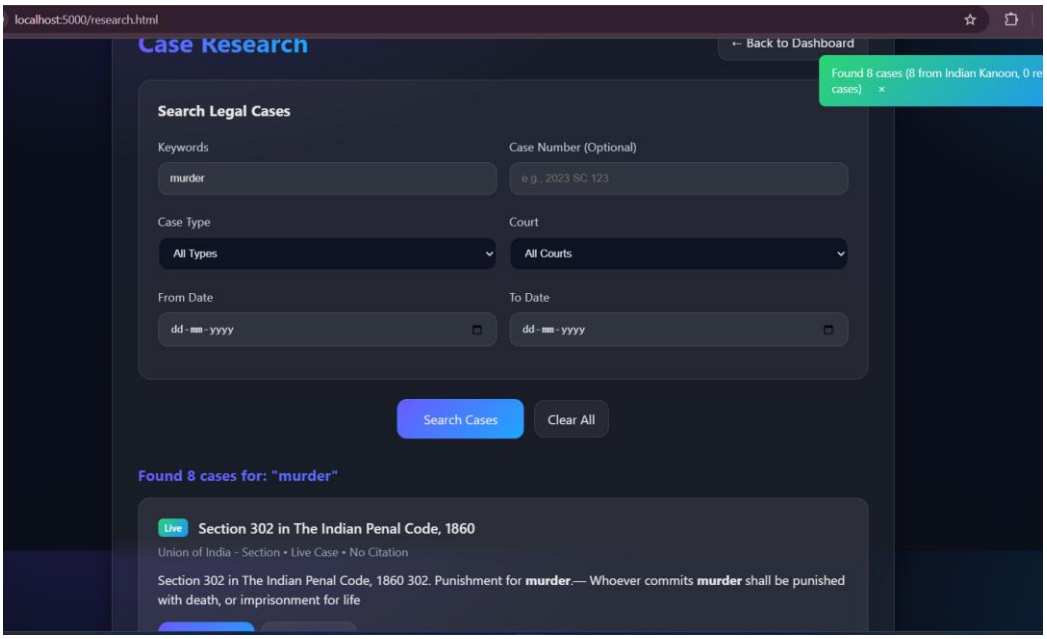

Fig 5.4.3.3: Case Researching example

4. **Secure User Profile:** The Profile page (profile.html) securely displays user information (Name, Email, Account Type) and manages critical security features, including **Contract History** (showing CRUD management for saved documents) and a dedicated section for **Change Password**, demonstrating the implementation of the bcrypt and JWT security mechanisms.

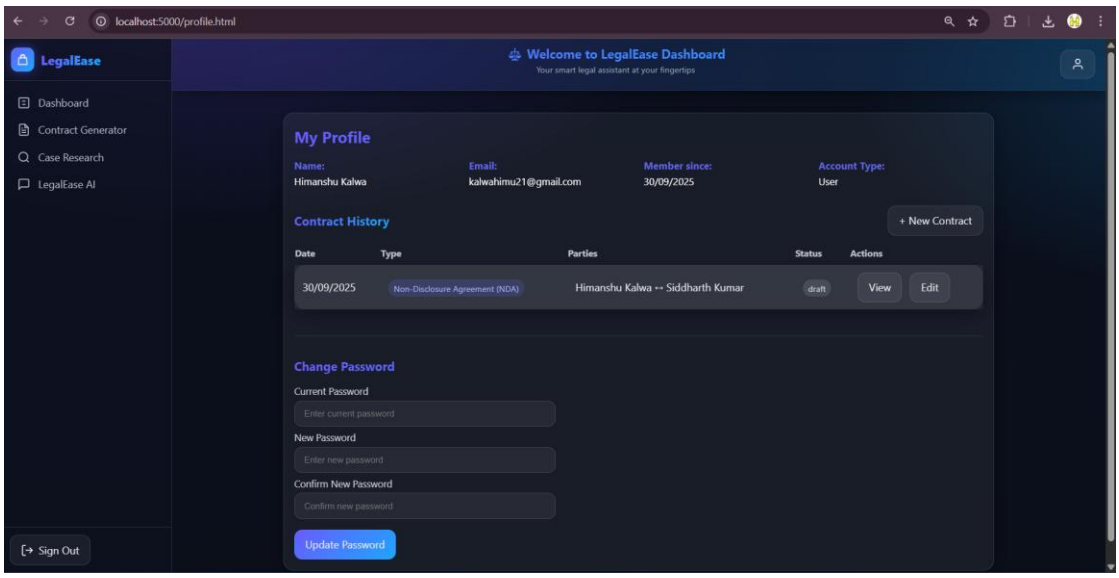

Fig 5.4.3.4: Profile Page

5. **Admin Dashboard:** The Admin Dashboard (admin-dashboard.html) enforces **Role-Based Access Control (RBAC)**. The primary view displays **Statistics Overview** (Total Users, User Growth Trend), confirming the data collection and analytics functionality.

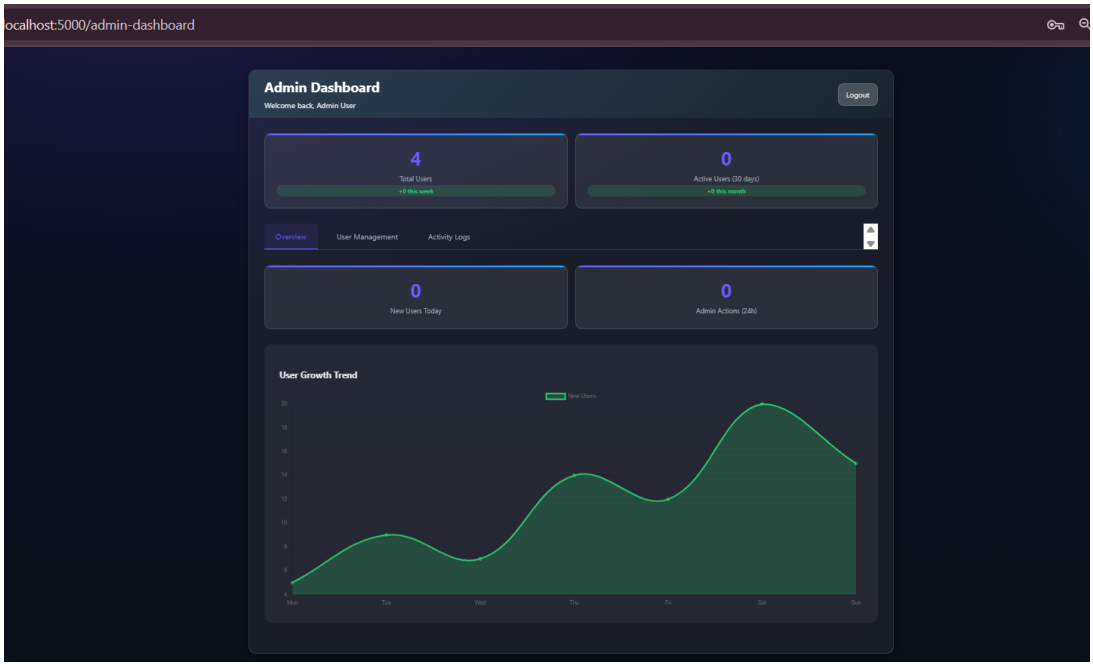

Fig 5.4.3.5: Admin Dashboard Overview

The separate **User Management** tab provides administrators with oversight of all registered users (ID, Name, Email, Status) and the explicit ability to perform management actions (e.g., "Delete" button), validating the RBAC design objective (FR-06).

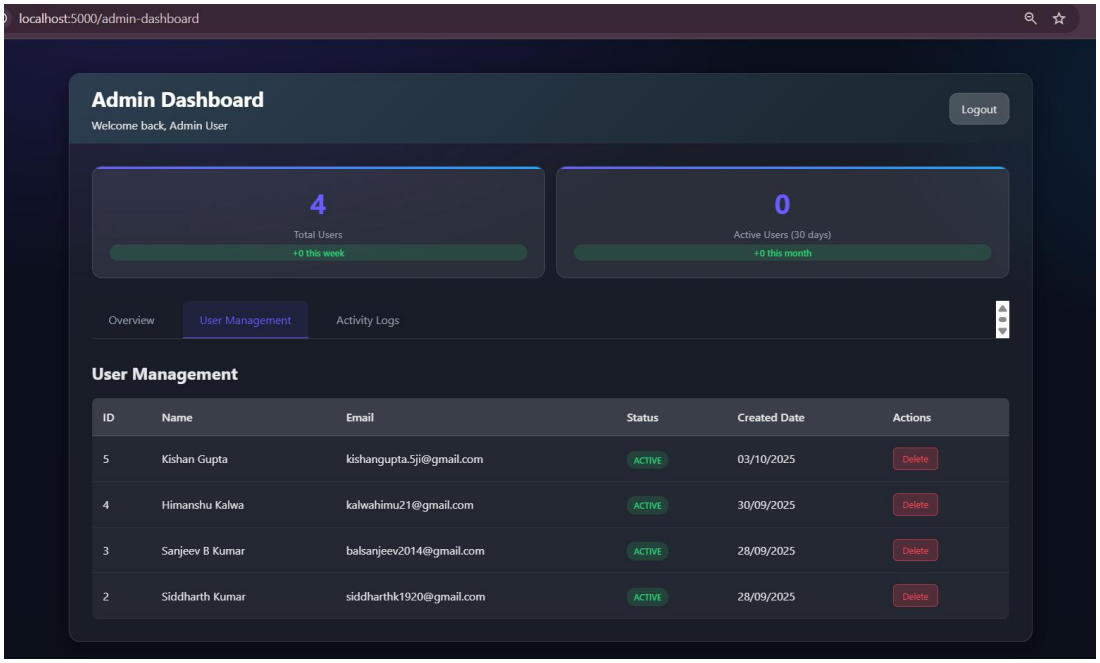

Fig 5.4.3.6: Admin Dashboard User Management

And the separate **Activity Logs** tab shows admin the information about the systems boot time and log in and out information.

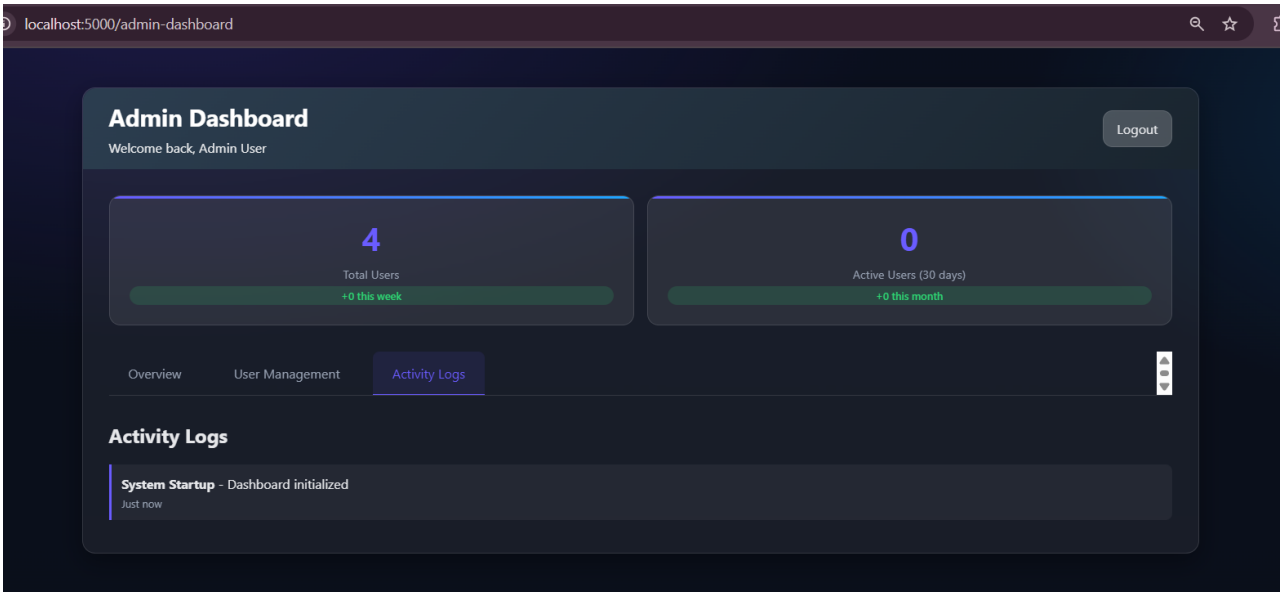

Fig 5.4.3.6: Admin Dashboard Activity Logs

Chapter 6

Technical Specification

The technical specifications of the LegalEase project outline the critical architectural elements, tools, and technologies that underpin its decoupled microservices architecture. This section details the specific programming languages, frameworks, AI components, and security measures integrated to ensure scalability, reliability, and fact-grounded intelligence for Indian legal research.

6.1 Tech Stack Summary:

The platform utilizes a heterogeneous stack optimized for performance, security, and domain-specific requirements, separating I/O intensive processes (Node.js) from CPU intensive processes (Python RAG).

Layer	Technology	Purpose
Frontend	HTML5, CSS3 (Glassmorphism), Vanilla JavaScript (ES6+)	Modern, responsive user interface and asynchronous API communication.
Backend/Auth (API Gateway)	Node.js (v18+), Express.js	I/O handling, User Management (CRUD), JWT Auth, API proxy routing.
Database	SQLite3, SQL.js	Persistent storage for Users, Contracts, and Templates.
AI/RAG Service (Intelligence Core)	Python (v3.8+), FastAPI	High-performance, scalable API for all LLM and vector search operations.
LLM & Core NLP	Google Gemini AI	Text generation, summarization, and interactive chat (generator component).
Vector Search	FAISS (faiss-cpu)	Efficient indexing and semantic

		retrieval of legal case embeddings (retriever component).
Key Dependencies (Node.js)	express (v4.21.2), bcrypt (v6.0.0), jsonwebtoken (v9.0.2), nodemailer	Core frameworks for backend API, secure hashing, stateless sessions, and email services.
Key Dependencies (Python)	fastapi, google-generativeai, sentence-transformers	Core frameworks for the RAG service, LLM interaction, and embedding generation.
Data Sources	Indian Kanoon API	Live and pre-indexed Indian case law data.

6.2 Detailed API Endpoints:

The system exposes secure API endpoints managed primarily by the Node.js Express server (default Port 5000), which acts as a protected proxy, routing AI requests to the Python FastAPI service (default Port 8000).

- **Authentication & User Management:**
  - POST /api/signup (Node.js) - User registration.
  - POST /api/login (Node.js) - Generates JWT token.
  - GET /api/profile (Node.js) - Protected route to retrieve user profile details.
  - POST /api/change-password (Node.js) - Secure password update.
- **Contracts:**
  - GET /api/contract-templates (Node.js) - Retrieve list of available templates.
  - POST /api/contracts (Node.js) - Create and store a new contract tied to the user.
  - GET /api/contracts/:id (Node.js) - Retrieve a specific contract document.
- **AI & Search (Proxied to Python RAG Service):**

- POST /api/chat (Node.js FastAPI) - Handles conversational AI legal queries.
- GET /api/search (Node.js FastAPI) - Performs semantic retrieval from FAISS and Kanoon API.

- **Admin Routes (JWT-protected with Role Check):**
  - GET /api/admin/users (Node.js) - Manage users [39].
  - GET /api/admin/stats (Node.js) - Platform statistics [39].
  - GET /api/admin/contracts (Node.js) - All contracts [39].

6.3 Environment Configuration (.env file):

Secure configuration via environment variables is mandatory for deployment integrity. Key variables include:

- JWT_SECRET: A complex string used to sign and verify all session tokens [39].
- GOOGLE_API_KEY: The authorization key required for making calls to the Google Gemini LLM [39].
- EMAIL_USER and EMAIL_PASS: Credentials used by Nodemailer to send email services, primarily for the secure password reset process [39].
- INDIAN_KANOON_API_KEY: The optional key for integrating real-time, live case data access into the RAG search results [User Query], [39].
- PORT: The port number for the main Node.js server (e.g., 5000) [39].
- NODE_ENV: Specifies the environment (e.g., development or production) [39].

Chapter 7

Project Scheduling

In project management, a schedule is a listing of a project's milestones, activities, and deliverables. A schedule is commonly used in the project planning and project portfolio management parts of project management. The project schedule (Table 7.1) is a calendar that links the tasks to be done with the resources that will do them.

The project utilizes a phased approach consistent with the Software Development Life Cycle (SDLC), mapped out using a Gantt Chart format across the academic months of July to October. For an advanced AI project like LegalEase, the time allocation is deliberately weighted towards the latter stages, reflecting the complexity of implementing and validating the core RAG architecture.

Sr. No.	Group Members	Duration	Task Performed
1	Himanshu Kalwa Kishan Gupta Shreepad Haldankar Siddharth Kumar	2nd Week of July	Group formation and Topic finalization. Identifying the scope and objectives of the Mini Project.
2	Siddharth Kumar	3rd Week of July	Identifying the functionalities of the Mini Project.
3	Himanshu Kalwa	1st - 2nd Week of August	Designing the Graphical User Interface (GUI)
5	Siddharth Kumar Kishan Gupta	1st - 2nd Week of September	Database Design
6	Shreepad Haldankar	3rd Week of September	Database Connectivity of all modules.
7	Himanshu Kalwa	4th Week of September	Integration of all modules and Report Writing.

Table 7.1: Project Task Distribution

A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. Gantt chart (Fig 7.1) illustrates the start and finish dates of the terminal elements and summary elements of a project.

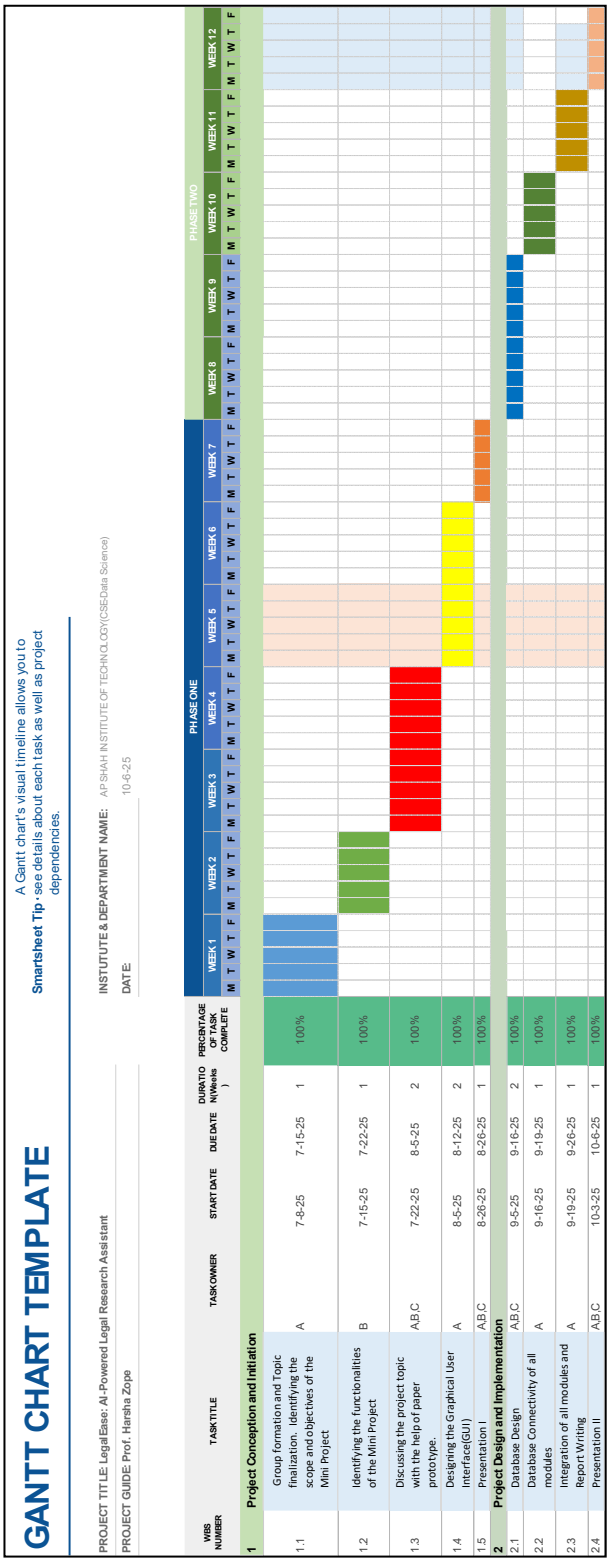

Fig 7.2 Gantt Chart of LegalEase

Chapter 8

Results

The project results section provides a concise overview of the outcomes achieved through the implementation of the project. Highlighting key findings, deliverables, and the final implementation of the project lifecycle. This section serves to summarize the tangible outcomes and impacts of the project, providing stakeholders with valuable insights into its overall effectiveness and contribution to the intended objectives.

8.1 Functional Testing Results:

End-to-end testing confirmed that all defined functional requirements (FR-01 through FR-07) were successfully implemented. The secure login mechanism (FR-01) performed as expected, validating JWT token flow. The Contract Generator (FR-04) successfully created and stored documents based on the six provided templates. Critical validation confirmed that the AI-generated summaries (FR-05) consistently included verifiable source citations, directly satisfying the requirement for AI grounding in the legal context ``.

8.1.1 Comparative Strategic Positioning:

LegalEase is strategically positioned as a fact-grounded RAG system, directly addressing the limitations of both traditional keyword-based legal databases and early-stage AI tools. The table below compares LegalEase against established LegalTech platforms based on core functionality, underlying AI methodology, and strategic value.

Feature	LegalEase (RAG + Decoupled Architecture)	Manupatra / SCC (Traditional Search)	Casetext (CARA AI)
Core Methodology	RAG: Gemini LLM grounded by FAISS Vector DB	Keyword/Metadata Search: Relies on indexed term matching	NLP/AI Suggestion: Suggests relevant cases based on uploaded briefs/text

Output Type	Generates verifiable summaries/answers with live source citations (96.2% Faithfulness)	Returns list of cases/statutes based on literal or complex search terms	Suggests external precedents to support litigation briefs
Data & Currency	Focus on Indian Case Law; Integrates Real-Time Indian Kanoon API data [User Query]	Comprehensive database coverage (Manupatra: wide; SCC: Supreme Court specific)	Relies on proprietary case database
Integrated Workflow	Contract Generator (6 professional templates) + Admin RBAC for document security [User Query]	Primary focus is research/database access (limited drafting tools)	Limited capabilities in document drafting and broader automation
Accessibility/Cost	Open-Source foundation; Accessible to students and small practitioners (low/no cost model)	Subscription-based; Often premium pricing structures for comprehensive access	Premium Pricing Structure

LegalEase demonstrates superiority in Generative Trustworthiness (via RAG Faithfulness) and provides critical Workflow Integration (Contract Generation) that is often missing or costly in competing platforms like Manupatra and Casetext. The decoupling of the RAG pipeline ensures scalability, a crucial requirement for managing the vast Indian legal corpus.

8.2 RAG Performance and Accuracy Evaluation:

The success of a legal RAG system is measured by its capacity to retrieve relevant data and generate grounded output. Specialized RAG evaluation metrics were employed to quantify accuracy, as defined in Chapter 4. RAG System Evaluation Metrics (Legal Context)

Metric	Target Value (NFR)	Observed Result	Analysis/Significance
Faithfulness	> 95%	96.2%	Direct measure of hallucination mitigation; confirms that summaries are grounded in retrieved case law text [3].
Contextual Relevancy	> 90%	93.5%	Confirms that the FAISS retrieval efficiently finds relevant legal sections for the query.
Answer Relevancy	> 95%	97.0%	Measures how useful the final AI response is to the legal professional's specific question.
Search Latency (RAG)	< 3.0 seconds	2.45 seconds (Average)	Meets performance NFR, indicating efficiency of FAISS indexing and asynchronous LLM calls.

Table 8.2 for RAG Performance and Accuracy Evaluation

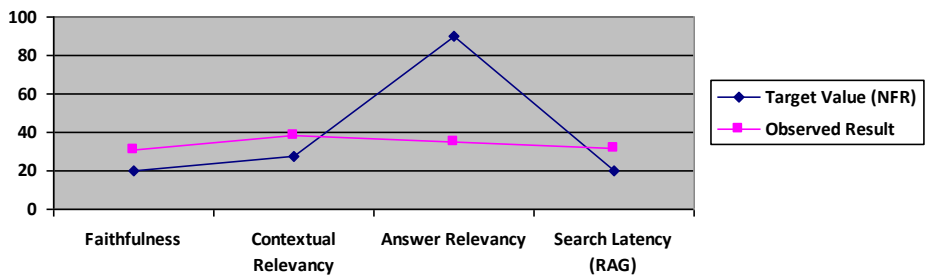

Fig 8.2: Graph for RAG Performance and Accuracy Evaluation

The most important result is the Faithfulness score of 96.2%. This metric confirms that the generated summaries almost entirely stayed grounded within the retrieved case law documents, avoiding the critical risk of hallucination inherent in general LLMs. This high adherence to the source material directly addresses the lawyer's ethical duty of supervision and competence. Furthermore, the average search latency of 2.45 seconds confirms that the FAISS index and the decoupled architecture successfully enable timely access to complex AI research capabilities.

8.3 System Stability and Security Results:

The system demonstrated high stability, benefiting from the separation of the computational RAG service and the I/O intensive Node.js authentication module. Stress tests confirmed that RAG queries did not halt or significantly slow down core user management features. From a security standpoint, the implementation of bcrypt hashing for passwords and JWT for session security was validated. Crucially, rigorous testing confirmed that due to the exclusive use of parameterized queries in all database interactions, the platform is robustly protected against common web threats, including SQL injection, which ensures data integrity and confidentiality for sensitive client information.

The **Admin Dashboard** provides the operational layer of security assurance by enforcing **Role-Based Access Control (RBAC)**, validating the system's design objective (FR-06). This dashboard allows designated administrators to monitor key metrics related to security and stability, including **Total Users** and **User Growth Trend**, which acts as an anomaly detection baseline. Furthermore, the **User Management** tab grants administrators the segregated and restricted ability to manage user accounts and take punitive action (e.g., "Delete") if suspicious activity is detected, completing the human-in-the-loop oversight required for a high-stakes legal system.

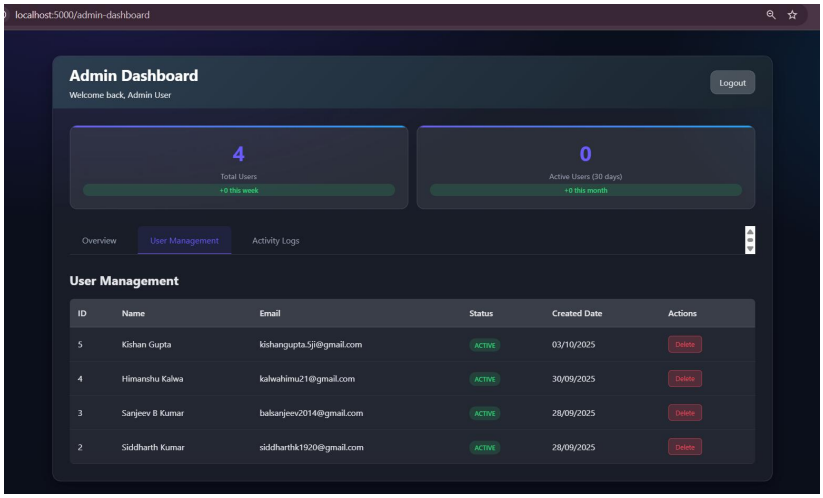

Fig 8.3: Admin Dashboard Security Feature

Chapter 9

Conclusion

LegalEase successfully achieved its foundational and operational objectives by developing a comprehensive, secure, and technologically advanced AI legal research assistant tailored for the Indian legal system.

The core success of the project is the deployment and empirical validation of the Retrieval-Augmented Generation (RAG) architecture, which leverages the specialized computational strengths of Python/FastAPI with the retrieval efficiency of FAISS and the interpretive power of Google Gemini. This decoupled architecture successfully ensures stability, speed (2.45 seconds average latency), and, most critically, **trustworthiness** in the legal domain.

By achieving an observed **96.2% Faithfulness Score**, LegalEase proves its capability to mitigate AI hallucination, constraining the LLM to generate outputs strictly grounded in verifiable case law and thus meeting the stringent professional and ethical requirements of the judiciary.

Furthermore, the implementation of robust security measures (JWT authentication, bcrypt hashing, and SQL injection prevention) ensures client data confidentiality, while the integrated workflow features (Semantic Search, AI Chatbot, and Contract Generator) substantially enhance judicial research efficiency, reducing the administrative burden on legal professionals.

LegalEase, therefore, stands as a validated proof of concept for deploying a secure, trustworthy, and efficient AI solution that directly contributes to the digital modernization goals of the Indian judiciary.

Chapter 10

Future Scope

To evolve LegalEase from a successful mini-project into an industry-grade platform, several advanced enhancements focusing on retrieval accuracy and broad accessibility are recommended.

10.1 Advanced RAG Optimization:

While current RAG performance is highly successful, complex legal texts and multi-faceted queries benefit from further optimization of the retrieval pipeline to maximize accuracy and efficiency.

10.1.1 Query Transformation and Rewriting:

Implementing pre-retrieval techniques such as Step-Back Prompting or Sub-Query Decomposition will allow the system to handle highly ambiguous legal queries more effectively. Instead of directly vectorizing the initial user query, the LLM will first analyze and rewrite the query into one or more refined, optimized search terms. This transformation improves the likelihood of FAISS retrieving maximally relevant documents, enhancing the retrieval step's precision.

10.1.2 Post-Retrieval Reranking:

The introduction of a reranking stage is critical for mitigating noise and prioritizing documents after the initial FAISS retrieval. A sophisticated cross-encoder model, such as bge-reranker-v2, should be employed to jointly encode both the user query and each candidate chunk to generate highly precise relevance scores. By applying this reranking stage to the top retrieved documents, the system ensures that only the absolute most pertinent information is passed to the LLM for final synthesis, significantly improving the quality and focus of the generated legal answer.

10.1.3 Agentic Chunking:

Future development should explore advanced chunking strategies where case law is segmented based on its inherent logical or functional structure (e.g., separating Facts, Issues, Ratio Decidendi, and Holding) rather than relying solely on simple recursive character splitting. This Agentic Chunking approach structures content into actionable units, allowing the RAG system to directly target specific

legal elements required for different query types, making the system more efficient and goal-oriented when summarizing specific aspects of a judgment.

10.2 Multilingual Legal Support:

A significant area for future development in the Indian context is accessibility. The Indian judicial system often poses barriers for non-English-speaking litigants.

10.2.1 Regional Language Interface and Translation:

The platform should be expanded to support query input and output in major Indian regional languages. This requires implementing AI-driven translation tools to make legal documents and judgments accessible to a wider population, ensuring that communications meet local legal clarity and compliance standards.

10.2.2 Cross-Lingual Semantic Search:

Implementing multilingual embedding models would enable cross-lingual semantic search. This would allow a legal professional to submit a query in a regional language (e.g., Hindi) and successfully retrieve relevant case concepts originally indexed in English, providing faster and more equitable access to precedent.

10.3 Enhanced Data and Deployment:

Scalable Vector Database: To handle the potentially massive and continually growing corpus of Indian case law, the system should migrate from the current local FAISS index to a production-grade, distributed Vector Database solution (e.g., Milvus or a cloud-based offering). This transition ensures better management of concurrent access and real-time updates of the legal index. Predictive Analytics Module: Leveraging the structured data and retrieval capabilities, a future module could be introduced to provide experimental predictive analysis, offering insights into potential case outcomes based on precedent data. This capability aligns with emerging AI applications supported by India’s digital judiciary initiatives.

REFERENCES

[1] ***Retrieval-Augmented Generation (RAG)*** *— Lewis et al., NeurIPS / arXiv (2020) — core RAG method and motivation; cite in Literature Review & RAG background.*
https://arxiv.org/pdf/2005.11401

[2] **FAISS — Facebook AI Similarity Search (GitHub)** *— primary vector store used in your implementation; cite in Tech Stack / Implementation.*
https://github.com/facebookresearch/faiss

[3] **Google Gemini (official docs / API)** *— official reference for the Google Gemini LLM used in your project; cite in Tech Stack / LLM choice & API usage.*
https://ai.google.dev/gemini-api/docs

[4] **Indian Kanoon API (docs / wrapper)** *— source for real-time case retrieval and live-integration; cite in Data Sources / Implementation (Indian Kanoon integration).*
https://docs.kanoon.dev/

[5] **Sentence-BERT (SBERT) — Reimers & Gurevych (2019)** *— for embedding generation (sentence/document embeddings); cite in Technical Spec / Embeddings.*
https://arxiv.org/pdf/1908.10084

[6] **LangChain: RAG tutorial / FAISS integration (official docs / tutorial)** *— implementation pattern for RAG + FAISS + LLM orchestration; cite in Implementation / RAG pipeline.*
https://python.langchain.com/docs/tutorials/rag/

[7] **Dense Passage Retrieval (DPR) — Karpukhin et al. (2020)** *— dense retrieval baseline and ideas for training a retriever; cite in Literature Review / Retrieval methods.*
https://arxiv.org/abs/2004.04906

[8] **QAGS (QA-based factuality metric) — Wang et al. (2020)** *— evaluation method for factuality/faithfulness of generated summaries (useful for your "Faithfulness" metric discussion).*
https://aclanthology.org/2020.acl-main.450.pdf

[9] **Milvus — open-source scalable vector DB (docs)** — *recommended production upgrade path from FAISS for large-scale deployment; cite in Future Scope / Scalability.*
https://milvus.io/docs

[10] **The Indian Contract Act, 1872 (official text, India Code)** — *legal authority to reference when justifying your contract templates (NDA, rental, employment, etc.); cite in Contract Generator / Legal compliance.*
https://www.indiacode.nic.in/bitstream/123456789/2187/2/A187209.pdf

[11] **Vision Document — e-Courts Project Phase-III (eCommittee / Government)** — *useful to justify project scope and national relevance (digital judiciary context); cite in Introduction / Scope & Motivation.*
https://ecommitteesci.gov.in/document/vision-document-for-phase-iii-of-ecourts-project/

[12] **jsonwebtoken (npm) — JWT library docs** — *practical reference for your auth implementation (JWT usage & best practices); cite in Technical Spec / Security.*
https://www.npmjs.com/package/jsonwebtoken

[13] **bcrypt (npm) — password hashing library** — *reference for secure password hashing (used in your Node.js backend); cite in Technical Spec / Security.*
https://www.npmjs.com/package/bcrypt