AI LEGAL BOT

Thesis Submitted in
Partial fulfillment for the award of

Bachelor of Computer Application

Submitted by :

1) **Rathod Kishan** : 2204030102471
2) **Bhoi Keyur** : 2204030101651
3) **Parmar Priya** : 2204030102182
4) **Shah Deep** : 2204030102535
5) **Chandarana Drashti :** 2204030101667

Department of Computer Application

Under the supervision of

Mr. Khush Jain

Silver Oak University

Gota, Ahmedabad-38248,
Gujarat, India

APRIL 2025

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

CERTIFICATE

It is certified that the project titled **AI LEGAL BOT** has been carried in groups of bonafide students of BCA VI semester, Department of Computer Application, Silver Oak University, Ahmedabad, Gujarat. The project is a record of the work accomplished during the sixth semester of BCA as a partial fulfillment of the requirement for the award of degree in Bachelor of Computer Applications (BCA), under my guidance.

Name of students	**ER. of student**
Rathod Kishan	2204030102471
Bhoi Keyur	2204030101651
Parmar Priya	2204030102182
Shah Deep	2204030102535
Chandarana Dhrashti	2204030101667___

Mr. Khush Jain, Lecturer

Department of Computer Application, Silver Oak University,

Ahmedabad, Gujarat.

DECLARATION

We hereby declare that project entitled **"AI LEGAL BOT"** in partial fulfillment for the degree of Bachelor of Computer Application to Silver Oak University, Ahmedabad, is bonafide record of the project work carried out at **SILVER OAK COLLEGE OF COMPUTER APPLICATION** under the supervision of **Mr. Khush Jain** and that no part of any of these reports has been directly copied from any student's report or taken from any other sources, without providing due reference.

Name of students

Rathod Kishan

Bhoi Keyur

Parmar Priya

Shah Deep

Chandarana Dhrashti

ACKNOWLEDGEMENT

We would like to express our profound sense of deepest gratitude to our guide **Mr. Khush Jain**, Silver Oak College of Computer Application (SOCCA), Ahmedabad for valuable guidance, sympathy and co-operation for providing necessary facilities and sources during the entire period of this project.

We wish to convey our sincere gratitude to all the faculties of the Department of Computer Application who have enlightened us during our studies. The facilities and cooperation received from the technical staff of the department is thankfully acknowledged.

The main credit for the successful completion of this project can be given to Principal of our college Dr. Hemal Patel who supported us throughout this enterprise.

We express our thanks to all those who helped us in one way or another.

Last, but not least, we would like to thank the authors of various research articles and books that were referred to.

Name of students	**ER. of student**
Rathod Kishan	2204030102471
Bhoi Keyur	2204030101651
Parmar Priya	2204030102182
Shah Deep	2204030102535
Chandarana Dhrashti	2204030101667

ABSTRACT

The Legal Bot is a web-based application designed to provide users with clear, context-aware legal information through an interactive chat interface. Built on a Flask backend and powered by Google Gemini's language understanding, the system retrieves and organizes relevant legal knowledge from a MongoDB database in real time. Each user session maintains its own conversation history, allowing the bot to deliver personalized responses and follow-up clarifications. The interface features a dynamic, typewriter-style message display and a sidebar for browsing past interactions.

By combining advanced natural-language processing with a user-friendly web design, the Legal Bot aims to support non-expert users in navigating common legal questions, improving access to legal guidance without requiring specialized background knowledge. Future work will focus on expanding the bot's coverage of legal topics and integrating real-time updates to reflect changes in legislation.

LIST OF FIGURES

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

TABLE OF CONTENTS

Section **Page no**

[1] INTRODUCTION

In an era where technology is reshaping every industry, the legal sector is no exception. Many individuals face challenges in accessing legal advice due to high costs, complexity of legal language, and lack of availability. To address these issues, this project introduces **Legal Bot**, a web-based chatbot that provides instant legal assistance for general queries.

Legal Bot leverages Artificial Intelligence (AI) and Natural Language Processing (NLP) to understand user input and deliver accurate, context-aware responses. It acts as a virtual legal assistant, helping users navigate through common legal topics such as consumer rights, contracts, cyber laws, and more. The system is designed with user-friendliness in mind, offering an intuitive interface that supports real-time conversations.

This chatbot is built using technologies such as **Flask** for the backend, **MongoDB** for data storage, and **Google Gemini** for generating intelligent responses. The aim of Legal Bot is not to replace legal professionals but to assist users in understanding their rights and available options before seeking expert legal help.

Overall, Legal Bot serves as an accessible, affordable, and efficient solution to basic legal inquiries, empowering individuals with knowledge and promoting legal awareness.

[1.1] Project Profile

Project Title: Legal Bot
Platform: Web-based Application
Technologies Used: Flask (Python), MongoDB, Google Gemini API, HTML, CSS, JavaScript
Project Domain: Legal Tech / AI Chatbot
Objective: To create a chatbot that provides instant and accurate legal information to users for general legal queries.

[1.2] Overview

The legal system can be complex, expensive, and inaccessible to many individuals. With advancements in artificial intelligence, it is now possible to bridge this gap using technology. **Legal Bot** is a smart legal assistant developed as a web application that provides users with reliable and immediate responses to legal queries.

The chatbot makes use of Natural Language Processing (NLP) and AI models to simulate human-like interaction and provide information on various legal topics including consumer protection, cyber law, property rights, and more. his project does not aim to replace legal experts but to assist users in gaining initial legal awareness and direction. With features like real-time messaging, session-based history, and multilingual support, Legal Bot offers a smooth and interactive user experience.

[1.3] Motivation

In today's world, legal awareness is essential, yet many people remain unaware of their rights and the laws that protect them. High consultation fees, fear of legal jargon, and unavailability of legal professionals in rural areas are significant barriers. The motivation behind this project stems from the need to develop a system that can offer **affordable, quick, and easy access** to legal information for everyone, regardless of their background or location.

Additionally, the rise of AI-powered assistants inspired the integration of a conversational interface that could guide users without the need for human intervention. The project also reflects an academic goal of applying machine learning and web technologies to solve real-world social issues.

[1.4] Goal

The main goal of the **Legal Bot** project is to develop an intelligent, accessible, and efficient legal chatbot system that:

- Provides quick and accurate answers to basic legal questions.
- Reduces dependency on in-person consultations for initial guidance.
- Offers a user-friendly and responsive web interface.
- Ensures data privacy and session-based chat handling.

[1.5] Organization of the Report/Thesis

This report is organized into multiple chapters, each focusing on a specific aspect of the project:

- **Chapter 1: Introduction**
  Provides an overview of the project, motivation, objectives, and the overall structure of the report.

- **Chapter 2: Literature Survey**
  Reviews existing legal bots, AI systems in the legal field, and related technologies.

- **Chapter 3: System Analysis and Design**
  Discusses system requirements, architecture, flow diagrams, and design models.

- **Chapter 4: Implementation**
  Details the development process, technologies used, and code structure.

- **Chapter 5: Testing and Evaluation**
  Covers testing methodologies, test cases, results, and performance analysis.

- **Chapter 6: Conclusion and Future Work**
  Summarizes the achievements, limitations, and potential future enhancements of the project.

[2] INITIAL SYSTEM STUDY

This section explains the background of the system and the reason why it was developed. It helps to understand the current situation, what problems exist, and how this project aims to solve them. It also highlights the purpose and the overall direction of the Legal Bot system.

[2.1] About Organization

This project was developed as part of the academic requirements under the guidance of Prof. Khush Jain (Silver Oak College of Computer Application) . The main purpose of the organization is to encourage students to create meaningful projects that solve real-life problems using modern technologies. The organization supports innovation, teamwork, and the practical use of tools like Artificial Intelligence (AI), web development, and database management systems. Legal Bot is one such initiative that applies these technologies to help people with legal information in a simple and easy way.

[2.2] Drawbacks of the Existing System

Currently, people mostly depend on lawyers or legal experts for any kind of legal help. While this is helpful, there are many problems in this system:

- **High cost**: Legal consultations can be very expensive for normal people.
- **Limited access**: In rural or remote areas, legal services are not easily available.
- **Complicated language**: Legal documents are written in difficult terms that are hard to understand.
- **Time-consuming:** Getting legal advice takes time, as appointments and delays are common

- **Lack of awareness**: Many people do not even know what their rights are or where to start. Because of these problems, many people stay confused or unaware about legal matters. This shows the need for a better, smarter system that can give help quickly and easily.

[2.3] Problem Definition

People often need legal help or advice but are not able to get it on time. Some are afraid to approach lawyers, while others cannot afford the cost or do not understand the legal language

There is no simple platform where a person can just ask a question and get a reliable legal answer.

The problem is clear – **there is no easy, accessible, and affordable way for common people to get basic legal guidance** without depending on experts or spending a lot of money.

[2.4] The Proposed System

To solve the problems mentioned above, the proposed solution is **Legal Bot** – a chatbot that gives legal information through a simple and interactive web platform. The user can type their legal question, and the chatbot will reply with a clear and helpful answer.

The system includes

- A clean and simple chat interface
- Real-time conversation like a human assistant
- AI-powered replies using Google Gemini
- Session-based storage of chats using MongoDB
- Categorized legal topics for easy reference

This chatbot is available anytime and does not require the user to register or pay. It is made for everyone—whether a student, employee, or general user—who needs quick legal help.

[2.5] Scope of the System

The system mainly focuses on giving legal help in the following areas:

- Consumer rights and complaints
- Cybercrime and online safety laws
- Employment rights and work-related issues
- Civil duties and general legal awareness

It is suitable for students, professionals, and the general public. However, it does not replace actual legal services but works as a basic guide to help users understand their problems and prepare them to seek further legal help if needed.

[2.6] Scope of this Project

This project is a **working prototype**, built to show how a legal chatbot system can function. The current version focuses on:

- Creating a responsive and interactive web interface
- Using Flask for backend logic and chatbot communication
- Using MongoDB for storing chat history with each session
- Integrating the Gemini AI model to provide intelligent and accurate responses
- Giving responses in multiple langauges

In the future, this chatbot can be improved by adding more legal topics, voice interaction, support for different languages, and links to legal documents or forms.

[2.7] System Development Approach

The system was developed using a **step-by-step process** that made it easy to plan, build, and test the chatbot properly. The following approach was followed:

1. **Requirement Gathering** – Understanding what users need from a legal chatbot.
2. **Planning and Design** – Designing the layout, chatbot flow, and selecting technologies.
3. **Development** – Writing code for frontend (chat screen), backend (Flask server), and connecting Gemini and MongoDB.
4. **Testing** – Checking if the chatbot gives correct responses and works smoothly.
5. **Deployment and Feedback** – Hosting the chatbot on a web platform and collecting user opinions for improvement.

This approach helped to complete the project in a systematic way and allowed changes or improvements during each step.

[3] FEASIBILITY

Feasibility analysis is an important step before starting any project. It helps us to decide whether the project is worth doing or not. It checks if the system is possible to build, how much it will cost, how it will work, and whether people will use it. In this project, we checked different types of feasibility to make sure that the Legal Bot system can be developed and used successfully.

[3.1] Technical Feasibility

This checks if the system can be developed using the current technology and tools.

The Legal Bot is technically feasible because:

- The technologies used (Flask, MongoDB, HTML, CSS, JavaScript) are well-known and easy to implement.
- Google Gemini provides powerful AI capabilities for generating smart responses.
- The chatbot can run on basic hosting servers without requiring heavy resources.
- Developers have enough knowledge of the tools and programming languages required.

So, from the technical side, this project is completely feasible.

[3.2] Operational Feasibility

This checks if the system will work well in real use and be accepted by the users.

The Legal Bot system is operationally feasible because:

- It is easy to use with a simple interface.
- It gives instant replies, which saves users time.
- It helps people understand legal terms and rights in simple language.

So, the system is likely to be accepted and used by many users who need quick legal help.

[3.3] Economic Feasibility

This checks if the system can be developed and maintained within a reasonable cost.

The Legal Bot is economically feasible because:

- All development tools and frameworks used are open-source or free.
- There is no need for expensive servers or software.
- The chatbot helps reduce the cost of legal consultations for users.
- Maintenance costs are low because it is a web-based system.

So, the system is cost-effective and affordable for both developers and users.

[3.4] Legal Feasibility

This checks if there are any legal problems with developing or using the system.

The Legal Bot does not offer actual legal advice or replace lawyers. It only provides **basic legal information** based on public laws and commonly known rights. It does not collect personal data or make legal decisions. Therefore, the system does not break any legal rules and is legally feasible.

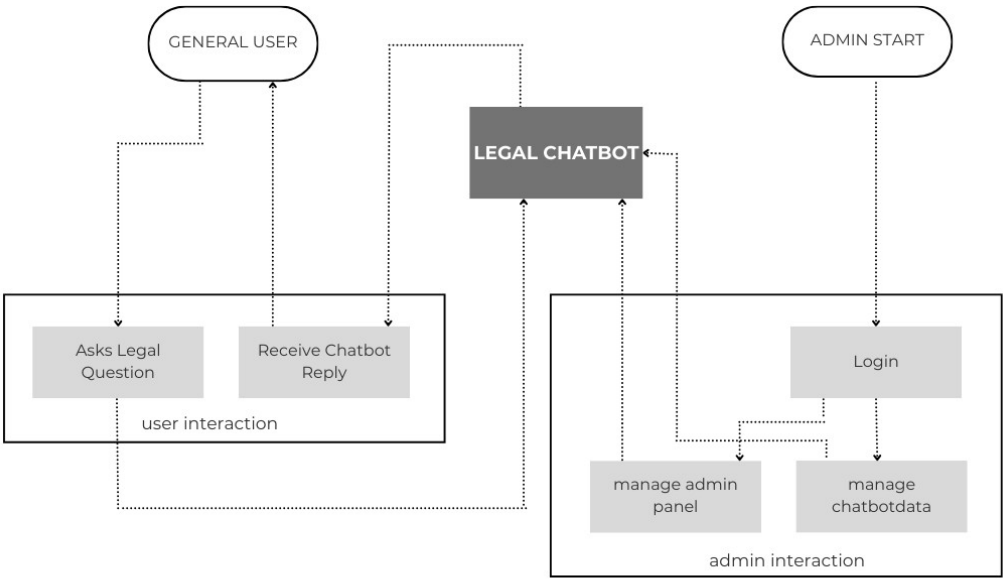
[4] SYSTEM SYSTEM ANALYSIS

This section explains the structure and working of the Legal Bot system, including both the **User Chat Interface** and the **Admin Page**. It covers how data flows, how different parts are connected, and how the system behaves when used. This gives a full picture of the entire system before and during development.

[4.1] Data Flow Diagram (DFD)

Level 0 DFD:

- Two main users: **General User** and **Admin**.
- User asks a legal question → System replies using chatbot.
- Admin logs in → Views user queries and manages chatbot data.

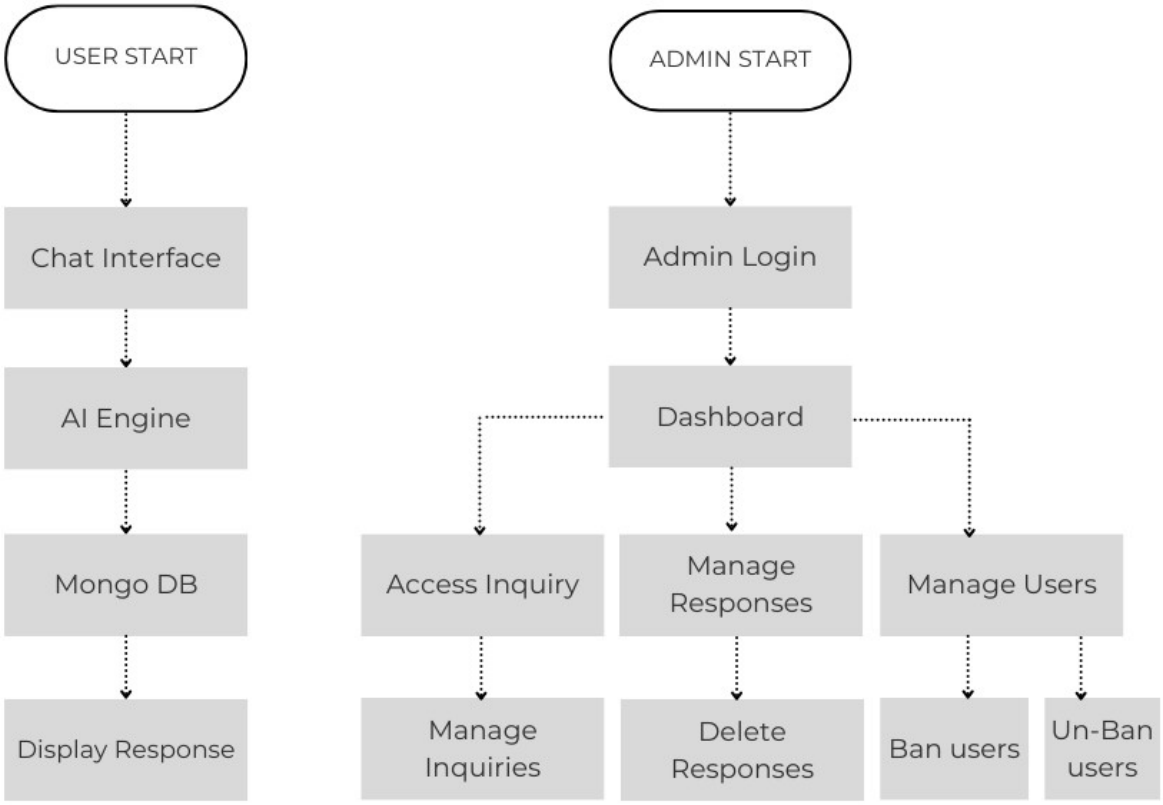


Level 1 DFD:

- **User Side**:
  - User → Chat Interface → AI Engine → MongoDB → Response displayed.

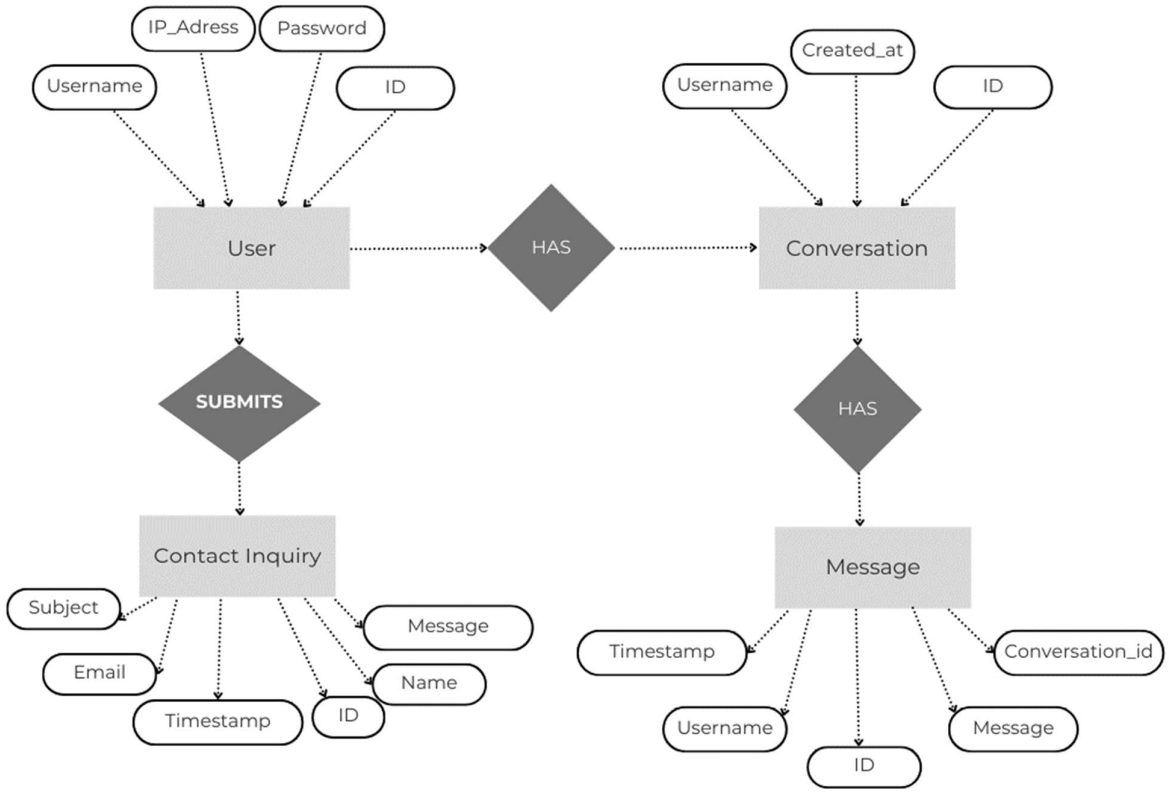
- **Admin Side**:
  - Admin Login → Dashboard → View chat logs → Update/Delete topic-related responses or manage chatbot behavior.

  This shows how data flows differently for users and admins



[4.2] Entity Relationship Diagram (ERD)

This Entity Relationship Diagram (ERD) models the backend structure of a chatbot-based system that handles user interactions, conversations, messages, and support inquiries. It outlines the relationships between key entities involved in the system and their attributes.



Entities and Attributes

1. **User**
   - Attributes: ID, Username, Password, IP_Address
   - Represents individuals who interact with the chatbot system.
   - A user can initiate conversations and submit contact inquiries.

2. **Conversation**
   - Attributes: ID, Username, Created_at
   - Represents a chat session initiated by a user.

- A conversation is linked to a single user.

3. **Message**
   - Attributes: ID, Message, Username, Timestamp, Conversation_id
   - Represents a single message within a conversation.
   - Each message belongs to one conversation and is sent by a user.

4. **Contact Inquiry**
   - Attributes: ID, Name, Email, Subject, Message, Timestamp
   - Captures support or help-related messages submitted by users.
   - Independent of chat conversations but still tied to the user submitting it.

Relationships

- **User HAS Conversation**
  A one-to-many relationship: one user can have multiple conversations.
- **Conversation HAS Message**
  A one-to-many relationship: each conversation consists of multiple messages.
- **User SUBMITS Contact Inquiry**
  A one-to-many relationship: one user may submit multiple contact inquiries.

[4.3] Physical and Behavioral Aspects of the System

4.4.1 Physical Aspects

- **Frontend (User)**: Chat interface built using HTML, CSS, JavaScript.
- **Frontend (Admin)**: Admin panel/dashboard built for login, viewing chat logs, managing legal topics.

- **Backend**: Flask handles logic for both chat and admin features.
- **Database**: MongoDB stores chat sessions, admin credentials, and legal categories.
- **AI**: Gemini API processes questions and generates smart replies.
- **Security**: Admin page includes secure login and session management.

4.4.2 Behavioral Aspects

- **User Behavior**:
  - Opens chatbot, types question, receives instant response.
  - System stores conversation for future reference.

- **Admin Behavior**:
  - Logs in securely to the admin page.
  - Views chat history from users.
  - Updates or deletes legal topics or bot-related data.
  - Monitors system performance and user feedback (if implemented).

Including an admin page makes the system more dynamic and easy to manage. It ensures the chatbot stays accurate, updated, and under control

[5] Software Requirements Specifications

[5.1] General Description

5.1.1 Product Perspective

The Legal Bot is a smart chatbot designed to provide users with legal information and support. It is a web-based system developed using Python's Flask framework for the backend and MongoDB for the database. The system is powered by the Gemini AI model, which helps generate human-like legal responses.

This chatbot works independently and does not depend on any other existing systems. However, in the future, it can be integrated with government legal websites, e-court systems, or legal databases to improve its functionality. It has a user-friendly front-end interface where users can chat naturally, and a separate admin panel for system monitoring and management.

5.1.2 Product Functions

The key functions of the Legal Bot system include:

- **User Query Input**: Allows users to ask legal questions through a chat interface.
- **AI Response Generation**: Sends the user's query to Gemini AI and receives a meaningful legal response.
- **Session-Based Chat Handling**: Maintains separate sessions for each user to manage context-based conversations.
- **Chat History Management**: Stores all messages in a MongoDB database for future reference.
- **Admin Access**: Admins can log in securely to view chat history and monitor user activity.

- **Data Handling**: Ensures that all communication and responses are properly stored and retrieved.

5.1.3 User Characteristics

There are two main types of users in the system:

- **End Users**:
  - General public or students seeking legal information.
  - Basic internet browsing and typing skills are enough.
  - No legal or technical background is required.

- **Admin Users**:
  - Usually system developers or authorized moderators.
  - Should know how to operate a secure login system.
  - Responsible for checking chat logs and managing misuse.

5.1.4 General Constraints

- The system requires an active internet connection to function, especially for AI responses.
- It must run on modern web browsers like Chrome, Firefox, or Edge.
- Performance depends on the speed and accuracy of the Gemini AI.
- The system should be hosted on a reliable server to ensure availability.
- Admin panel must be protected with secure credentials to prevent unauthorized access.

5.1.5 Assumptions and Dependencies

The following assumptions are made while developing and using the system:

- Users will ask their questions in simple and clear English.
- The Gemini AI API will remain active and deliver accurate responses.
- MongoDB is properly configured to handle all data operations.
- Flask server will be deployed correctly and remain active.

- Admin users will remember and use correct login credentials.
- There is no misuse or spam from users in the chat system.

5.1.6 Functional Requirements

These are the core requirements that the system must fulfill:

- **User Chat Functionality: Users must be able to type queries and get** responses.
- **AI Integration**: Each user query must be sent to Gemini AI, and a reply must be returned.
- **Database Storage**: All chat messages should be stored in MongoDB, linked to their session.
- **Session Management**: Each user session must be tracked for organized chat.
- **Admin Login**: Admin must be able to log in securely with username and password.
- **Admin Features**: Admin should be able to view chat history for all sessions.

5.1.7 Non-Functional Requirements

These requirements define how the system should behave:

- **Performance**: System should respond within 2–3 seconds for a good user experience.
- **Scalability**: It should be able to handle multiple users at the same time.
- **Security**: Admin login and data must be protected with strong authentication.
- **Reliability**: The system must be dependable and not crash unexpectedly.
- **Usability**: The user interface should be clean, responsive, and easy to use.
- **Maintainability**: Codebase and backend should be modular so updates can be made easily.
- **Backup**: Data should be backed up regularly to avoid loss.

[5.2] External Interface Requirements

5.2.1 User Interfaces

The Legal Bot system provides a simple and interactive web-based user interface:

- **Chat Interface**: A chat box where users can type their legal queries and view AI-generated replies.
- **Session View**: Users see their current conversation history for reference.
- **Admin Panel**: A secure login interface where the admin can enter credentials and view all chat records.
- **Design Features**: The interface is responsive, user-friendly, and cleanly designed using HTML, CSS, and JavaScript.

5.2.2 Hardware Interfaces

- The system is hosted on a server, and users can access it using a device such as:
  - Desktop
  - Laptop
  - Tablet
  - Smartphone
- There are no special hardware dependencies on the client-side; any device with an internet connection and a web browser can use the system.
- The backend server requires basic hardware configuration capable of running a Flask application with MongoDB.

5.2.3 Software Interfaces

- **Frontend Technologies**: HTML, CSS, JavaScript for user interaction.
- **Backend**: Flask (Python-based web framework) used for handling requests and managing the chatbot logic.

- **Database**: MongoDB used for storing session data and chat history.
- **AI Engine**: Integrated with Google Gemini API for generating responses.
- **Web Browser**: Chrome, Firefox, Edge, or any modern browser required to access the system.

[5.3] Performance Requirements

- The system should respond to user queries within 2–3 seconds.
- Must be able to handle at least 10–20 simultaneous users without significant delay.
- The database should store and retrieve data efficiently to support smooth conversation flow.
- The admin panel should load previous conversations quickly for monitoring.
- AI response time depends on the Gemini API but should be optimized for minimal delay.

[5.4] Design Constraints

5.4.1 Standard Compliance

- The system follows standard web development practices.
- Interface design follows basic UI/UX guidelines for accessibility.
- Data handling is done using secure and structured methods through the Flask API.

5.4.2 Hardware Constraints

- The performance of the backend depends on the server hardware.
- A minimum of 4 GB RAM and a modern processor is recommended for the server.
- Client-side devices should have sufficient RAM and processing power to run a modern browser smoothly.

5.4.3 Other Requirements

- The Gemini API key must be valid and active.
- MongoDB must be properly configured and accessible for data storage.
- Admins should keep credentials safe to prevent unauthorized access.
- Network connectivity must be stable to ensure real-time communication.

5.4.4 Scope of This Project

- The project is limited to answering general legal queries via chat.
- It does not offer personalized legal advice or replace a real lawyer.
- Admin functionalities are limited to login and viewing user messages.
- Future scope includes integrating authentication, live chat with legal experts, or language support.

[6] SYSTEM DESIGN

[6.1] External Interface Requirements

The system design phase defines the architecture, components, interfaces, and data flow of the chatbot application. This chatbot facilitates user interactions through natural language input,

enabling efficient message exchange, conversation tracking, and legal inquiry submissions. The design focuses on modularity, scalability, and maintainability using a NoSQL database (MongoDB) and RESTful API principles.

[6.2] System Architecture

The chatbot system follows a client-server architecture consisting of the following layers:

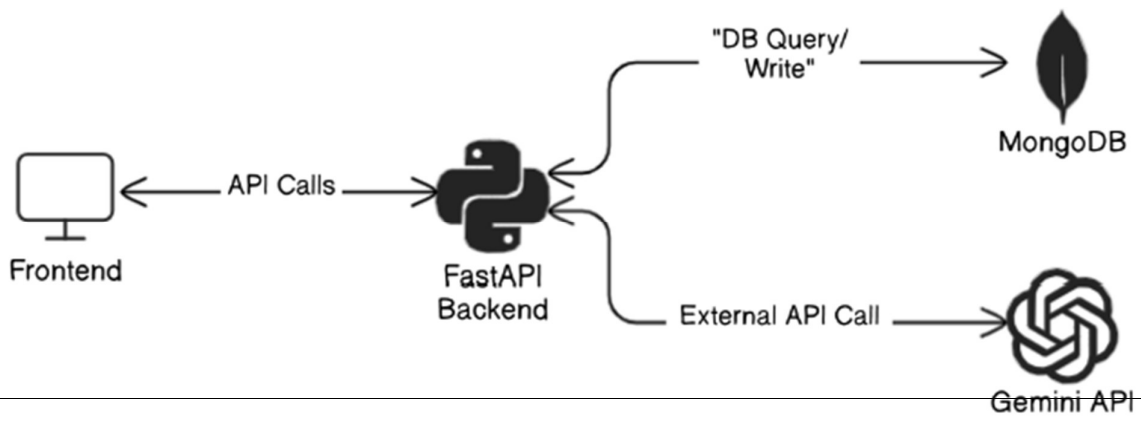
- **Frontend (Client Side):**
  - Provides a user-friendly interface for sending messages and submitting contact forms.
  - Communicates with the backend via HTTP requests.

- **Backend (Server Side):**
  - Built using FastAPI, handles user authentication, conversation management, and message processing.
  - Interfaces with the Gemini AI model for intelligent responses.

- **Database Layer:**
  - MongoDB stores user details, conversation logs, messages, and contact inquiries.

- **External Services:**

  - Integration with Gemini Pro API for chatbot functionality.



[6.3] Module Design

The system is divided into several key modules:

- **User Management Module:**
  - Handles user registration, login, and status tracking.
- **Conversation Module:**
  - Manages the creation and retrieval of conversations initiated by users.
- **Chat Message Module:**
  - Stores, retrieves, and timestamps all messages exchanged in a conversation.
- **Contact Inquiry Module:**
  - Allows users to submit support or legal requests with name, email, subject, and message.
- **Authentication Module:**
  - Uses JWT tokens for secure API access.

[6.4] Database Design

The MongoDB collections and their fields are as follows:

- **users:**
  - _id, username, password, status, ip_address, last_login
- **conversations:**
  - _id, username, created_at

- **messages:**
  - _id, message, username, timestamp, conversation_id
- **contact_inquiries:**
  - _id, name, email, subject, message, timestamp, status

The relationships among these collections are represented through an ER diagram to ensure data integrity and traceability.

[6.5] Input Output Design

Inputs:

- User login credentials (username, password)
- Chat message text
- Contact inquiry form data (name, email, subject, message)

Outputs:

- Chatbot responses generated via Gemini
- List of conversations and messages
- Confirmation of inquiry submission
- User authentication tokens

[6.6] Algorithm Design

Key logic/algorithm includes:

- **User Authentication:**
  - Password validation with hashed values
  - JWT generation upon successful login

- **Message Flow:**
  - Messages are received, stored, and passed to Gemini for AI processing
  - Response is returned and logged in the messages collection

- **Contact Submission:**
  - Data is validated and saved to contact_inquiries collection with a default status of unread

[6.7] Electronic Data Communication Design

- Data is exchanged via RESTful API over HTTPS.
- Authentication is handled using JWT in the Authorization header.
- MongoDB stores all interaction data securely.
- JSON is used as the data format between frontend and backend.

[6.8] System Maintenance

- **Logs**: System maintains logs for login, message exchanges, and submissions.
- **Scalability**: The modular design allows new features (e.g., admin panels or analytics) to be added without affecting core functionality.
- **Backup**: Regular backups of MongoDB ensure data persistence and recovery.

[6.9] Other Alternatives Considered

- **Relational Database (MySQL/PostgreSQL):**
  - Considered for structured data but rejected due to MongoDB's flexibility and faster iteration for document-based data.

- **Socket-based Communication:**

- Considered for real-time messaging but replaced with REST API for simplicity and maintainability in this phase.

[7] SYSTEM IMPLEMENTATION

[7.1] Hardware Components

To implement the Legal Bot system effectively, minimal hardware is required, as it's primarily a web-based application. Below are the necessary hardware components:

- **Client-Side Hardware (User/Admin):**
  - Desktop or laptop
  - Minimum 2 GHz processor
  - 4 GB RAM or more
  - Internet connection

- **Server-Side Hardware (Hosting/Development):**
  - Computer with at least i5 processor
  - 8 GB RAM or more
  - Stable internet connection
  - Hosting environment (can be local or cloud-based)

[7.2] Software Environment

The Legal Bot system uses modern and open-source tools for development. The software environment includes:

- **Frontend:**
  - HTML, CSS, JavaScript (Vanilla JS with fetch API)
  - Animation and UI design for chat interface

- **Backend:**
  - Python (Flask framework)
  - Google Gemini API for AI responses
  - PyMongo for MongoDB integration

- **Database:**
  - MongoDB (NoSQL database)

- **Development Tools:**
  - Visual Studio Code (IDE)
  - Postman (for API testing)
  - Browser developer tools (for debugging)

[7.3] System Development Platform

The system was developed and tested on:

- **Operating System:** Windows 10
- **Development Environment:** VS Code with integrated terminal
- **Browser:** Google Chrome / Mozilla Firefox / Microsoft Edge
- **Backend Server:** Flask development server (localhost)
- **Database Server:** MongoDB (local and optionally MongoDB Atlas)

This platform ensures smooth testing, debugging, and version control during the development cycle.

[7.4] Project Accomplishment Status

As of now, the project is in an advanced stage of development. Below are key accomplishments:

- Chat interface with smooth animation is complete.
- Flask backend integrated with Gemini API and MongoDB.
- Admin login and dashboard developed and functioning.
- Session-based conversation saving implemented.
- Functional testing completed for all key modules.
- Final deployment (optional cloud hosting) is the next step.

The project is functioning as expected, with all core features implemented.

[7.5] Guidelines for Continuation

To continue or expand the project, the following guidelines are recommended:

- Use a **cloud service** like Render or Railway for hosting the Flask app.
- Secure Gemini API keys using environment variables or .env files.
- Implement **authentication layers** (e.g., JWT) for advanced user login.
- Extend functionality to support **multiple bots** or **multi-language** chats.
- Backup MongoDB regularly using cloud tools or scheduled exports.
- Plan for **feedback and rating system** to improve bot accuracy.

[8] SYSTEM IMPLEMENTATION

[8.1] Test Plan

The purpose of testing is to ensure that the Legal Bot system works as expected, is free of major bugs, and provides a smooth user experience. The testing plan includes:

- **Unit Testing:**
  Each individual component or function (e.g., chatbot API request, admin login, session storage) is tested separately to verify it behaves correctly.

- **Integration Testing:**
  Checks the interaction between modules like frontend → backend → Gemini API → MongoDB to ensure smooth data flow.

- **System Testing:**
  Validates the complete system as a whole to confirm that it meets the specified requirements.

- **User Acceptance Testing (UAT):**
  Conducted to ensure that the chatbot behaves accurately and that the admin panel meets user expectations.

- **Security Testing:**
  Verifies that admin credentials are protected, API keys are hidden, and sessions are properly handled.

[8.2] Test Cases

Test ID	Test Case Description	Input	Expected Output	Status
TC01	Test chatbot message sending	"What is an FIR?"	AI gives a proper legal explanation	✓ Pass
TC02	Test empty input in chatbot	Empty message	Displays error or prevents submission	✓ Pass

TC03	Admin login with valid credentials	Valid username & password	Redirects to admin dashboard	✓ Pass
TC04	Admin login with invalid credentials	Wrong username/password	Shows error message	✓ Pass
TC05	Check if chat is saved in database	Chat session in progress	Stored correctly in MongoDB with session ID	✓ Pass
TC06	Test network failure during message send	Disconnect internet	Shows error message / retry option	✓ Pass
TC07	Check typewriter effect and chat animation	Chat input	Message appears with typing animation	✓ Pass
TC08	Multiple sessions from different browsers	Open in different tabs	Separate sessions maintained in MongoDB	✓ Pass
TC09	Admin view chat history	Admin clicks view chats	Displays chat data grouped by sessions	✓ Pass

Note: Testing was done in a **localhost** environment. Before live deployment, further testing in a hosted environment is recommended.

[9] CONCLUSION AND FUTURE DECISION WORK

[9.1] Conclusion

The **Legal Bot** project successfully demonstrates how AI and web technologies can be combined to create an interactive, accessible, and user-friendly legal assistant platform. The system is designed to help users get instant responses to general legal questions in a

conversational format. It features a smooth user interface, a responsive chatbot powered by Google Gemini, and an admin panel for managing sessions and conversations.

By using **Flask**, **JavaScript**, **MongoDB**, and the **Gemini API**, the project integrates both frontend and backend components effectively. All core functionalities—chat handling, session storage, admin access, and user interaction—were implemented and tested thoroughly.

Overall, this project meets its intended goals, provides a working prototype, and lays a solid foundation for future expansion.

[9.2] Future Direction of Work

Although the Legal Bot is functional and complete at the prototype level, there are several ways it can be improved or extended in the future:

- **Real Legal Data Integration:**
  Currently, the bot responds based on a language model (Gemini). Integrating it with an up-to-date legal database or government portal can make the answers more reliable.
- **Multi-Language Support:**
  Adding support for regional languages (e.g., Hindi, Gujarati, Tamil) will make the bot accessible to a wider audience.
- **User Authentication:**
  Allowing users to create accounts can help track their queries and provide personalized experiences.
- **Live Chat with Lawyers:**
  A feature where users can request to talk to a real lawyer through chat or video after the bot answers could be a great enhancement.
- **Mobile App Version:**
  Developing an Android or iOS version of Legal Bot would make it easier to access on the go.

- **Feedback System:**
  Implementing a simple feedback or rating system to evaluate how helpful each answer was can help improve response quality.
- **Chatbot Personality & Tone Customization:**
  Making the chatbot's tone more formal or casual depending on the user's choice can improve interaction quality.

[10.] Bibliography

Websites and Online Resources:

1. Flask Documentation – For backend development using Python Flask.
2. MongoDB Official Docs – For NoSQL database integration and usage.
3. Google Gemini API (Generative AI) – For implementing AI-based responses.
4. W3Schools – For frontend technologies and JavaScript references.
5. Stack Overflow – For debugging, community support, and best practices.

6. GitHub – For version control and collaboration.
7. Postman – For API testing and simulation.

Tools and Software Used:

- **Visual Studio Code (VS Code)** – IDE used for development.
- **MongoDB Compass** – GUI for managing and viewing MongoDB collections.
- **Google Chrome** – Browser used for testing the web application.
- **Git & GitHub** – Version control and code hosting.