

# **SRM CHATBOT**

**A PROJECT REPORT SUBMITTED TO**

**SRM INSTITUTE OF SCIENCE & TECHNOLOGY**

**IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE**

**AWARD OF THE DEGREE OF**

**MASTER OF COMPUTER APPLICATIONS**

**IN**

**GENERATIVE ARTIFICIAL INTELLIGENCE**

**BY**

**KAARTHIK NARAIN (REG NO.RA2432242010095)**

**GANESH GURU SURYAVEL (REG NO. RA2432242010101)**

**JAGATH ASWIN B (REG NO. RA2432242010127)**

**VIJAY D (REG NO. RA2432242010099)**

**UNDER THE GUIDANCE OF**

**Dr. S.SIVAKUMAR M.C.A., M.Phil., M.S., PhD**



**DEPARTMENT OF COMPUTER APPLICATIONS**

**FACULTY OF SCIENCE AND HUMANITIES**

**SRM INSTITUTE OF SCIENCE & TECHNOLOGY**

**KATTANKULATHUR, CHENNAI 603 203**

**OCTOBER - 2025**

## **BONAFIDE CERTIFICATE**

This is to certify that the project report titled “**SRM CHATBOT**” is a bonafide work carried out by **KAARTHIK NARAIN** (RA2432242010095), **GANESH GURU SURYAVEL** (RA2432242010101), **JAGATH ASWIN B** (RA2432242010127), **VIJAY D** (RA2432242010099) under my supervision for the award of the Degree of Master of Computer Applications in Generative Artificial Intelligence. To my knowledge the work reported herein is the original work done by these students.

**Dr. S.Sivakumar**

Assistant Professor,

Department of Computer Applications

(GUIDE)

**Dr.P.J.Arul Leena Rose**

Professor & Head,

Department of Computer Applications

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

With profound gratitude to the ALMIGHTY, I take this chance to thank the people who helped me to complete this project.

We take this as a right opportunity to say THANKS to my parents who are there to stand with me always with the words “YOU CAN”.

We are thankful to **Dr.T.R.Paarivendhar**, Chancellor, and **Prof.A.Vinay Kumar**, Pro Vice-Chancellor (SBL), SRM Institute of Science & Technology who gave us the platform to establish me to reach greater heights.

We earnestly thank **Dr.A.Duraisamy**, Dean, Faculty of Science and Humanities, SRM Institute of Science & Technology who always encourages us to do novel things.

A great note of gratitude to **Dr. S. Albert Antony Raj**, Deputy Dean, Faculty of Science and Humanities for his valuable guidance and constant Support to do this Project.

We express our sincere thanks to **Dr. P.J.Arul Leena Rose**, Professor & Head for her support to execute all inclines in learning.

It is our delight to thank our project guide **Dr. S.Sivakumar**, Assistant Professor, Department of Computer Applications for his help, support, encouragement, suggestions, and guidance throughout the development phases of the project.

We convey our gratitude to all the faculty members of the department who extended their support through valuable comments and suggestions during the reviews.

Our gratitude to friends and people who are known and unknown to me who helped in carrying out this project work a successful one.

**KAARTHIK NARAIN S**  
**GANESH GURU SURYAVEL**  
**JAGATH ASWIN B**  
**VIJAY D**

## TABLE OF CONTENTS

<b>1. INTRODUCTION.....</b>	<b>1</b>
<b>2. SOFTWARE REQUIREMENT ANALYSIS.....</b>	<b>2</b>
2.1 HARDWARE SPECIFICATION.....	2
2.2 SOFTWARE SPECIFICATION.....	2
2.3 ABOUT THE SOFTWARE AND ITS FEATURE.....	3
<b>3. SYSTEM ANALYSIS.....</b>	<b>5</b>
3.1 EXISTING SYSTEM.....	5
3.2 PROPOSED SYSTEM.....	5
3.3 FEASIBILITY STUDY.....	5
<b>4. SYSTEM DESIGN.....</b>	<b>7</b>
4.1 DATA FLOW DIAGRAM.....	7
4.2 UML DIAGRAMS.....	9
<b>5. CODE TEMPLATES.....</b>	<b>11</b>
5.1 MODULE DESCRIPTION.....	11
5.2 TABLES.....	13
<b>6. TESTING.....</b>	<b>14</b>
6.1 TESTING METHODOLOGIES.....	14
6.2 TEST CASE.....	15
<b>7. OUTPUT SCREENS.....</b>	<b>18</b>
<b>8. CONCLUSION.....</b>	<b>22</b>
<b>9. FURTHER ENHANCEMENTS.....</b>	<b>23</b>

<b>10.</b>	<b>REFERENCES.....</b>	<b>24</b>
<b>11.</b>	<b>APPENDICES.....</b>	<b>25</b>
11.1	USER DOCUMENTATION.....	25
11.2	README.....	26
11.3	SAMPLE SOURCE PROGRAM.....	27

## **ABSTRACT**

The SRM Chatbot is an AI-driven web-based virtual assistant designed to provide students, faculty, and staff at SRM Institute of Science and Technology (SRMIST) with instant and accurate information. In educational institutions, students often require quick access to exam schedules, timetables, seat allocations, course details, and campus announcements. Traditional methods like emails, notice boards, or manual inquiry counters can be slow, inefficient, and unable to handle multiple queries simultaneously. The SRM Chatbot addresses these challenges by offering a real-time, interactive platform that ensures timely and accurate responses.

Built using Python for the backend and Dialogflow for natural language processing, the chatbot interprets queries in conversational language, recognizes abbreviations, and handles minor errors. The frontend is developed with HTML, CSS, and JavaScript, providing a responsive and user-friendly interface. Information such as timetables, seat data, and announcements is stored in a SQLite database, allowing dynamic retrieval. Features like fuzzy matching, acronym expansion, and gibberish detection enhance accuracy and relevance.

The chatbot provides personalized support, including exam schedules, seat allocations by register number and session, and real-time notifications. It also reduces administrative workload by automating repetitive queries and operates 24/7, ensuring continuous access to information. By combining AI-based retrieval, natural language understanding, and web technologies, the SRM Chatbot improves operational efficiency, enhances student engagement, and streamlines campus communication, demonstrating a practical application of AI in higher education.

## LIST OF TABLES

S.No	Name	Table. No	Page no
1	Seat Data	5.1.1	13
2	Timetable Data	5.1.2	13

## LIST OF FIGURES

S.No	Name	Fig. No	Page no
1	Data Flow Diagram	4.1.1	7
2	Query Processing Flow	4.1.2	8
3	Use case Diagram	4.2.1	9
4	Sequence Diagram	4.2.2	10
5	Chatbot UI	7.1	18
6	Terms and condition	7.2	18
7	Admin Access Page	7.3	19
8	Meet the Team	7.4	19
9	Admin Upload Module	7.5	20
10	Admin News Management Module	7.6	21



# 1. INTRODUCTION

In modern educational institutions, students and staff require quick and accurate access to information such as exam timetables, seat allocations, course details, and campus announcements. Traditional methods like notice boards, emails, and manual inquiry counters are often slow, inefficient, and unable to handle multiple simultaneous queries. To address these challenges, the SRM Chatbot has been developed as an AI-powered virtual assistant for SRM Institute of Science and Technology (SRMIST).

Unlike conventional chatbots, this system leverages Retrieval-Augmented Generation (RAG) with semantic search over structured and unstructured campus data. Key information, including timetables, seat details, and announcements, is stored in a SQLite database or embedded as vectors using SentenceTransformers. A FAISS index enables high-speed similarity search, retrieving relevant documents or entries in response to user queries. The retrieved context is then processed by an LLM to generate coherent, accurate, and context-aware responses, ensuring users receive personalized answers even for queries not explicitly pre-programmed.

Advanced features like fuzzy matching, acronym expansion, and gibberish detection improve query understanding and accuracy. The chatbot can interpret natural language, handle abbreviations, and identify department-specific requests. Its web interface, built with HTML, CSS, and JavaScript, provides a responsive and interactive user experience, while the backend orchestrates retrieval, generation, and database management.

The SRM Chatbot demonstrates the practical application of AI and RAG in higher education, automating routine queries, enhancing operational efficiency, and delivering real-time, personalized assistance to students and staff, improving overall campus communication.

# 1. SOFTWARE REQUIREMENT ANALYSIS

## 2.1 HARDWARE SPECIFICATION

- **Processor:** Intel Core i5 or higher (8th Gen or above)
- **RAM:** Minimum 8 GB (16 GB recommended for AI model inference)
- **Storage:** 500 GB HDD/SSD (for datasets, embeddings, and model weights)
- **GPU:** NVIDIA GPU with CUDA support (for faster model inference, e.g., GTX 1050 Ti or above; optional if using CPU-only mode)
- **Network:** Stable internet connection (for API calls and dataset/model downloads)

## 2.2 SOFTWARE SPECIFICATION

- **Operating System:** Windows 10/11, Linux (Ubuntu 20.04+), or macOS
- **Backend:** Python 3.9+
- **Web Framework:** Flask for backend API and routing
- **Database:** SQLite (lightweight, file-based storage for timetables, seats, announcements)
- **Frontend:** HTML, CSS, JavaScript (responsive web interface)

## **2.3 ABOUT THE SOFTWARE AND ITS FEATURE**

### **2.3.1 RAG-POWERED QUERY PROCESSING**

The SRM Chatbot uses Retrieval-Augmented Generation (RAG) to combine semantic document retrieval and large language model (LLM) generation. It creates vector embeddings of structured and unstructured data, including timetables, seat allocations, and JSON datasets. Using FAISS for fast similarity search, the system identifies the most relevant documents for each user query. These retrieved documents are then processed by the LLM to generate coherent, context-aware answers, ensuring high accuracy even for questions not explicitly pre-programmed.

### **2.3.2 TIMETABLE MANAGEMENT**

The chatbot allows students to access exam schedules, department-specific timetables, and subject-wise exam details. It intelligently handles fuzzy matching and acronym expansion, enabling it to correctly interpret queries like “AI exam 2nd year CSE.” Results are displayed in a concise and readable format, making it easy for students to quickly find the information they need without confusion.

### **2.3.3 SEAT ALLOCATION CHECKER**

Students can retrieve their **seat numbers** by providing their register number, session, and date. The system supports natural language queries such as “What’s my seat for RA24322 on next Friday morning?” and ensures accurate retrieval from the SQLite database. This functionality allows students to check their seating arrangements easily and efficiently.

### **2.3.4 ANNOUNCEMENTS AND NOTIFICATIONS**

The SRM Chatbot enables administrators to add, edit, or delete announcements related to college events, exams, or important updates. Students can access urgent notifications in real-time, ensuring they stay informed about critical campus activities. This feature helps maintain effective communication across the university.

### **2.3.5 INTELLIGENT QUERY UNDERSTANDING**

The chatbot incorporates gibberish detection, fuzzy matching, and NLP preprocessing to interpret queries accurately, even if they contain typos, abbreviations, or variations. If relevant information is unavailable, the system provides a fallback response, ensuring that the user experience remains smooth and informative.

### **2.3.6 USER-FRIENDLY WEB INTERFACE**

Built with HTML, CSS, and JavaScript, the chatbot offers a responsive and interactive web interface. Students and staff can access the system 24/7, and the backend seamlessly integrates AI modules, database management, and retrieval systems to provide a fast, reliable experience.

### **2.3.7 ADDITIONAL FEATURES**

The SRM Chatbot also provides campus weather updates, collects user feedback for continuous improvement, and supports dynamic integration of new datasets, announcements, or timetables without retraining the model. These features enhance the overall utility of the system, making it a comprehensive and intelligent assistant for SRMIST students and staff.

### **3. SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM**

In the existing system, students at SRMIST rely heavily on manual processes and static information sources to get answers about timetables, seat allocations, announcements, and general college information. Students often have to check multiple sources such as notice boards, emails, official websites, or contact administrators personally to find the required information. This process is time-consuming, prone to errors, and not available 24/7. Additionally, queries related to exam schedules or seat allocations may be misunderstood due to inconsistent formats or unclear instructions, which often leads to confusion. There is no intelligent mechanism in place to understand natural language queries, interpret acronyms, or retrieve information from multiple sources dynamically.

#### **3.2 PROPOSED SYSTEM**

The proposed SRM Chatbot system aims to provide a smart, automated, and intelligent assistant for students and staff. It uses Retrieval-Augmented Generation (RAG) to combine the power of semantic document retrieval and large language models, providing precise and context-aware answers. The system consolidates all student-related information, including timetables, seat allocations, announcements, and FAQs, into a single interactive web interface. Students can ask queries in natural language, and the system interprets their intent accurately using fuzzy matching, NLP preprocessing, and acronym expansion. The chatbot is accessible 24/7, ensures fast response times, and continuously learns from feedback, making it a highly efficient solution compared to the existing manual system.

#### **3.3 FEASIBILITY STUDY**

A feasibility study helps in determining if the proposed SRM Chatbot can be successfully developed and deployed within the available resources and constraints. This study examines four main aspects: system feasibility, economic feasibility, technical feasibility, and operational feasibility. By analyzing these factors, we can ensure that the project will meet its objectives efficiently and provide tangible benefits to students and staff.

### **3.3.1 SYSTEM FEASIBILITY**

The SRM Chatbot is highly feasible from a system perspective because it uses existing technologies and frameworks such as Python, Flask, AI models (LLMs), FAISS for vector search, and SQLite for database management. The combination of RAG (Retrieval-Augmented Generation) and NLP allows the system to dynamically fetch relevant information and generate accurate responses. The architecture can be implemented using standard web and server resources, making the project technically achievable.

### **3.3.2 ECONOMIC FEASIBILITY**

Economically, the project is viable as it reduces manual administrative work, minimizes errors in information dissemination, and saves time for both students and staff. Since the system leverages open-source technologies and existing campus infrastructure, the development and deployment costs are minimal. The project also has long-term benefits in terms of efficiency and scalability, which further justifies the investment.

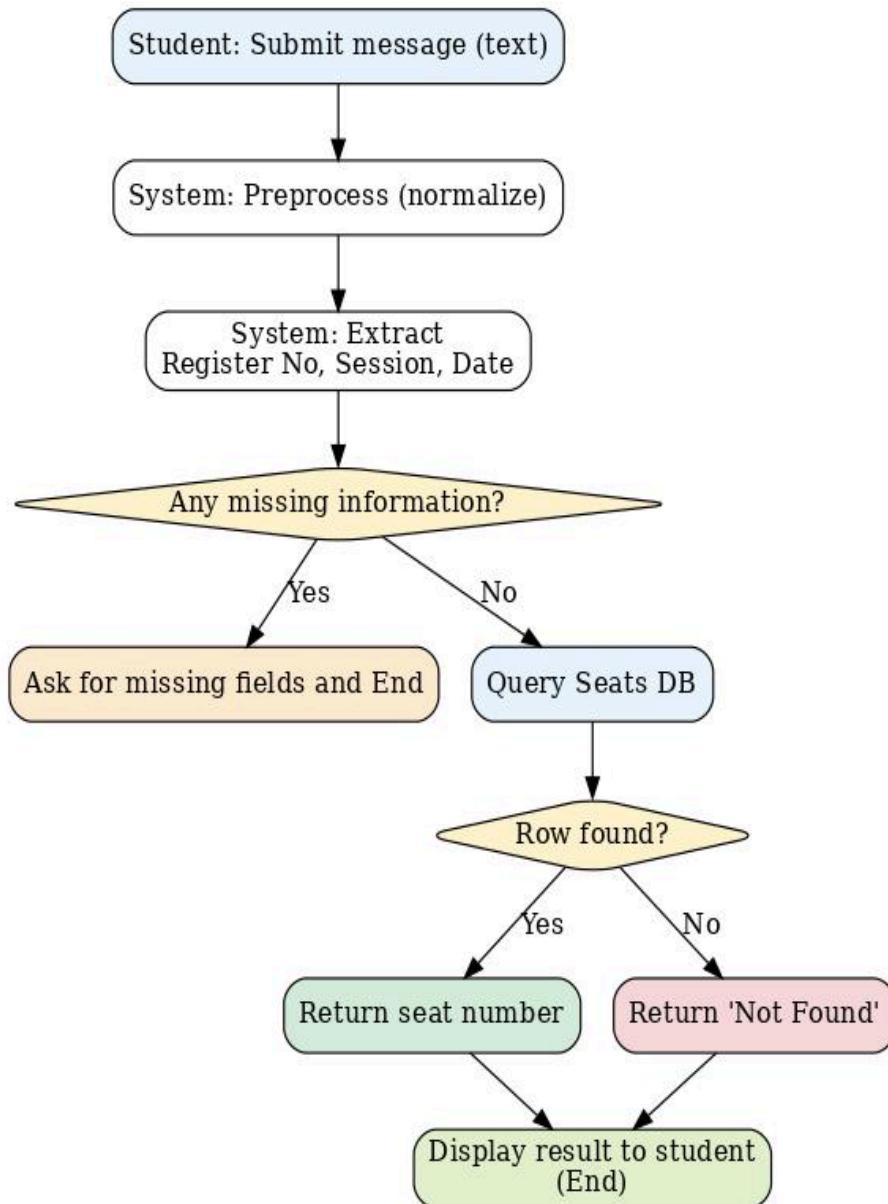
### **3.3.3 TECHNICAL FEASIBILITY**

The technical feasibility focuses on the availability of necessary hardware, software, and technical expertise. Modern computers, servers, and cloud resources are sufficient to handle the AI model and database operations. The system can efficiently manage multiple user queries concurrently, while the use of FAISS ensures fast retrieval from large datasets. The required AI, web, and database technologies are well-supported and documented, making implementation realistic.

### **3.3.4 OPERATIONAL FEASIBILITY**

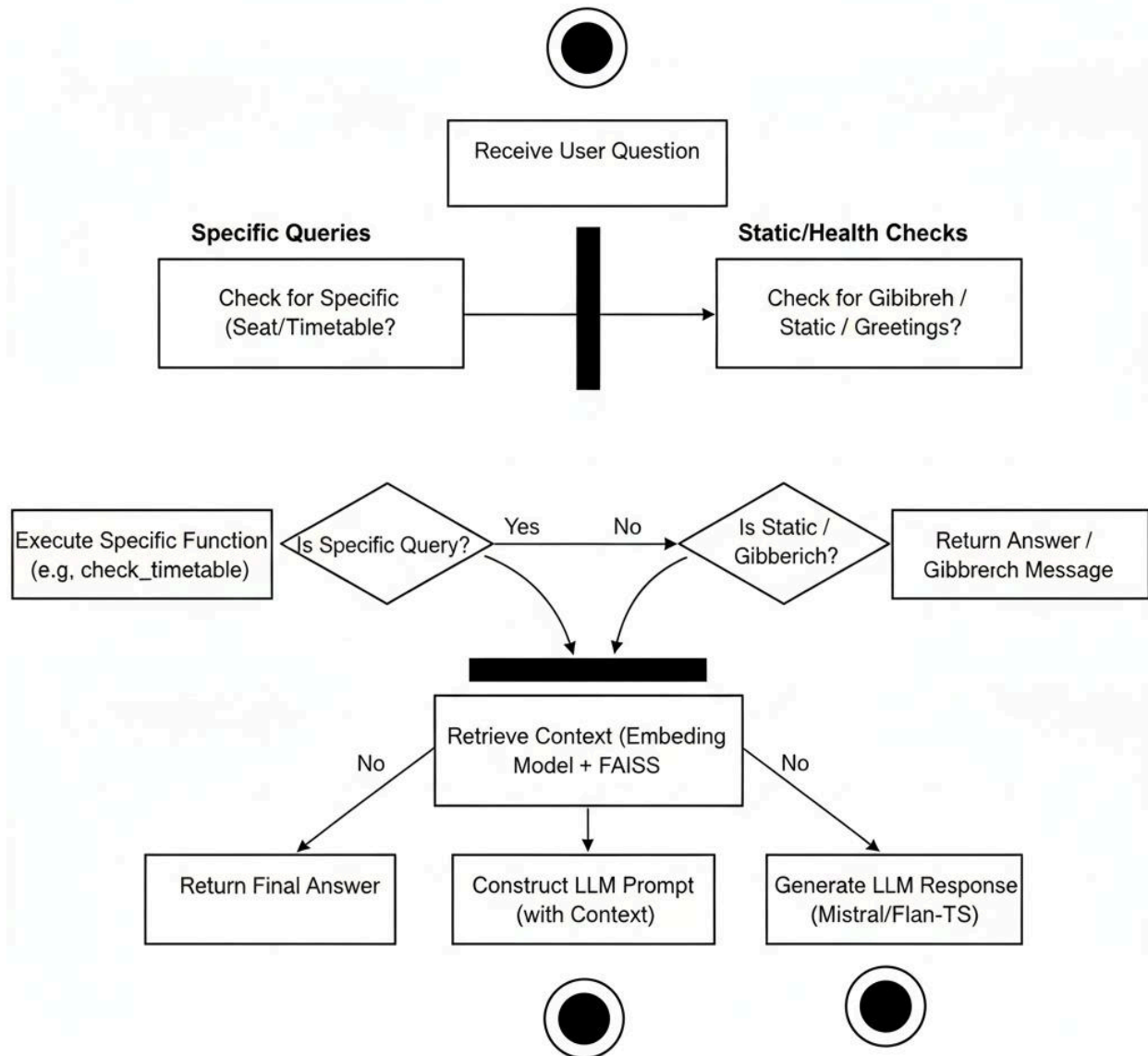
From an operational perspective, the chatbot is easy to use and integrates seamlessly with the existing workflow of students and administrators. The user-friendly web interface, 24/7 accessibility, and intelligent query handling ensure that users can interact with the system efficiently. Training requirements are minimal, and the system can quickly adapt to updates in timetables, seat allocations, or announcements, ensuring smooth day-to-day operations.

#### 4.1 DATA FLOW DIAGRAM



#### 4.2.2 QUERY PROCESSING FLOW

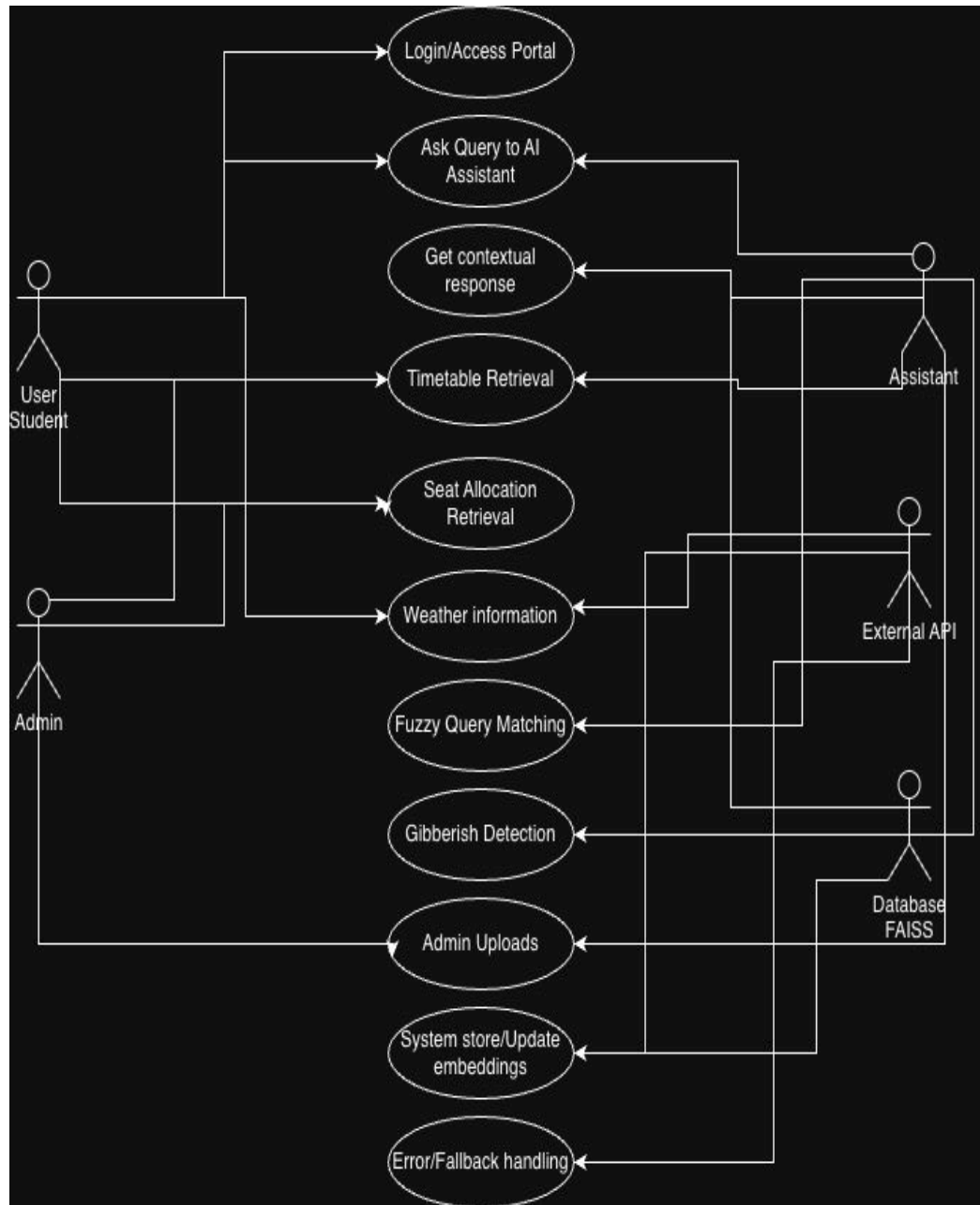
### SRM AI Assistant: Full Query Processing Flow



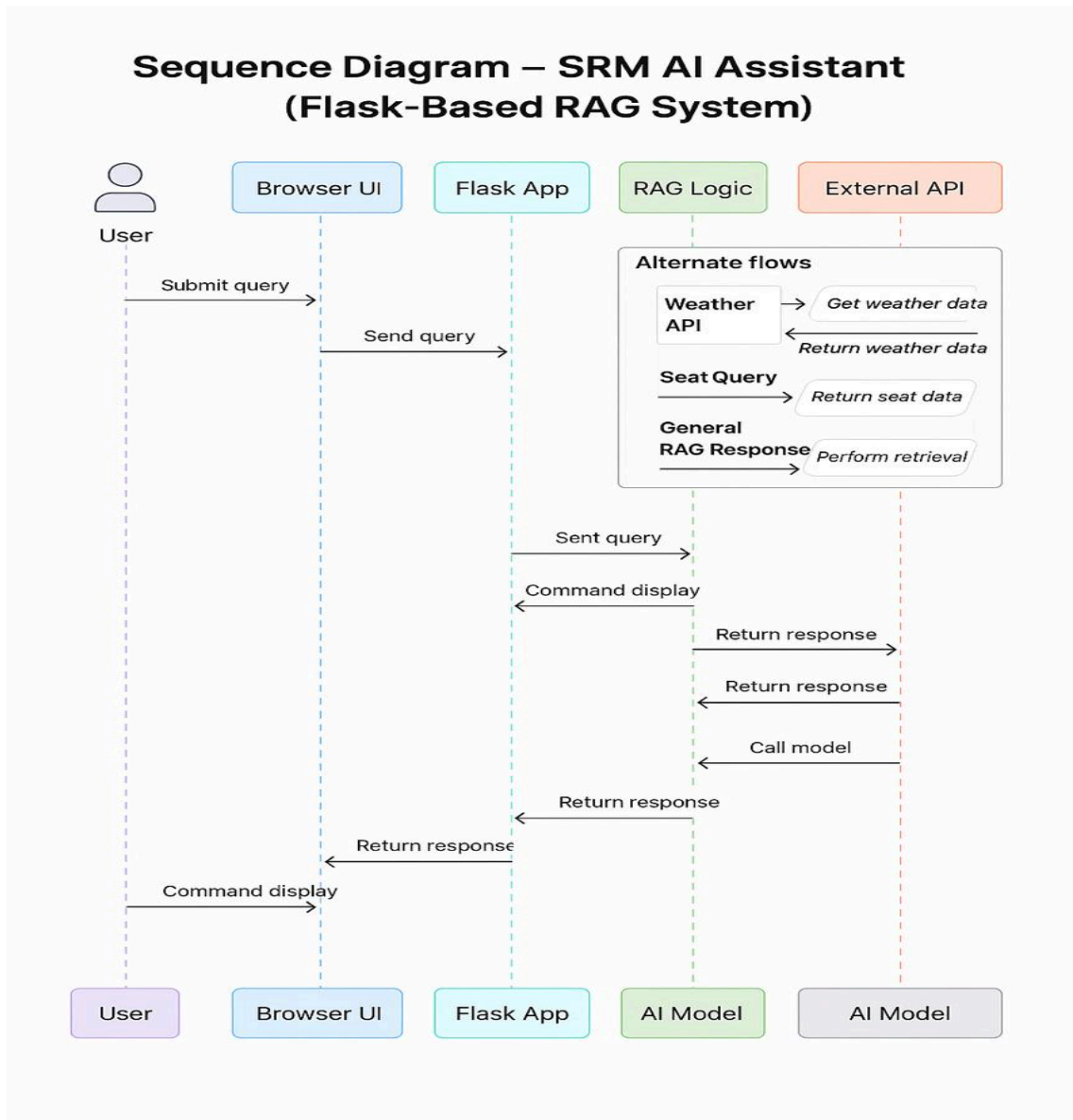


## 4.2 UML DIAGRAM

### 4.2.1 USE CASE DIAGRAM



#### 4.2.3 SEQUENCE DIAGRAM



## **5. CODE TEMPLATES**

### **5.1 MODULE DESCRIPTION**

This SRM Chatbot provides two main modules to operate efficiently:

#### **5.1.1 STUDENT MODULE**

##### **5.1.1.1 ASK QUESTIONS**

In this module, students can ask queries related to exam timetables, seat allocations, announcements, and general college information. The chatbot processes natural language questions and provides accurate answers using Retrieval-Augmented Generation (RAG).

##### **5.1.1.2 CHECK TIMETABLE**

This module allows students to view their department-specific and year-specific exam schedules. The chatbot can understand subject names, codes, acronyms, and provide detailed information such as date, time, and subject code.

##### **5.1.1.3 CHECK SEAT ALLOCATION**

In this module, students can check their exam seat numbers by providing their register number, session (FN/AN), and exam date. The chatbot retrieves the seat information from the database and displays it clearly.

##### **5.1.1.4 VIEW ANNOUNCEMENTS**

This module allows students to see all announcements, including urgent and regular updates. The announcements are fetched in descending order of date, ensuring that students always get the latest information.

##### **5.1.1.5 FEEDBACK**

Students can provide feedback on the chatbot responses. This module collects user feedback and stores it for analysis to improve response quality and overall system performance.

##### **5.1.1.6 GENERAL INFORMATION**

In this module, students can ask about SRM facilities, contact details, faculty information, and other campus-related queries. The chatbot uses RAG and preloaded datasets to answer these questions accurately.

## **5.1.2 ADMIN MODULE**

### **5.1.2.1 UPLOAD TIMETABLE**

This module allows admins to upload or update exam timetables in .xlsx format. The chatbot then dynamically fetches timetable information for student queries.

### **5.1.2.2 UPLOAD SEAT DATA**

Admins can upload seat allocation data for exams. The chatbot retrieves seat information from the uploaded database to respond to student queries.

### **5.1.2.3 MANAGE ANNOUNCEMENTS**

Admins can add, edit, or delete announcements. This module ensures that all students receive updated and relevant information.

### **5.1.2.4 VIEW FEEDBACK**

In this module, admins can review student feedback collected by the chatbot and analyze it to improve system performance.

### **5.1.2.5 MONITOR CHATBOT PERFORMANCE**

This module allows admins to track system queries, errors, and logs to ensure the chatbot operates efficiently and accurately.

## 5.2 TABLES

### 5.2.1 Seat Data

SEAT DATA			
Register Number	Session	Date	Seat Number
RA243224201010095	FN	13/10/2025	A24
RA243224201010129	AN	13/10/2025	B26
RA243224201010113	FN	13/10/2025	A16

### 5.2.2 Time Table Data

TIME TABLE					
Department	Year	Date	Subject	Code	Time
MCA	1	7-11-2025	Object Oriented Analysis And Design	PGICJ12A	2PM - 5PM
MCA	1	8-11-2025	Ethical Hacking	PGICJ12B	2PM - 5PM
MCA	1	9-11-2025	Artificial Intellegence	PGICJ12C	2PM - 5PM
MCA GEN AI	1	7-11-2025	Object Oriented Analysis And Design	PGICJ12D	2PM - 5PM
MCA GEN AI	1	8-11-2025	Full Stack Development	PGICJ12E	2PM - 5PM
MCA GEN AI	1	9-11-2025	Cognitive Analytical Skills	PGICJ12F	2PM - 5PM

## **6. TESTING**

Testing is a critical phase in the development of the SRM Chatbot to ensure that the system is reliable, accurate, and user-friendly. Multiple testing methodologies are applied to validate the system's performance, functionality, and usability.

### **6.1 TESTING METHODOLOGIES**

#### **6.1.1 UNIT TESTING**

Unit testing focuses on verifying the smallest components of the chatbot system individually. Functions for seat allocation retrieval, timetable queries, session parsing, and RAG-based responses are tested separately. This ensures that each module works as expected before integration. For example, the function that extracts the student register number from a query is tested with different input formats to confirm accurate extraction.

#### **6.1.2 INTEGRATION TESTING**

Integration testing validates that the modules work together correctly. This involves testing the interaction between the frontend interface, Flask server, AI model, and SQLite database. Queries from students are passed through the system, and the responses are checked for correctness and timeliness. Integration testing ensures seamless communication between all components.

#### **6.1.3 SYSTEM TESTING**

System testing evaluates the entire chatbot application as a whole. It simulates real-world scenarios, such as students asking multiple types of queries, uploading timetables, or checking seat allocations simultaneously. The objective is to verify that the SRM Chatbot functions correctly under normal and peak usage conditions.

#### **6.1.4 FUNCTIONALITY TESTING**

Functional testing ensures that all features work as intended. Key functionalities tested include RAG-based query answering, timetable and seat checking, announcement retrieval, feedback submission, and natural language understanding. Functional tests validate that the system meets all specified requirements.

### 6.1.5 USER ACCEPTANCE TESTING

UAT is performed by actual students and administrators to confirm that the chatbot meets their expectations. Feedback is collected regarding ease of use, clarity of responses, accuracy, and overall experience. Any discrepancies or issues found during UAT are corrected before final deployment.

### 6.1.6 PERFORMANCE TESTING


Performance testing evaluates the system's response time, scalability, and reliability. Queries are simulated under heavy load to ensure the chatbot can handle multiple users without delays or errors. This guarantees a smooth experience for all students during high-demand periods, such as exam schedules.

### 6.1.7 REGRESSION TESTING

Regression testing is carried out whenever updates or new features are added. It ensures that existing functionalities continue to perform correctly and that no new errors are introduced.

## 6.2 TEST CASE

Test Case ID	Module	Test Description	Input	Expected Output	Result
TC001	Student Module – Ask Questions	Verify that the chatbot responds to a general question about SRM	Who is the director of SRM?	The director of SRM Institute of Science and Technology is Dr. [Name].	Pass/Fail
TC002	Student Module – Check Timetable	Verify timetable query using subject name	When is the AI exam for CSE 2nd year?	Artificial Intelligence exam for CSE (Year 2) is on 20	Pass/Fail

				Oct 2025 from 10:00 AM.	
TC003	Student Module – Check Timetable	Verify timetable query using subject code	Tell me the date for AI101 exam.	AI101 exam for CSE (Year 2) is on 20 Oct 2025 from 10:00 AM.	Pass/Fail
TC004	Student Module – Check Seat Allocation	Verify seat check for valid register number, date, and session	RA24322420101 01 FN 20/10/2025	 Seat for RA24322420101 01 on 2025-10-20 (FN): 45	Pass/Fail
TC005	Student Module – Check Seat Allocation	Verify seat check for missing information	RA24322420101 01 FN	Please provide your date (MM/DD/YYYY, YYYY-MM-DD, or natural language like 'tomorrow', 'next Friday') to check your seat.	Pass/Fail
TC006	Student Module – View Announcements	Verify fetching all announcements	Show announcements	List of announcements with title, date, message, and urgency flag	Pass/Fail
TC007	Student Module – Feedback	Verify feedback submission	Feedback: "Helpful", Message: "The timetable info is accurate"	Status: Success. Feedback submitted.	Pass/Fail



TC008	Admin Module – Upload Timetable	Verify timetable upload with correct file	Upload "timetable.xlsx"	Status: Success. Timetable uploaded successfully!	Pass/Fail
TC009	Admin Module – Upload Seats	Verify seat data upload with correct file	Upload "seat_data.xlsx"	Status: Success. Seat data updated successfully!	Pass/Fail
TC010	Admin Module – Manage Announcements	Verify adding a new announcement	Title: "Holiday Notice", Message: "SRM will be closed on 25th Oct", Date: "2025-10-25", Urgent: Yes	Status: Success. Announcement added!	Pass/Fail
TC011	Student Module – RAG Query	Verify response for query with multiple words	Tell me about AI subject exam for CSE 2nd year	Properly generated RAG-based response about exam details	Pass/Fail
TC012	Student Module – Gibberish Input	Verify chatbot handles invalid input	asdhjkl qweoi	Hmm... I cannot understand that! Could you rephrase that?	Pass/Fail
TC013	Student Module – Weather	Verify fetching current weather	What is the temperature in Potheri?	The current temperature in SRM Potheri, Chennai is 32°C with clear sky.	Pass/Fail
TC014	Student Module – Date & Time Query	Verify chatbot responds to current date/time	What is the time now?	The current time is 14:35.	Pass/Fail

## 7. OUTPUT SCREENS

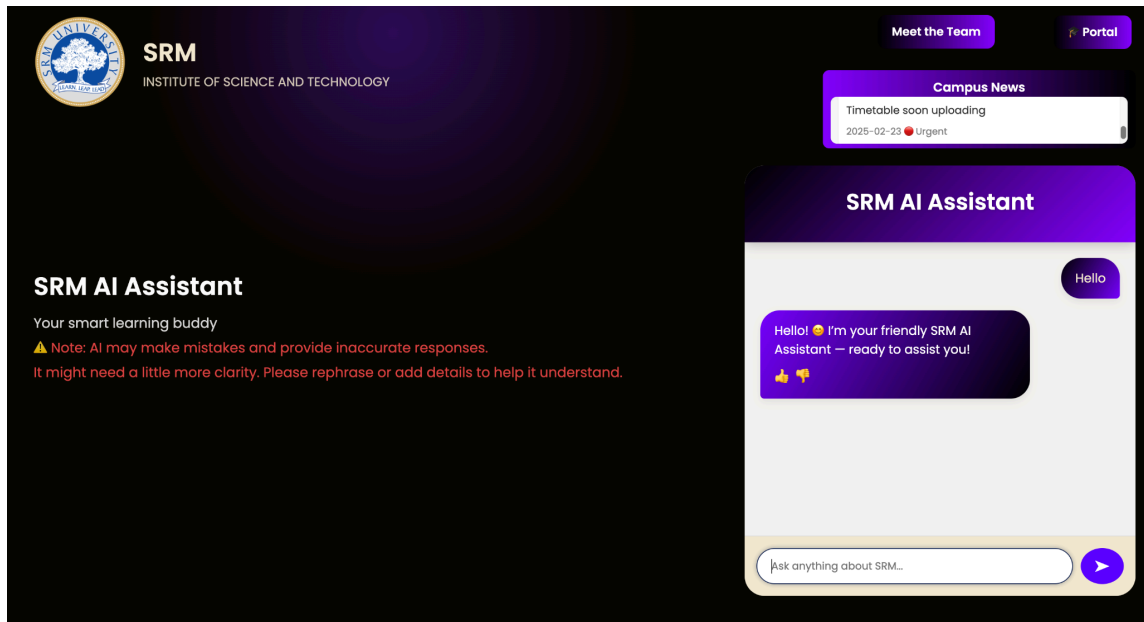


Figure 7.1 Chatbot User Interface

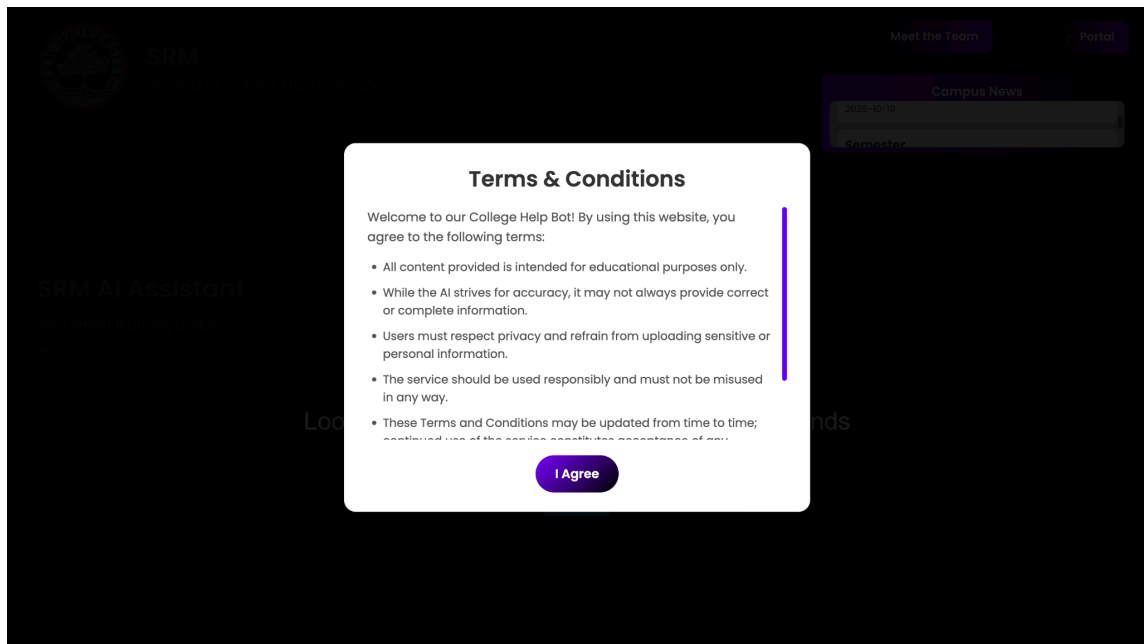


Figure 7.2 Terms And Conditions

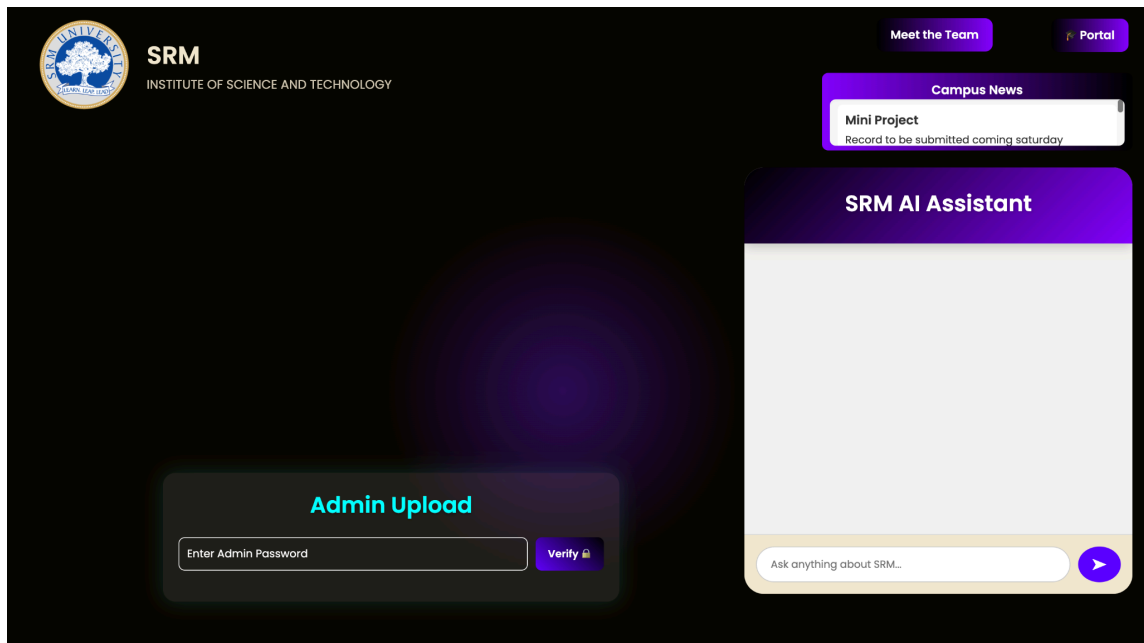


Figure 7.3 Admin Access Page

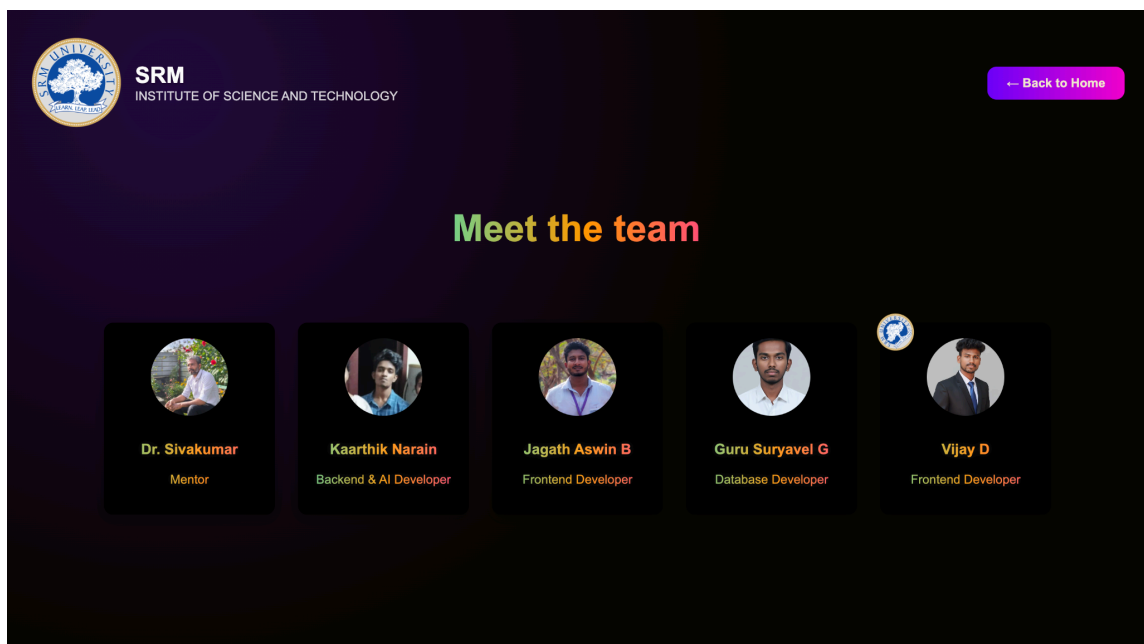


Figure 7.4 Meet the Team Page

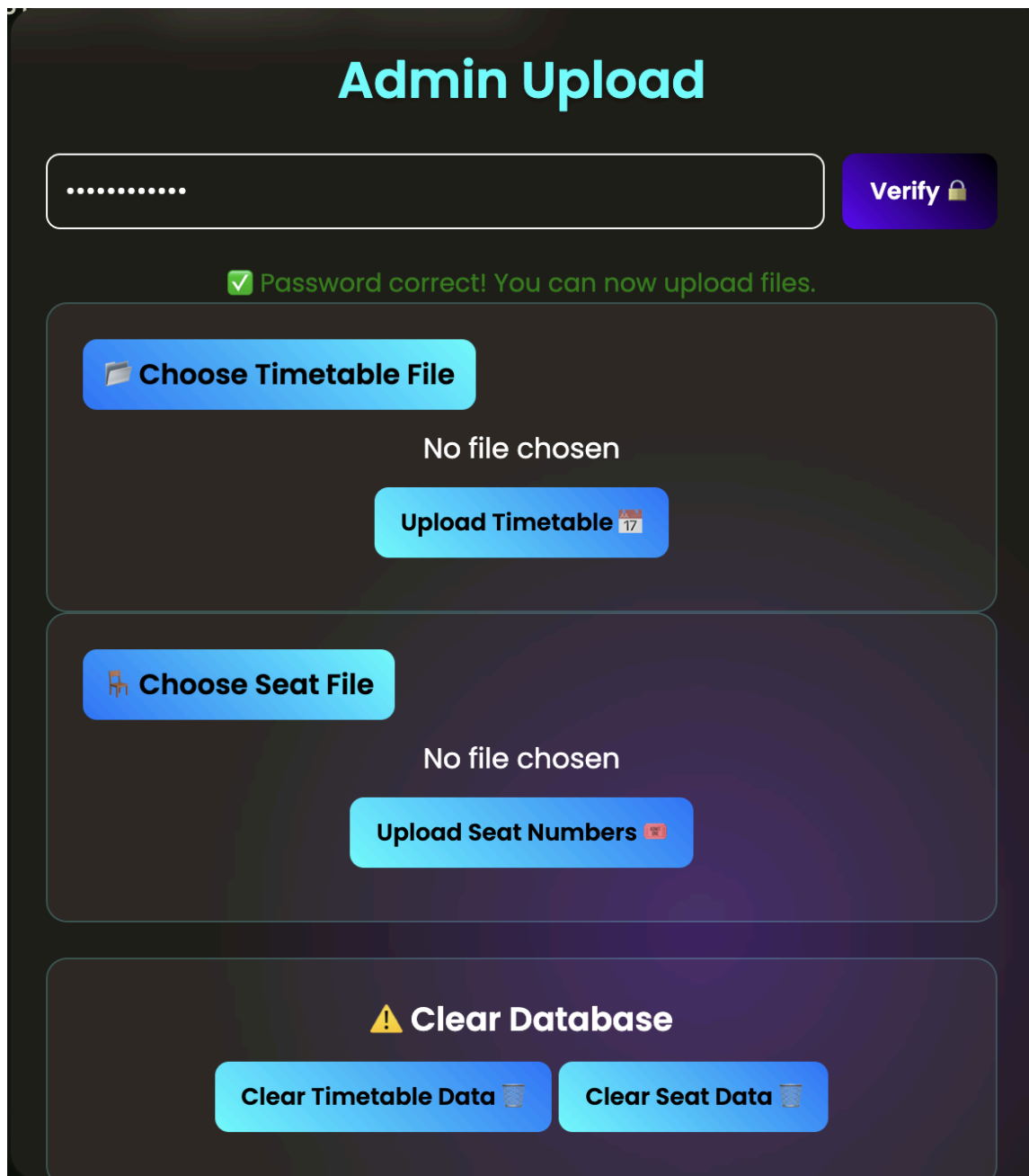


Figure 7.5 Admin Upload Module

⚠️ Clear Database

Clear Timetable Data 🗑️

Clear Seat Data 🗑️

### Campus News Management

Title

Message

dd/mm/yyyy 📅

☐

Mark as urgent

Add News 📰

#### Delete Existing News

-- Select News to Delete --

Delete Selected

**Figure 7.6** Admin news management Module

## 8. CONCLUSION

The SRM Chatbot project demonstrates the effective integration of AI and web technologies to enhance student engagement and streamline access to information within SRM University. By leveraging Retrieval-Augmented Generation (RAG) techniques, the chatbot can provide accurate, context-aware responses to a wide variety of student queries, including exam timetables, seat allocations, announcements, and general college-related information. The system intelligently handles both structured data, such as timetable and seat records stored in SQLite, and unstructured data from JSON knowledge bases, ensuring comprehensive coverage of student queries.

The chatbot's capability to process natural language inputs, handle flexible date formats, and recognize variations in subject names or acronyms greatly improves usability and accessibility. Additionally, the gibberish detection module ensures that irrelevant or malformed inputs are filtered, maintaining the quality of interactions. The administrative modules, including timetable and seat data uploads and announcement management, provide a robust interface for managing content efficiently. By automating repetitive tasks and providing timely responses, the system reduces the workload of administrative staff and improves the overall experience for students.

Furthermore, the integration of analytics and dynamic RAG responses enables the chatbot to continuously improve over time, providing more accurate and relevant answers. The web-based interface, combined with AI-driven responses, ensures accessibility from multiple devices, supporting the needs of a modern educational environment. Overall, the SRM Chatbot exemplifies a practical application of AI in higher education, promoting efficient information dissemination, reducing response time, and enhancing student satisfaction. This project sets a foundation for future expansions, including multilingual support and more advanced predictive query handling.

## 9. FURTHER ENHANCEMENTS

The SRM Chatbot project, while already functional and efficient, has significant scope for further enhancements to make it more intelligent, interactive, and user-friendly. One potential enhancement is the integration of multilingual support. Currently, the chatbot primarily handles queries in English. By incorporating support for regional languages like Tamil, Telugu, and Hindi, the system can cater to a wider student base and improve accessibility for non-English speakers.

Another improvement could be the implementation of voice-based interaction. Integrating speech-to-text and text-to-speech modules would allow students to interact with the chatbot through voice commands, making the system more interactive and suitable for mobile or hands-free usage. This can particularly benefit visually impaired students or those on the move.

Enhancing the RAG knowledge base is another area of growth. By continuously updating the knowledge base with the latest announcements, academic guidelines, and FAQs, the chatbot can provide more precise and real-time information. Integration with other university systems such as the Learning Management System (LMS) or library database can further expand the range of information the chatbot can provide.

Additionally, implementing predictive query suggestions and personalized recommendations can make interactions more intuitive. For example, based on past queries or user profiles, the chatbot could proactively suggest relevant exam updates, seat allocation notifications, or departmental announcements.

Finally, advanced analytics can be incorporated to track user engagement, commonly asked questions, and response accuracy. This data can guide continuous improvements to the chatbot, ensuring it evolves to meet student needs more effectively. Overall, these enhancements can transform the SRM Chatbot into a more intelligent, adaptive, and comprehensive campus assistant.

## 10. REFERENCES

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. Advances in Neural Information Processing Systems, 30, 5998–6008.
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., & Levy, O. (2020). *BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*. arXiv preprint arXiv:1910.13461.
- Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). *Dense passage retrieval for open-domain question answering*. Proceedings of EMNLP 2020.
- Hugging Face Documentation. (2025). *Transformers and pipelines*. Retrieved from <https://huggingface.co/docs/transformers>
- Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence embeddings using Siamese BERT-networks*. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.
- RapidFuzz Documentation. (2025). *String matching and fuzzy searching in Python*. Retrieved from <https://maxbachmann.github.io/RapidFuzz/>
- SQLite Documentation. (2025). *Using SQLite with Python*. Retrieved from <https://www.sqlite.org/docs.html>
- SRM Institute of Science and Technology. (2025). *Official website and student resources*. Retrieved from <https://www.srmist.edu.in>
- OpenWeatherMap API Documentation. (2025). *Current weather data API*. Retrieved from <https://openweathermap.org/api>
- NLTK Documentation. (2025). *Natural Language Toolkit*. Retrieved from <https://www.nltk.org>



## **11. APPENDICES**

### **11.1 USER DOCUMENTATION**

#### **11.1.1 INSTALLATION INSTRUCTION**

##### **Step 1: Install Python and Dependencies**

1. Ensure Python 3.8+ is installed.
2. Install required libraries using pip:

```
pip install flask pandas nltk rapidfuzz faiss-cpu transformers  
sentence-transformers requests
```

##### **Step 2: Download Project Files**

1. Clone or download the project folder containing app.py, templates/, static/, and srm\_data.json.

##### **Step 3: Setup Database**

1. The project automatically creates srm\_data.db with necessary tables if not present.
2. Admin can upload timetable.xlsx and seats.xlsx via the web interface to populate tables.

##### **Step 4: Run the Application**

1. Open terminal in project directory.

2. Run the Flask app:

```
python app.py
```

3. Access the chatbot at <http://127.0.0.1:5001> in your browser.

### **Step 5: Optional Configuration**

- Update `srm_data.json` to add new question-answer pairs.
- Replace `YOUR_HUGGINGFACE_TOKEN` with a valid token.
- Update `API_KEY` for OpenWeatherMap in [app.py](#).

## **11.2 README**

### **11.2.1 STUDENT MODULE GUIDE**

The student module of the SRM Chatbot allows students to interact seamlessly with the system to get information related to SRM University. Students can type their queries in the input box on the home page, which can include questions about exam timetables, seat allocations, college events, or general SRM-related information. The chatbot responds in real-time, providing answers based on its retrieval-augmented generation (RAG) knowledge base, database lookups, or predefined responses. For timetable queries, students can ask about the exam schedule of a specific department and year, or for a particular subject, and the chatbot retrieves and displays the relevant information in a clear and concise format. To check seat allocations, students need to provide their register number, the session (such as FN, AN, morning, or afternoon), and the date of the exam. The chatbot then queries the database and returns the allocated seat number. Students can also view the latest announcements, where the chatbot fetches urgent and regular announcements from the database and presents them in order of priority. Additionally, students can inquire about the current weather conditions at SRM Potheri, Chennai, and the chatbot fetches real-time information using the OpenWeatherMap API. Another important feature is the feedback system, which allows students to submit their

feedback on chatbot responses. This feedback is stored for admin review and helps improve the system's performance and accuracy.

## **11.2.2 ADMIN MODULE GUIDE**

The admin module is designed to manage the backend operations of the SRM Chatbot efficiently. Admins can upload exam timetables in Excel format, and the system validates the file to ensure all required columns are present before storing the data in the SQLite database. Similarly, seat allocation details can be uploaded through an Excel file, which is also validated and saved to the database. Admins have full control over announcements, enabling them to add new announcements with details such as title, message, date, and urgency, as well as delete existing announcements. The module also provides the option to clear old timetable or seat data to prevent conflicts with new uploads. Another critical feature of the admin module is feedback monitoring, which allows admins to review student feedback collected by the chatbot and make necessary improvements. Moreover, admins can update the chatbot's knowledge base by editing the `srm_data.json` file to add or modify question-answer pairs, ensuring that the system remains accurate, relevant, and up-to-date. Overall, the admin module ensures that the SRM Chatbot operates smoothly while keeping all student-facing information reliable and current.

## **11.3 SAMPLE SOURCE PROGRAM**

### **11.3.1 CHATBOT USER INTERFACE**

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>SRM AI Assistant ✨</title>
```

```
<link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
```

```
<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap
" rel="stylesheet">
```

```
</head>
```

```
<body>
```

```
<div class="mouse-effect"></div>
```

```
<!-- Terms & Conditions Modal -->
```

```
<div id="tcModal" class="modal">
```

```
<div class="modal-content">
```

```
<h2>Terms & Conditions</h2>
```

```
<div class="tc-text">
```

```
<p>Welcome to our College Help Bot! By using this website, you agree to the
following terms:</p>
```

```
<ul>
```

```
<li>All content provided is intended for educational purposes only.</li>
```

```
<li>While the AI strives for accuracy, it may not always provide correct or
complete information.</li>
```

```
<li>Users must respect privacy and refrain from uploading sensitive or personal
information.</li>
```

```
<li>The service should be used responsibly and must not be misused in any
way.</li>
```

```
<li>These Terms and Conditions may be updated from time to time; continued use
of the service constitutes acceptance of any changes.</li>
```

<li>No personal data is stored by the service; however, feedback may be utilized to improve and train the AI.</li>

</ul>

</div>

<button id="acceptBtn">I Agree</button>

</div>

</div>

<header class="srm-header">

<div class="srm-container">



<div class="srm-text">

<h1>SRM</h1>

<p>INSTITUTE OF SCIENCE AND TECHNOLOGY</p>

</div>

</div>

</header>

<!-- Floating Buttons -->

<button id="portalBtn" class="shortcut-btn">🎓 Portal</button>

<button id="backBtn" class="shortcut-btn" style="position:fixed;right:200px;top:19.6px;z-index: 10000;">Meet the Team</button>

```
<script>

document.getElementById("backBtn").addEventListener("click", () => {

    window.location.href = "/team"; // <-- Flask route for team page

});

</script>
```

```
<canvas id="matrixCanvas"></canvas>
```

```
<div id="rightPanel">

    <h3>Campus News</h3>

    <div id="alertsSidebar" class="news-container"></div>

</div>
```

```
<!-- Side Text -->
```

```
<div class="side-text">

    <h2>SRM AI Assistant</h2>

    <p>Your smart learning buddy</p>

    <p class="disclaimer">⚠️ Note: AI may make mistakes and provide inaccurate responses.</p>

    <p class="disclaimer">It might need a little more clarity. Please rephrase or add details to help it understand.</p>

</div>
```

```

<!-- Chat Wrapper -->

<div class="chat-wrapper">

  <div class="header">

    <h1>SRM AI Assistant</h1>

  </div>

  <div class="chat-box" id="chat-box"></div>

  <div class="input-area">

    <input id="user-input" type="text" placeholder="Ask anything about SRM..."
autofocus autocomplete="off">

    <button id="send-btn">➤</button>

  </div>

</div>

```

```

<!-- Admin Panel -->

<section id="adminPanel" class="admin-panel">

  <h2>Admin Upload</h2>

  <div class="password-container">

    <input type="password" id="adminPassword" placeholder="Enter Admin
Password">

    <button id="verifyPasswordBtn">Verify 🔑 </button>

  </div>

  <p id="passwordStatus" class="status-text"></p>

```

```

<!-- Timetable Upload -->

<div id="timetableUploadDiv" class="upload-section" style="display:none;">

  <div class="file-upload">

    <label for="timetableFile" class="custom-file-label">📁 Choose Timetable
File</label>

    <input type="file" id="timetableFile" accept=".xlsx">

  </div>

  <p class="file-name" id="timetableFileName">No file chosen</p>

  <button id="uploadTimetableBtn">Upload Timetable 📅 17</button>

  <p id="timetableStatus" class="status-text"></p>

</div>

<!-- Seat Upload -->

<div id="seatUploadDiv" class="upload-section" style="display:none;">

  <div class="file-upload">

    <label for="seatFile" class="custom-file-label">📁 Choose Seat File</label>

    <input type="file" id="seatFile" accept=".xlsx">

  </div>

  <p class="file-name" id="seatFileName">No file chosen</p>

  <button id="uploadSeatBtn">Upload Seat Numbers 📅 17</button>

  <p id="seatStatus" class="status-text"></p>

</div>

```



```

<!-- Clear Data -->

<div id="clearDataDiv" class="upload-section" style="display:none;
margin-top:20px;">

    <h3>⚠ Clear Database</h3>

    <button onclick="clearData('timetable')">Clear Timetable Data 🗑</button>

    <button onclick="clearData('seats')">Clear Seat Data 🗑</button>

    <p id="clearStatus" class="status-text"></p>

</div>


<div id="newsUploadDiv" style="display:none">

    <h3>Campus News Management</h3>


    <div class="news-flex-container">


        <!-- Add News Column -->

        <div class="news-add-column">

            <div class="news-input-row">

                <input type="text" id="newsTitle" placeholder="Title">

                <textarea id="newsMessage" placeholder="Message"></textarea>

            </div>


            <input type="date" id="newsDate">

            <label>

```

```

        <input type="checkbox" id="newsUrgent"> Mark as urgent
    </label>

    <button id="addNewsBtn">Add News  </button>

    <p id="newsStatus"></p>
</div>

<!-- Delete News Column -->
<div class="news-delete-column">

    <h4>Delete Existing News</h4>

    <select id="delete-news-select">

        <option value="">-- Select News to Delete --</option>

    </select>

    <button id="delete-news-btn">Delete Selected</button>

</div>

</div>

</div>

</section>

<script src="{{ url_for('static', filename='script.js') }}"></script>

<script src="{{ url_for('static', filename='announcement.js') }}"></script>

</body>

```

```
</html>
```

### 11.3.2 CSS FOR USER INTERFACE

```
/* Reset & font */
```

```
* {  
  
  box-sizing: border-box;  
  
  margin: 0;  
  
  padding: 0;  
  
  font-family: 'Poppins', sans-serif;  
  
}
```

```
body {  
  
  background: #0a0700;  
  
  height: 100vh;  
  
  display: block;  
  
  justify-content: center;  
  
  align-items: center;  
  
}
```

```
body {  
  
  min-height: 200vh; /* or some height taller than viewport */  
  
}
```

```

/* Push chat wrapper below header */

.chat-wrapper {
  position: fixed;
  bottom: 70px;
  right: 20px;
  width: 500px;
  max-width: 95%;
  height: 550px;
  background: #f2efefcc;
  backdrop-filter: blur(12px);
  border-radius: 25px;
  box-shadow: 0 10px 30px rgba(0,0,0,0.25);
  display: flex;
  flex-direction: column;
  overflow: hidden;
  z-index: 9999;
}

/* Chat box */

.chat-box {
  flex: 1;          /* take all remaining space */

```

```
padding: 20px;

overflow-y: auto;      /* scroll inside chat box */

display: flex;

flex-direction: column;

gap: 15px;

background: #ffffff;   /* SRM white */

scrollbar-width: thin;

scrollbar-color: rgba(19,36,80,0.5) transparent;

}
```

```
/* Header */
```

```
.header {

    position:static;

    background: linear-gradient(135deg, #000000, #8400ff);

    color: #fff;

    padding: 25px 20px;

    text-align: center;

    box-shadow: 0 4px 15px rgba(0,0,0,0.2);

    position: relative;

}
```

```
.header .logo {  
  width: 150px;  
  height: auto;  
  display: block;  
  margin: 0 auto 10px auto;  
  opacity: 0;  
  animation: fadeInLogo 1.5s forwards;  
}
```

```
@keyframes fadeInLogo {  
  0% { opacity: 0; transform: scale(0.5); }  
  100% { opacity: 1; transform: scale(1); }  
}
```

```
.header h1 {  
  font-size: 1.8rem;  
  margin-bottom: 5px;  
  text-shadow: 1px 1px 3px rgba(0,0,0,0.2);  
}
```

```
.header p {  
  font-size: 1rem;
```

```
    opacity: 0.85;  
}
```

```
/* Chat box */
```

```
.chat-box {  
    flex: 1;  
    padding: 20px;  
    overflow-y: auto;  
    display: flex;  
    flex-direction: column;  
    gap: 15px;  
    background: #f4f3f0; /* SRM white */  
}
```

```
/* Message bubbles */
```

```
.message {  
    max-width: 75%;  
    padding: 14px 18px;  
    border-radius: 25px;  
    line-height: 1.5;  
    word-wrap: break-word;  
    box-shadow: 0 4px 10px rgba(0,0,0,0.08);
```

```

    animation: fadeIn 0.3s ease-in-out;

    position: relative;
}

.user-message {

    background: linear-gradient(120deg, #000000, #7700ff); /* SRM dark blue shades */
    color: #f2e8cf; /* white text */
    align-self: flex-end;
    border-bottom-right-radius: 5px;
}

.assistant-message {

    background: linear-gradient(120deg, #7700ff, #000000); /* SRM gold shades */
    color: #ffffff; /* blue text */
    align-self: flex-start;
    border-bottom-left-radius: 5px;
}

/* Typing indicator */

.typing {

    display: flex;
    align-items: center;

```



```
gap: 4px;  
}
```

```
.typing span {  
  width: 6px;  
  height: 6px;  
  background: #132450; /* blue dots */  
  border-radius: 50%;  
  animation: blink 1s infinite;  
}
```

```
.typing span:nth-child(2) { animation-delay: 0.2s; }  
.typing span:nth-child(3) { animation-delay: 0.4s; }
```

```
/* Input area */  
.input-area {  
  display: flex;  
  padding: 15px;  
  background: #f2e8cf;  
  border-top: 1px solid #ddd;  
}
```

```
.input-area input {  
    flex: 1;  
    padding: 12px 18px;  
    border-radius: 30px;  
    border: 1px solid #ccc;  
    outline: none;  
    transition: 0.3s;  
}
```

```
.input-area input:focus {  
    border-color: #132450;  
    box-shadow: 0 0 6px rgba(19,36,80,0.4);  
}
```

```
.input-area button {  
    margin-left: 10px;  
    padding: 0 18px;  
    border: none;  
    border-radius: 50%;  
    background: #5d00ff;  
    color: #ffffff;  
    cursor: pointer;
```

```
font-size: 1.3rem;

transition: 0.2s;

}
```

```
.input-area button:hover {

background: #000000;

color: #ffffff;

}
```

```
/* Animations */

@keyframes fadeIn {

0% {opacity: 0; transform: translateY(20px);}

100% {opacity: 1; transform: translateY(0);}

}
```

```
@keyframes blink {

0%, 80%, 100% {opacity: 0;}

40% {opacity: 1;}

}
```

```
/* Scrollbar styling */

.chat-box::-webkit-scrollbar {
```

```

width: 6px;

}

.chat-box::-webkit-scrollbar-thumb {

background: rgba(19,36,80,0.5);

border-radius: 3px;

}


/* Full-width SRM header */

.srm-header {

position: fixed;

top: 0;

left: 0;


width: 100%;

padding: 20px 40px;

box-shadow: none;

z-index: 9998;

display: flex;

justify-content: flex-start; /*  Push content to the left */


align-items: center;

background: rgba(0,0,0,0);

animation: fadeInHeader 1s ease forwards;

}

```

```
.srm-container {  
  display: flex;  
  align-items: center;  
  justify-content: flex-start; /*  Align items to left */  
  width: 100%;  
  gap: 20px;  
  margin-left: 0; /* remove any margin */  
}
```

```
.srm-logo {  
  width: 120px;  
  height: auto;  
  animation: slideInLeft 1s ease forwards;  
}
```

```
@keyframes slideInLeft {  
  from { opacity: 0; transform: translateX(-30px); }  
  to { opacity: 1; transform: translateX(0); }  
}
```

```
/* Text next to logo */
```

```
.srm-text h1 {  
    font-size: 2rem;  
    color: #f2e8cf; /* SRM white */  
    margin: 0;  
}
```

```
.srm-text p {  
    font-size: 1rem;  
    color: #f2e8cf; /* SRM white */  
    opacity: 0.85;  
    margin: 2px 0 0 0;  
}
```

```
/* Fade-in animation */  
  
@keyframes fadeInHeader {  
    0% {opacity: 0; transform: translateY(-20px);}  
    100% {opacity: 1; transform: translateY(0);}  
}
```

```
.mouse-effect {  
    position: fixed;  
    top: 0;
```

```
left: 0;

width: 100%;

height: 100%;

pointer-events: none;

background: radial-gradient(circle at 0 0, rgb(13, 0, 254), transparent 25%);

transition: background 0.1s ease;

z-index: 1;

}
```

```
.message:hover {

transform: translateY(-1.5px) scale(1.01);

transition: transform 0.15s ease, box-shadow 0.15s ease;

box-shadow: 0 5px 10px rgba(0,0,0,0.12);

}
```

```
/* Full-screen overlay for modal */

.modal {

display: flex;

justify-content: center;

align-items: center;

position: fixed;

z-index: 99999; /* make sure it's above everything */

}
```

```
left: 0;

top: 0;

width: 100%;

height: 100%;

background: rgba(0,0,0,0.9); /* almost fully opaque dark background */

pointer-events: auto; /* capture clicks */

}
```

```
/* Modal content */
```

```
.modal-content {

    background: #ffffff; /* fully opaque */

    border-radius: 15px;

    width: 90%;

    max-width: 600px;

    padding: 25px 30px;

    box-shadow: 0 15px 40px rgba(0,0,0,0.3);

    position: relative;

    z-index: 100000; /* above the overlay */

}
```

```
/* Heading */
```

```
.modal-content h2 {
```



```

    font-family: 'Poppins', sans-serif;

    font-size: 28px;

    margin-bottom: 15px;

    color: #333;

    text-align: center;
}

/* Terms text */

.tc-text {

    max-height: 300px; /* scrollable if content overflows */

    overflow-y: auto;

    padding-right: 10px;
}

.tc-text p {

    font-size: 16px;

    line-height: 1.6;

    margin-bottom: 15px;

    color: #555;
}

.tc-text ul {

```

```
list-style: disc;

padding-left: 20px;

color: #555;

}
```

```
.tc-text ul li {

margin-bottom: 10px;

font-size: 15px;

}
```

```
/* Scrollbar styling */
```

```
.tc-text::-webkit-scrollbar {

width: 6px;

}
```

```
.tc-text::-webkit-scrollbar-thumb {

background-color: #5d00ff;

border-radius: 3px;

}
```

```
/* Accept Button */
```

```
#acceptBtn {
```

```
display: block;

margin: 20px auto 0 auto;

padding: 12px 25px;

background: linear-gradient(135deg, #7803ff, #000000);

border: none;

border-radius: 50px;

color: white;

font-size: 16px;

font-weight: 600;

cursor: pointer;

transition: all 0.3s ease;

}
```

```
#acceptBtn:hover {

    transform: scale(1.05);

    box-shadow: 0 8px 20px rgba(124, 58, 237, 0.4);

}
```

```
/* Animation */

@keyframes fadeInUp {

    0% {

        opacity: 0;
```

```

        transform: translateY(30px);
    }

    100% {
        opacity: 1;
        transform: translateY(0);
    }
}

/* Optional: smooth fade for modal background */

.modal.show {
    opacity: 1;
    pointer-events: auto;
}

.modal.hide {
    opacity: 0;
    pointer-events: none;
}

body.modal-open .chat-wrapper,
body.modal-open .input-area,
body.modal-open .srm-header {

```

```
    pointer-events: none;

    user-select: none;

    opacity: 0.3;
}
```

```
/* But allow admin panel inputs to be clickable */
```

```
body.modal-open .admin-panel,
body.modal-open .admin-panel input,
body.modal-open .admin-panel button {

    pointer-events: auto;
}
```

```
.side-text {

    position: fixed;    /* stays in viewport */

    top: 50%;          /* vertically centered at start */

    left: 40px;

    color: #fafafa;

    font-family: 'Poppins', sans-serif;

    z-index: 9997;      /* behind header */

    opacity: 1;

    transform: translateY(-50%); /* offset to truly center */

    transition: opacity 0.2s ease, transform 0.2s ease;
```

```
}
```

```
.side-text.hidden {  
  opacity: 0;  
  transform: translateY(-20px); /* slightly move up */  
}
```

```
.side-text h2 {  
  font-size: 2rem;  
  font-weight: 700;  
  margin-bottom: 10px;  
}
```

```
.side-text p {  
  font-size: 1.1rem;  
  opacity: 0.85;  
}
```

```
body, html {  
  height: 100%;  
}
```

```
.disclaimer {  
  
  font-size: 0.5px;  
  
  color: #ff4d4d; /* red color for warning */  
  
  margin-top: 5px;  
  
}
```

```
.feedback {  
  
  margin-top: 8px;  
  
  display: flex;  
  
  gap: 8px;  
  
  font-size: 1.2rem;  
  
  cursor: pointer;  
  
}
```

```
.feedback span.thumb:hover {  
  
  transform: scale(1.2);  
  
  transition: transform 0.2s ease;  
  
}
```

```
.feedback-thanks {  
  
  font-size: 0.9rem;
```

```
    color: #7c3aed;

    font-weight: 500;
}
```

```
.shortcut-container {

    display: flex;

    justify-content: center;

    margin-top: 10px;
}
```

```
/* -----

    Floating "Open Portal" Button

----- */
```

```
.shortcut-btn {

    position: fixed;

    top: 20px;

    right: 25px;

    background: linear-gradient(90deg, #000000, #8400ff);

    color: white;

    border: none;

    padding: 10px 18px;

    border-radius: 10px;
```



```

font-size: 15px;

font-weight: 600;

cursor: pointer;

box-shadow: 0 4px 10px rgba(0,0,0,0.15);

transition: all 0.3s ease;

z-index: 10000; /* stays above everything */
}

.shortcut-btn:hover {

transform: scale(1.05);

background: linear-gradient(90deg, #8200fb, #000000);
}

/* ----- Admin Panel Styles ----- */

.admin-panel {

position: fixed;

bottom: 20px;

right: 500px;

left: 20px;

background: linear-gradient(45deg, #6a00ff, #000000);

backdrop-filter: blur(12px);

```

```
padding: 25px;

border-radius: 15px;

box-shadow: 0 0 25px rgba(106, 0, 255, 0.3);

width: 600px;

color: #5d00ff;

font-family: 'Poppins', sans-serif;

z-index: 10000;


/* animation setup */

opacity: 0;

transform: translateY(50px);

transition: transform 0.6s ease, opacity 0.6s ease;

}


/* when visible */

.admin-panel.visible {

opacity: 1;

transform: translateY(0);

animation: popIn 0.6s ease forwards;

}
```

```
.admin-panel h2 {  
    text-align: center;  
    margin-bottom: 15px;  
    font-weight: 600;  
    text-shadow: 0 2px 5px rgba(0,0,0,0.3);  
}
```

```
@keyframes popIn {  
    0% {  
        transform: translateY(50px) scale(0.9);  
        opacity: 0;  
    }  
    60% {  
        transform: translateY(-5px) scale(1.05);  
        opacity: 1;  
    }  
    80% {  
        transform: translateY(0px) scale(0.98);  
    }  
    100% {  
        transform: translateY(0) scale(1);  
    }  
}
```

```
}
```

```
.password-container {  
  display: flex;  
  gap: 10px;  
  margin-bottom: 10px;  
}
```

```
.password-container input {  
  flex: 1;  
  padding: 8px 12px;  
  border-radius: 8px;  
  border: none;  
  outline: none;  
  font-size: 14px;  
  background: rgba(255, 255, 255, 0.2);  
  color: #ffffff;  
}
```

```
.password-container input::placeholder {  
  color: #ffffff;  
}
```

```
.password-container button {  
  
  padding: 8px 12px;  
  
  border-radius: 8px;  
  
  border: none;  
  
  background: linear-gradient(45deg, #7500fa, #000000);  
  
  color: #fff;  
  
  cursor: pointer;  
  
  transition: all 0.3s ease;  
  
}
```

```
.password-container button:hover {  
  
  transform: scale(1.05);  
  
  box-shadow: 0 4px 12px rgba(0,0,0,0.3);  
  
}
```

```
#matrixCanvas {  
  
  position: fixed;  
  
  top: 0;  
  
  left: 0;  
  
  width: 100%;  
  
  height: 100%;
```

```
z-index: 9998; /* below floating images but above page content if needed */  
  
pointer-events: none;  
  
display: none; /* hidden by default */  
  
}
```

```
.matrix {  
  
background: rgba(0, 0, 0, 0.8);  
  
color: #00ff00;  
  
font-family: 'Courier New', monospace;  
  
padding: 10px 15px;  
  
margin: 8px 0;  
  
border-radius: 8px;  
  
text-shadow: 0 0 5px #00ff00, 0 0 10px #00ff00, 0 0 20px #00ff00;  
  
box-shadow: 0 0 10px #00ff00 inset;  
  
opacity: 0;  
  
transform: translateY(10px);  
  
transition: opacity 0.8s ease, transform 0.8s ease, box-shadow 0.8s ease;  
  
}
```

```
/* Fade-in animation */
```

```
.matrix.show {  
  
opacity: 1;
```

```
transform: translateY(0);  
}
```

```
@keyframes matrixPulse {  
  0%, 100% {  
    text-shadow: 0 0 8px #00ff00, 0 0 15px #00ff00;  
  }  
  50% {  
    text-shadow: 0 0 20px #00ff00, 0 0 30px #00ff00;  
  }  
}
```

```
.matrix.pulse {  
  animation: matrixPulse 1.5s infinite alternate;  
}
```

```
#rightPanel {  
  position: fixed;  
  top: 90px;  
  right: 20px;  
  width: 400px;      /* smaller width */  
  max-height: 100px; /* smaller height */
```

```
padding: 10px;

background: linear-gradient(90deg, #8400ff, #000000);

border-radius: 8px;

box-shadow: 0 2px 8px rgba(0,0,0,0.2);

z-index: 10000;

overflow: hidden;    /* hide overflow outside the panel */

}
```

```
#rightPanel h3 {

margin-top: 0;

font-size: 16px;

color: #fff;

text-align: center;

}
```

```
#alertsSidebar {

width: 100%;

height: 60px;    /* scrollable area height */

overflow-y: scroll;    /* enable vertical scroll */

background: #767676;

border-radius: 6px;

padding: 8px;
```



```
    box-shadow: inset 0 2px 6px rgba(0,0,0,0.1);  
    position: relative;  
}
```

```
.alertsWrapper {  
    display: flex;  
    flex-direction: column;  
    position: absolute;  
    top: 0;  
    width: 100%;  
}
```

```
.alert-item.urgent {  
    border-left: 5px solid red;  
    background: #ffe5e5;  
}
```

```
.alert-item {  
    opacity: 0;  
    transform: translateY(-10px);  
    animation: fadeInAlert 0.5s forwards;  
}
```

```
@keyframes fadeInAlert {  
  to {  
    opacity: 1;  
    transform: translateY(0);  
  }  
}
```

```
/* Alerts Sidebar Styling */
```

```
#alertsSidebar {  
  max-height: 350px;  
  overflow-y: auto;  
  padding: 10px;  
  border-radius: 8px;  
  background: #f9f9f9;  
  color: #333;  
  box-shadow: inset 0 2px 4px rgba(0,0,0,0.05);  
}
```

```
.announcement {  
  background: #fff;
```

```
padding: 8px 10px;

margin-bottom: 8px;

border-radius: 6px;

box-shadow: 0 2px 4px rgba(0,0,0,0.05);

}
```

```
.announcement h4 {

margin: 0 0 3px 0;

font-size: 1rem;

}
```

```
.announcement p {

margin: 3px 0;

font-size: 0.85rem;

}
```

```
.announcement small {

color: gray;

font-size: 0.75rem;

}
```

```
/* ----- Admin Panel ----- */
```

```

.admin-panel {

    background: rgba(255, 255, 255, 0.1);

    color: #fff;

    border-radius: 16px;

    box-shadow: 0 0 25px rgba(0, 255, 255, 0.15);

    margin: 40px auto;

    padding: 20px;

    width: 40%;

    max-width: 1200px;

    text-align: center;

    transition: all 0.4s ease;

    overflow: hidden;

    max-height: 200px; /* collapsed height */

}

/* Expanded state (after password verified) */

.admin-panel.expanded {

    max-height: 80vh; /* takes 80% of viewport height */

    overflow-y: auto;

    padding-bottom: 40px;

}

```

```
/* Scrollbar styling */

.admin-panel::-webkit-scrollbar {

    width: 8px;

}

.admin-panel::-webkit-scrollbar-thumb {

    background: rgba(0, 255, 255, 0.4);

    border-radius: 8px;

}

.admin-panel::-webkit-scrollbar-thumb:hover {

    background: rgb(255, 255, 255);

}


.admin-panel h2 {

    font-size: 1.8em;

    color: #00ffff;

    margin-bottom: 20px;

}


/* Password section */

.password-container {

    display: flex;

    justify-content: center;
```

```

    gap: 10px;

    margin-bottom: 20px;
}

.password-container input {

    padding: 10px;

    border-radius: 8px;

    border: 1px solid #ffffff;

    background: transparent;

    color: #fff;
}

.password-container button {

    background: linear-gradient(45deg, #6200ff, #000000);

    border: none;

    padding: 10px 16px;

    border-radius: 8px;

    color: #ffffff;

    font-weight: 600;

    cursor: pointer;

    transition: all 0.3s ease;
}

.password-container button:hover {

    transform: scale(1.05);
}

```

```
}
```

```
/* Status text */
```

```
.status-text {
```

```
  color: #00ffff;
```

```
  font-size: 0.9em;
```

```
  margin-top: 10px;
```

```
}
```

```
/* ----- Section Layout ----- */
```

```
.admin-sections {
```

```
  display: grid;
```

```
  grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
```

```
  gap: 25px;
```

```
  margin-top: 20px;
```

```
  text-align: left;
```

```
}
```

```
/* Each card/section */
```

```
.upload-section {
```

```
  background: rgba(255, 255, 255, 0.05);
```

```
  border: 1px solid rgba(0, 255, 255, 0.2);
```

```
border-radius: 12px;

padding: 20px;

transition: 0.3s;

}

.upload-section:hover {

transform: translateY(-4px);

border-color: #00ffff;

}
```

```
/* File upload buttons */
```

```
.file-upload {

display: flex;

flex-direction: column;

align-items: start;

gap: 10px;

margin-bottom: 10px;

}
```

```
.custom-file-label {

background: linear-gradient(45deg, #0077ff, #00ffff);

padding: 8px 12px;

border-radius: 8px;

cursor: pointer;
```



```

    color: #000;

    font-weight: 600;

    display: inline-block;
}

input[type="file"] {

    display: none;
}


/* Generic buttons */

.upload-section button {

    background: linear-gradient(45deg, #00ffff, #0077ff);

    border: none;

    color: #000;

    padding: 10px 15px;

    border-radius: 8px;

    font-weight: 600;

    cursor: pointer;

    margin-top: 10px;
}

.upload-section button:hover {

    transform: scale(1.05);
}

```

```

/* ----- News Section ----- */

#newsUploadDiv {

    margin-top: 10px;

    position: relative;    /* ensure it sits above overlays */

    z-index: 1001;        /* higher than modal background but below modal if needed */

}


.news-flex-container {

    display: flex;

    gap: 20px;

    flex-wrap: wrap;

    pointer-events: auto;    /* allow clicks inside */

}


.news-add-column,

.news-delete-column {

    flex: 1;

    min-width: 280px;

    background: rgba(0, 0, 0, 0.05);

    border-radius: 12px;

    padding: 20px;

```

```

position: relative;    /* relative for z-index */

z-index: 1002;        /* above overlay layers */

pointer-events: auto;  /* ensure clicks register */
}

/* Inputs and textarea */

.news-input-row input,
.news-input-row textarea,
.news-add-column input[type="date"],
.news-add-column input,
.news-add-column textarea,
.news-delete-column input,
.news-delete-column textarea {
    width: 100%;

    background: rgba(0, 0, 0, 0.15); /* slightly visible background */

    border: 1px solid #00ffff;

    color: #fff;

    border-radius: 8px;

    padding: 8px;

    margin-bottom: 10px;

    pointer-events: auto;    /* allow typing */
}

```

```
/* Ensure placeholder is visible */

.news-input-row input::placeholder,
.news-input-row textarea::placeholder,
.news-add-column input::placeholder,
.news-add-column textarea::placeholder {

    color: #ddd;

    opacity: 1;

}


/* Buttons */

.news-add-column button,
.news-delete-column button {

    width: 100%;

    background: linear-gradient(45deg, #00ffff, #0077ff);

    border: none;

    padding: 10px;

    border-radius: 8px;

    color: #000;

    font-weight: 600;

    cursor: pointer;

    transition: all 0.3s;
```

```
    pointer-events: auto;    /* allow clicks */  
}
```

```
.news-add-column button:hover,  
.news-delete-column button:hover {  
    transform: scale(1.05);  
}
```

```
/* Ensure admin panel allows all nested elements */  
body.modal-open .admin-panel * {  
    pointer-events: auto !important;  
}
```

```
.news-add-column input[type="date"] {  
    pointer-events: auto !important; /* allow interaction */  
    position: relative;              /* make it above any overlapping layers */  
    z-index: 1003;                   /* higher than modal overlays */  
    background: rgba(0,0,0,0.15); /* visible background */  
    color: #fff;  
    border: 1px solid #00ffff;  
    border-radius: 8px;  
    padding: 8px;
```

```
}
```

```
/* Style the delete news dropdown to match other inputs */
```

```
#delete-news-select {
```

```
width: 100%;
```

```
padding: 10px 12px; /* similar to other inputs */
```

```
border-radius: 8px;
```

```
border: 1px solid #00ffff; /* cyan border like other inputs */
```

```
background: rgba(0, 0, 0, 0.15); /* match other input backgrounds */
```

```
color: #fff; /* white text */
```

```
font-size: 1rem;
```

```
outline: none;
```

```
appearance: none; /* remove default arrow for consistency */
```

```
cursor: pointer;
```

```
transition: all 0.3s ease;
```

```
}
```

```
/* Hover & focus effects */
```

```
#delete-news-select:hover,
```

```
#delete-news-select:focus {
```

```
border-color: #ffffff; /* highlight border on hover/focus */
```

```
box-shadow: 0 0 6px rgba(0, 255, 255, 0.5);
```

```
}
```

```
/* Optional: custom arrow */
```

```
#delete-news-select::after {
```

```
    content: '▼';          /* simple dropdown arrow */
```

```
    position: absolute;
```

```
    right: 12px;
```

```
    top: 50%;
```

```
    transform: translateY(-50%);
```

```
    pointer-events: none;
```

```
    color: #fff;
```

```
}
```

### 11.3.3 JAVASCRIPT

```
// -----
```

```
// DOM Elements
```

```
// -----
```

```
const chatBox = document.getElementById("chat-box");
```

```
const userInput = document.getElementById("user-input");
```

```
const sendBtn = document.getElementById("send-btn");
```

```
const timetable = document.getElementById("exam-timetable");
```

```

let isTyping = false;

let typingTimeout;

let controller;

// -----

// Mouse Parallax Effect

// -----

document.addEventListener("mousemove", (e) => {

  const mouseEffect = document.querySelector(".mouse-effect");

  mouseEffect.style.background = `radial-gradient(circle at ${e.clientX}px
  ${e.clientY}px, rgba(113, 15, 251, 0.34), transparent 25%)`;

});

// -----

// Terms & Conditions Modal

// -----

window.onload = () => {

  const modal = document.getElementById("tcModal");

  const acceptBtn = document.getElementById("acceptBtn");

  document.body.classList.add("modal-open");

  modal.style.display = "flex";

  acceptBtn.onclick = () => {

```



```

    modal.style.display = "none";

    document.body.classList.remove("modal-open");

};

modal.onclick = (e) => {
    if (e.target === modal) e.stopPropagation();
};

};

// -----
// Render Timetable
// -----

function renderTimetable(timetable) {

    const tbody = document.querySelector("#timetable-table tbody");

    tbody.innerHTML = ""; // clear old rows

    timetable.forEach(row => {

        const tr = document.createElement("tr");

        tr.innerHTML = `

            <td>${row.subject}</td>

            <td>${row.code}</td>

```

```

        <td>${row.department}</td>

        <td>${row.year}</td>

        <td>${row.date}</td>

        <td>${row.time}</td>

    `;

    tbody.appendChild(tr);

});

const tableDiv = document.getElementById("exam-timetable");

tableDiv.style.display = "block";          // show table

tableDiv.style.opacity = 0;                // start invisible

setTimeout(() => tableDiv.style.opacity = 1, 50); // fade in

tableDiv.scrollIntoView({ behavior: "smooth" });

}

// -----

// Feedback Buttons

// -----

function addFeedbackButtons(msgDiv, assistantMessage) {

    const feedbackDiv = document.createElement("div");

    feedbackDiv.classList.add("feedback");

    feedbackDiv.innerHTML = `

```

```

<span class="thumb" data-feedback="up">👍</span>

<span class="thumb" data-feedback="down">👎</span>

`;

msgDiv.appendChild(feedbackDiv);

feedbackDiv.querySelectorAll(".thumb").forEach(thumb => {

  thumb.addEventListener("click", () => {

    const feedback = thumb.dataset.feedback;

    feedbackDiv.innerHTML = `<span class="feedback-thanks">Thanks! ${feedback
    === "up" ? "😊" : "😞"}</span>`;

    fetch("/feedback", {

      method: "POST",

      headers: { "Content-Type": "application/json" },

      body: JSON.stringify({ message: assistantMessage, feedback })

    });

  });

});

}

// -----

// Chat Functions

// -----

```

```

function appendMessage(message, sender) {

  const msgDiv = document.createElement("div");

  msgDiv.classList.add("message", sender === "assistant" ? "assistant-message" :
"user-message");

  msgDiv.innerHTML = `<p>${message}</p>`;

  chatBox.appendChild(msgDiv);

  chatBox.scrollTop = chatBox.scrollHeight;

  if (sender === "assistant") addFeedbackButtons(msgDiv, message);

}

```

```

function typeEffect(text) {

  return new Promise(resolve => {

    const msgDiv = document.createElement("div");

    msgDiv.classList.add("message", "assistant-message");

    chatBox.appendChild(msgDiv);

    let i = 0;

    const speed = 20;

    isTyping = true;

    function type() {

      if (i < text.length && isTyping) {

```

```

    msgDiv.textContent = text.substring(0, i + 1);

    i++;

    chatBox.scrollTop = chatBox.scrollHeight;

    typingTimeout = setTimeout(type, speed);

} else {

    isTyping = false;

    sendBtn.textContent = "➤";

    addFeedbackButtons(msgDiv, text);

    resolve();

}

}

type();

});

}

```

```

function showTyping() {

    const typing = document.createElement("div");

    typing.classList.add("message", "assistant-message");

    typing.id = "typing";

    typing.textContent = "Typing...";

    chatBox.appendChild(typing);

    chatBox.scrollTop = chatBox.scrollHeight;

```

```
}
```

```
function removeTyping() {  
  const typing = document.getElementById("typing");  
  if (typing) typing.remove();  
}
```

```
function stopTyping() {  
  if (isTyping) {  
    clearTimeout(typingTimeout);  
    if (controller) controller.abort();  
    isTyping = false;  
    sendBtn.textContent = "➤";  
    removeTyping();  
    appendMessage("⚠ Response stopped by user.", "assistant");  
  }  
}
```

```
// -----
```

```
// Send Message
```

```
// -----
```

```
async function sendMessage() {
```

```

const question = userInput.value.trim();

if (!question) return;

// Portal shortcut

if (question.toLowerCase().includes("open portal")) {

window.open("https://sp.srmist.edu.in/srmiststudentportal/students/loginManager/youLogin.jsp", "_blank");

    appendMessage("✅ Opening SRM Student Portal...", "assistant");

    userInput.value = "";

    return;

}

//Team Shortcut

if (question.toLowerCase().includes("created you")) {

    window.open("/team", "_self");

    appendMessage("Meet the team", "assistant");

    userInput.value = "";

    return;

}

if (question.toLowerCase().includes("your code")) {

    appendMatrixMessage("🟩 Well, well... look who's curious. Kudos, genius.",
"assistant");

    startMatrixEffect();

    userInput.value = "";

```

```

    return;
}

if (!isTyping) {
    appendMessage(question, "user");
    userInput.value = "";
    isTyping = true;
    sendBtn.textContent = "📄";
    showTyping();

    controller = new AbortController();
    const signal = controller.signal;

    try {
        const response = await fetch("/ask", {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ question }),
            signal
        });

        const data = await response.json(); // ONLY ONE CALL
        removeTyping();
    }
}

```



```

// Render timetable if exists

if (data.timetable && data.timetable.length > 0) {

    renderTimetable(data.timetable);

} else {

    await typeEffect(data.answer);

}

} catch (err) {

    removeTyping();

    appendMessage(err.name === "AbortError" ? "⚠️ Response stopped by user." : "⚠️
    Something went wrong.", "assistant");

    isTyping = false;

    sendBtn.textContent = "➤";

}

} else {

    stopTyping();

}

}

// -----

// Event Listeners

```

```

// -----

document.addEventListener("DOMContentLoaded", () => {

    const sendBtn = document.getElementById("send-btn");

    const userInput = document.getElementById("user-input");

    const portalBtn = document.getElementById("portalBtn");

    sendBtn.addEventListener("click", sendMessage);

    userInput.addEventListener("keydown", e => { if (e.key === "Enter") sendMessage();
});

    portalBtn.addEventListener("click", () => {

window.open("https://sp.srmist.edu.in/srmiststudentportal/students/loginManager/youLogin.jsp", "_blank");

    });

});

// Timetable Upload

document.getElementById("uploadTimetableBtn").addEventListener("click", async ()
=> {

    const fileInput = document.getElementById("timetableFile");

    const file = fileInput.files[0];

    if (!file) {

```

```
    alert("Please select a file first!");  
    return;  
}
```

```
const formData = new FormData();  
formData.append("file", file);
```

```
const res = await fetch("/upload_timetable", {  
  method: "POST",  
  body: formData  
});
```

```
const data = await res.json();  
document.getElementById("timetableStatus").textContent = data.message;  
});
```

// Seat Upload

```
document.getElementById("uploadSeatBtn").addEventListener("click", async () => {  
  const fileInput = document.getElementById("seatFile");  
  const file = fileInput.files[0];  
  if (!file) {  
    alert("Please select a file first!");  
  }  
});
```

```

    return;
}

const formData = new FormData();
formData.append("file", file);

const res = await fetch("/upload_seats", {
    method: "POST",
    body: formData
});

const data = await res.json();
document.getElementById("seatStatus").textContent = data.message;
});

// -----
// -----
// Admin Upload Password Protection
// -----

const adminPassword = "SRMAdmin2025"; // change to your secret password

```

```

document.getElementById("verifyPasswordBtn").addEventListener("click", () => {

    const inputPass = document.getElementById("adminPassword").value.trim();

    const status = document.getElementById("passwordStatus");

    const timetableDiv = document.getElementById("timetableUploadDiv");

    const seatDiv = document.getElementById("seatUploadDiv");

    const newsUploadDiv = document.getElementById("newsUploadDiv");

    if (inputPass === adminPassword) {

        status.textContent = "✅ Password correct! You can now upload files.";

        status.style.color = "green";

        // Show upload divs

        [timetableDiv, seatDiv].forEach(div => {

            div.style.display = "block";

            div.style.opacity = 0;

            setTimeout(() => div.style.opacity = 1, 100);

            div.style.transition = "opacity 0.8s ease";

        });

        // Show Clear Data div

        const clearDiv = document.getElementById("clearDataDiv");

        clearDiv.style.display = "block";
    }

```

```

clearDiv.style.opacity = 0;

setTimeout(() => clearDiv.style.opacity = 1, 100);

clearDiv.style.transition = "opacity 0.8s ease";


newsUploadDiv.style.display = "block";

newsUploadDiv.style.opacity = 0;

setTimeout(() => newsUploadDiv.style.opacity = 1, 100);

newsUploadDiv.style.transition = "opacity 0.8s ease";

} else {

status.textContent = "✗ Incorrect password. Access denied!";

status.style.color = "red";

timetableDiv.style.display = "none";

seatDiv.style.display = "none";


const clearDiv = document.getElementById("clearDataDiv");

clearDiv.style.display = "none";

}

});


// Upload Timetable CSV

document.getElementById("uploadTimetableBtn").addEventListener("click", () => {

const file = document.getElementById("timetableFile").files[0];

```

```

const status = document.getElementById("timetableStatus");

if (!file) return status.textContent = "Please select a file!";


const formData = new FormData();

formData.append("file", file);

formData.append("password", document.getElementById("adminPassword").value);


fetch("/upload_timetable", {

  method: "POST",

  body: formData

}).then(res => res.json())

  .then(data => status.textContent = data.message)

  .catch(err => status.textContent = "Upload failed!");

});


// Upload Seat CSV

document.getElementById("uploadSeatBtn").addEventListener("click", () => {

  const file = document.getElementById("seatFile").files[0];

  const status = document.getElementById("seatStatus");

  if (!file) return status.textContent = "Please select a file!";


  const formData = new FormData();

```

```
formData.append("file", file);

formData.append("password", document.getElementById("adminPassword").value);
```

```
fetch("/upload_seat", {
  method: "POST",
  body: formData
}).then(res => res.json())
  .then(data => status.textContent = data.message)
  .catch(err => status.textContent = "Upload failed!");
});
```

```
// Admin Panel Dynamic Appearance
```

```
const adminPanel = document.getElementById("adminPanel");
```

```
window.addEventListener("scroll", function() {
  const scrollTopPosition = window.scrollY + window.innerHeight;
  const pageHeight = document.documentElement.scrollHeight;
```

```
if (scrollTopPosition >= pageHeight - 100) {
  adminPanel.classList.add("visible");
} else {
  adminPanel.classList.remove("visible");
```



```

    }
  });

  // Display chosen file name

  document.getElementById("timetableFile").addEventListener("change", (e) => {

    document.getElementById("timetableFileName").textContent =

      e.target.files[0] ? e.target.files[0].name : "No file chosen";

  });

  document.getElementById("seatFile").addEventListener("change", (e) => {

    document.getElementById("seatFileName").textContent =

      e.target.files[0] ? e.target.files[0].name : "No file chosen";

  });

  window.addEventListener("scroll", function() {

    const adminPanel = document.querySelector(".adminPanel");

    const scrollTopPosition = window.scrollY + window.innerHeight;

    const pageHeight = document.documentElement.scrollHeight;

    // When near bottom → show it

    if (scrollTopPosition >= pageHeight - 100) {

      adminPanel.classList.add("visible");
    }
  });

```

```

    }

    // When scrolled up → hide it

    else {

        adminPanel.classList.remove("visible");

    }

});

// -----

// Side Text Scroll Fade (Hide down, Show up)

// -----

const sideText = document.querySelector(".side-text");

let lastScrollTop = 0;

window.addEventListener("scroll", () => {

    const currentScroll = window.scrollY || document.documentElement.scrollTop;

    if (currentScroll > lastScrollTop) {

        // Scrolling down → hide

        sideText.style.opacity = "0";

        sideText.style.transform = "translateY(-60%)";

    } else {

        // Scrolling up → show

        sideText.style.opacity = "1";
    }
});

```

```

    sideText.style.transform = "translateY(-50%)";
  }

  sideText.style.transition = "opacity 0.4s ease, transform 0.4s ease";
  lastScrollTop = currentScroll <= 0 ? 0 : currentScroll;
});

function clearData(type) {
  if (!confirm(`Are you sure you want to delete all ${type} data?`)) return;

  const formData = new FormData();
  formData.append("type", type);

  fetch("/clear_data", {
    method: "POST",
    body: formData
  })
  .then(response => response.json())
  .then(data => {
    document.getElementById("clearStatus").innerText = data.message;
  })

```

```

        .catch(err => {

            document.getElementById("clearStatus").innerText = "Error clearing data: " + err;

        });
    }

```

```

document.addEventListener("DOMContentLoaded", () => {

    document.getElementById("teamBtn").addEventListener("click", () => {

        window.location.href = "team.html";

    });

});

```

```

document.addEventListener("DOMContentLoaded", () => {

    const tcModal = document.getElementById("tcModal");

    const acceptBtn = document.getElementById("acceptBtn");

    const teamBtn = document.getElementById("teamBtn");

    // Show modal / button based on localStorage

    if (localStorage.getItem("tcAccepted") === "true") {

        tcModal.style.display = "none";

        teamBtn.style.display = "block";

    } else {

```

```

    tcModal.style.display = "flex"; // flex for centering modal

    teamBtn.style.display = "none";
}

// Accept T&C
acceptBtn.addEventListener("click", () => {
    localStorage.setItem("tcAccepted", "true");
    tcModal.style.display = "none";
    teamBtn.style.display = "block";
});

// Navigate to team page
teamBtn.addEventListener("click", () => {
    window.location.href = "/team";
});
});

function startMatrixEffect() {
    const canvas = document.getElementById("matrixCanvas");
    canvas.style.display = "block";
    const ctx = canvas.getContext("2d");

```

```

canvas.width = window.innerWidth;

canvas.height = window.innerHeight;


const letters =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*(";

const fontSize = 18;

const columns = Math.floor(canvas.width / fontSize);

const drops = Array(columns).fill(0);


function draw() {

    ctx.fillStyle = "rgba(0, 0, 0, 0.05)";

    ctx.fillRect(0, 0, canvas.width, canvas.height);


    ctx.fillStyle = "#00ff00";

    ctx.font = fontSize + "px monospace";


    for (let i = 0; i < drops.length; i++) {

        const text = letters[Math.floor(Math.random() * letters.length)];

        ctx.fillText(text, i * fontSize, drops[i] * fontSize);


        if (drops[i] * fontSize > canvas.height && Math.random() > 0.975) {

            drops[i] = 0;

        }
    }
}

```

```

        drops[i]++;
    }

    requestAnimationFrame(draw);
}

draw();

// Stop effect after 10 seconds
setTimeout(() => {
    canvas.style.display = "none";
}, 10000);
}

function appendMatrixMessage(msg) {
    const chatBox = document.getElementById("chat-box");
    const message = document.createElement("div");
    message.classList.add("message", "matrix");
    message.textContent = msg;
    chatBox.appendChild(message);

    // Trigger fade-in

```

```

setTimeout(() => {
    message.classList.add("show", "pulse");
}, 50);

// Optional: Fade-out after 4 seconds
setTimeout(() => {
    message.classList.remove("show");
    setTimeout(() => message.remove(), 800); // remove after fade-out
}, 10000);

chatBox.scrollTop = chatBox.scrollHeight;
}

document.getElementById("verifyPasswordBtn").addEventListener("click", () => {
    const panel = document.querySelector(".admin-panel");
    panel.classList.add("expanded");
});

```

### 11.3.4 SRM CHATBOT

```

from flask import Flask, render_template, request, jsonify
import json

```



```

import torch

import faiss

import numpy as np

from huggingface_hub import login

from langchain.docstore.document import Document

from sentence_transformers import SentenceTransformer

from transformers import AutoModelForCausalLM, AutoTokenizer,
BitsAndBytesConfig, pipeline

import pandas as pd

from datetime import datetime, timedelta

from difflib import SequenceMatcher

import string

import re

import pandas as pd

import nltk

from nltk.corpus import words

import re

from rapidfuzz import fuzz

import sqlite3

import dateparser

# -----

# Gibberish Detection

```

```
# -----

nltk.download('words')

english_words = set(words.words())

indian_names = {

    "kaarthik", "narain", "yazhini", "ananya", "sivakumar", "anand", "rajesh",

    "albert", "arjun", "priya", "deepak", "krishna", "lakshmi", "manoj", "rohini",

    "ganesh", "vignesh", "naveen", "ramesh", "arun", "anitha", "gopal", "sundar",

    "sathish", "bharathi", "saravanan", "kavya", "devi", "hari", "varun", "dinesh",

    "nandhini", "subash", "meena", "revathi", "gowtham", "karthikeyan", "anandhi",

    "vijay", "jagath", "ashwin", "guru",

    "surya", "Belina", "Anu", "Jeeva", "Sathya", "Kavin",

    "Sakthi", "Prakash", "Vimal", "Nithish", "Kavin", "Pradeep", "Sanjay", "Ravi", "Ajith", "Kumar", "Senthil", "Rajeshwari", "Lakshmanan", "Mohan", "Suresh", "Vijaya", "Kalyani", "Nandha", "Sangeetha", "Divya", "Pooja", "Rajini", "Sowmya", "Nithya", "Harish", "Vasanth", "Ashok", "Kishore", "Raghav", "Yash", "Aditya", "Aravind", "Chitra", "Bhavani", "Janani", "Keerti", "Latha", "Madhavi", "Nisha", "Pallavi", "Radha", "Sahana", "Tamilarasi", "Usha", "Vijayalakshmi"

}

indian_names = {name.lower() for name in indian_names}

def is_gibberish_word(word):

    word = word.lower()

    # Ignore very short words
```

```

if len(word) <= 1:

    return False

# Mostly symbols/numbers → gibberish

letter_count = sum(c.isalpha() for c in word)

if letter_count / max(len(word), 1) < 0.5:

    return True

# If in dictionary → not gibberish

if word in english_words:

    return False

if word.lower() in indian_names:

    return False

# Not in dictionary → possible gibberish

return True

```

```

def is_gibberish(text, threshold=0.5):

    words_list = re.findall(r'\b\w+\b', text)

    if not words_list:

        return True

    gibberish_count = sum(is_gibberish_word(w) for w in words_list)

    ratio = gibberish_count / len(words_list)

    return ratio > threshold

```

```

# -----
# Timetable Data Placeholder
# -----

timetable = None # Will be loaded dynamically after admin uploads a file

def answer_from_timetable(query):
    global timetable

    if timetable is None or timetable.empty:
        return "⚠ Timetable not uploaded yet. Please ask admin to upload it."

    q = query.lower()

    # Identify department
    departments = timetable['department'].str.lower().unique()
    dept = next((d for d in departments if d in q), None)

    # Identify year
    year_match = re.search(r"\b(d)(?:st|nd|rd|th)? year\b", q)
    year = int(year_match.group(1)) if year_match else None

```

```

# Identify subject or code

for _, row in timetable.iterrows():

    if row["subject"].lower() in q.lower() or row["code"].lower() in q.lower() or
q.lower() in row["subject"].lower():

        return (f'{row["subject"]} exam for {row["department"]} (Year {row["year"]}) "

                f'is on {row["date"]} from {row["time"]}."')

# If department + year are mentioned → show all their exams

if dept and year:

    df = timetable[

        (timetable["department"].str.lower() == dept) &

        (timetable["year"] == year)

    ]

    if not df.empty:

        result = "\n".join([

            f'■ {r["subject"]} ({r["code"]}) — {r["date"]} | {r["time"]}'

            for _, r in df.iterrows()

        ])

        return f'Here's the exam schedule for {dept.upper()} - Year {year}:\n\n{result}'

    return "I couldn't find that in the timetable."

# Load embedding model (only once)

```

```

embedder = SentenceTransformer('all-MiniLM-L6-v2')

from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
import sqlite3
from datetime import datetime
import numpy as np

from rapidfuzz import fuzz
import re
import sqlite3
from datetime import datetime

# Add your acronym mapping here
SUBJECT_ACRONYMS = {
    "ood": "object oriented analysis and design",
    "ai": "artificial intelligence",
    "cas": "cognitive analytical skills",
    "eh": "ethical hacking",
    "fsd": "full stack development"
}

```

```

def check_timetable(user_message):
    """
    Returns timetable info based on user query.

    Handles:
    - Full subject match
    - Partial/fuzzy match
    - Acronym mapping
    - Department + year queries
    - Department-only queries
    """
    q = user_message.lower().strip()

    # Step 0: Replace acronyms dynamically
    for acr, full in SUBJECT_ACRONYMS.items():
        q = re.sub(rf"\b{acr}\b", full, q)

    # Step 1: Fetch all timetable rows
    conn = sqlite3.connect(DB_FILE)
    c = conn.cursor()
    c.execute("SELECT department, year, date, time, subject, code FROM timetable")
    rows = c.fetchall()

```

```

conn.close()

# Step 2: Exact subject match

for r in rows:

    if q == r[4].lower():

        exam_date = datetime.strptime(r[2], "%Y-%m-%d").strftime("%d %b %Y")

        return f"{r[4]} exam for {r[0]} (Year {r[1]}) is on {exam_date} from {r[3]}."

# Step 3: Partial/fuzzy match on subject or code

best_score, best_row = 0, None

for r in rows:

    score_subj = fuzz.partial_ratio(q, r[4].lower())

    score_code = fuzz.partial_ratio(q, r[5].lower())

    max_score = max(score_subj, score_code)

    if max_score > best_score:

        best_score = max_score

        best_row = r

if best_score > 70:

    r = best_row

    exam_date = datetime.strptime(r[2], "%Y-%m-%d").strftime("%d %b %Y")

    return f"{r[4]} exam for {r[0]} (Year {r[1]}) is on {exam_date} from {r[3]}."

```



```

# Step 4: Department + year query

dept_match = None

year_match = None

departments = sorted(set([r[0].lower() for r in rows]), key=lambda x: -len(x))

# Try exact word match first

for d in departments:

    if f" {d} " in f" {q} ":

        dept_match = d

        break

# Fallback: partial match

if not dept_match:

    for d in departments:

        if d in q:

            dept_match = d

            break

# Year extraction

ym = re.search(r"\b(\d)(?:st|nd|rd|th)? year\b", q)

if ym:

    year_match = int(ym.group(1))

```

```

# Dept + year

if dept_match and year_match:

    matching_rows = [r for r in rows if r[0].lower() == dept_match and r[1] ==
year_match]

    if matching_rows:

        # Try fuzzy match within filtered rows first

        best_score, best_row = 0, None

        for r in matching_rows:

            score_subj = fuzz.partial_ratio(q, r[4].lower())

            score_code = fuzz.partial_ratio(q, r[5].lower())

            max_score = max(score_subj, score_code)

            if max_score > best_score:

                best_score = max_score

                best_row = r

        if best_row and best_score >= 50:

            r = best_row

            exam_date = datetime.strptime(r[2], "%Y-%m-%d").strftime("%d %b %Y")

            return f"{r[4]} exam for {r[0]} (Year {r[1]}) is on {exam_date} from {r[3]}."

# Fallback: list all exams for dept+year

result = "\n".join([

```

```

        f" {r[4]} ({r[5]}) — {datetime.strptime(r[2], '%Y-%m-%d').strftime('%d
        %b %Y')} | {r[3]}"

```

```

        for r in matching_rows

```

```

    ])

```

```

    return f"Here's the exam schedule for {dept_match.upper()} - Year
    {year_match}:\n\n{result}"

```

# Step 5: Department-only (no year)

```

if dept_match:

```

```

    matching_rows = [r for r in rows if dept_match in r[0].lower()]

```

```

    if matching_rows:

```

```

        result = "\n".join([

```

```

            f" {r[4]} ({r[5]}) — {datetime.strptime(r[2], '%Y-%m-%d').strftime('%d
            %b %Y')} | {r[3]}"

```

```

            for r in matching_rows

```

```

        ])

```

```

        return f"Here's the exam schedule for {dept_match.upper()}:\n\n{result}"

```

```

    return "I couldn't find that in the timetable. Please check the subject name, code, or
    department/year."

```

```

# -----

```

```

# Seat Data Placeholder

# -----

seat_df = None # Will be loaded dynamically after admin uploads a file


def load_seat_data(excel_file):
    """
    Loads seat data dynamically when admin uploads a file.
    """
    try:
        df = pd.read_excel(excel_file)

        df['Date'] = pd.to_datetime(df['Date']).dt.date # Convert Excel dates to
datetime.date

        print(f"✅ Seat data loaded successfully with {len(df)} entries!")

        return df

    except Exception as e:

        print(f"⚠️ Error loading seat data: {e}")

        return pd.DataFrame(columns=["Register number", "Session", "Date", "Seat
Number"])


import re

import sqlite3

import dateparser

from datetime import datetime

```

```

# -----

# Flexible Session Mapping

# -----

SESSION_MAP = {

    "FN": ["FN", "forenoon", "morning"],

    "AN": ["AN", "afternoon", "evening"]

}


def parse_session(msg):

    """Return standard session code 'FN' or 'AN' based on flexible terms."""

    msg = msg.lower()

    for standard, variants in SESSION_MAP.items():

        for v in variants:

            if v.lower() in msg:

                return standard

    return None


def parse_date(msg):

    """Handles natural language dates using dateparser and outputs YYYY-MM-DD."""

    dt = dateparser.parse(msg, settings={'PREFER_DATES_FROM': 'future'})

    if dt:

```

```

        return dt.date().strftime('%Y-%m-%d')

# Fallback for MM/DD/YY or MM/DD/YYYY

date_match = re.search(r'(\d{1,2}/\d{1,2}/\d{2,4}|\d{4}-\d{2}-\d{2})', msg)

if date_match:

    try:

        return datetime.strptime(date_match.group(0),
            "%m/%d/%y").strftime('%Y-%m-%d')

    except:

        try:

            return datetime.strptime(date_match.group(0),
                "%m/%d/%Y").strftime('%Y-%m-%d')

        except:

            try:

                return datetime.strptime(date_match.group(0),
                    "%Y-%m-%d").strftime('%Y-%m-%d')

            except:

                return None

    return None

# -----

# Updated Seat Checker

# -----

DB_FILE = "srm_data.db"

```

```

def check_seat_from_message(user_message):
    """
    Check seat for a student from a user message.

    Handles:
    - Flexible session names (FN, AN, morning, afternoon)
    - Natural language dates
    - Standard register numbers
    """
    msg = user_message.strip()

    #1 Extract register number
    reg_match = re.search(r'RA\d{13}', msg.upper())
    reg_no = reg_match.group(0) if reg_match else None

    #2 Extract session
    session = parse_session(msg)

    #3 Extract date
    date = parse_date(msg)

    #4 Check missing info

```

```

missing = []

if not reg_no:

    missing.append("register number")

if not session:

    missing.append("session (FN/AN or morning/afternoon)")

if not date:

    missing.append("date (MM/DD/YYYY, YYYY-MM-DD, or natural language like
'tomorrow', 'next Friday')")

if missing:

    return "Please provide your " + ", ".join(missing) + " to check your seat."

```

#5 Query SQLite DB

```

try:

    conn = sqlite3.connect(DB_FILE)

    c = conn.cursor()

    c.execute("""

        SELECT seat_number FROM seats

        WHERE register_number = ? AND session = ? AND date = ?

        """, (reg_no, session, date))

    row = c.fetchone()

    conn.close()

except Exception as e:

```



```
return f"⚠ Database error: {e}"
```

```
# 6 Return seat info
```

```
if row:
```

```
    return f"📅 Seat for {reg_no} on {date} ({session}): **{row[0]}**"
```

```
else:
```

```
    return "No seat details found for the provided information."
```

```
# -----
```

```
# Authenticate Hugging Face
```

```
# -----
```

```
login("YOUR_HUGGINGFACE_TOKEN")
```

```
# -----
```

```
# Load JSON dataset for RAG
```

```
# -----
```

```
with open("srm_data.json", "r") as f:
```

```
    data = json.load(f)
```

```
docs = [Document(page_content=entry["completion"], metadata={"question":  
entry["prompt"]}) for entry in data]
```

```
documents = [d.page_content for d in docs]
```

```

# -----

# Embeddings + FAISS

# -----

embedder = SentenceTransformer("all-MiniLM-L6-v2")

doc_embeddings = embedder.encode(documents, convert_to_numpy=True)

dimension = doc_embeddings.shape[1]

index = faiss.IndexFlatL2(dimension)

index.add(doc_embeddings)


# -----

# Load Model

# -----

try:

    model_name = "mistralai/Mistral-7B-Instruct-v0.2"

    bnb_config = BitsAndBytesConfig(load_in_8bit=True)

    tokenizer = AutoTokenizer.from_pretrained(model_name, use_auth_token=True)

    model = AutoModelForCausalLM.from_pretrained(

        model_name,

        quantization_config=bnb_config,

        device_map="auto",

        dtype=torch.float16

    )

```

```

    generator = pipeline("text-generation", model=model, tokenizer=tokenizer,
device_map="auto", dtype=torch.float16)

except Exception as e:

    from transformers import AutoModelForSeq2SeqLM

    model_name = "google/flan-t5-base"

    tokenizer = AutoTokenizer.from_pretrained(model_name)

    model = AutoModelForSeq2SeqLM.from_pretrained(model_name)

    generator = pipeline("text2text-generation", model=model, tokenizer=tokenizer)


# -----

# Normalize text

# -----

def normalize(text):

    return text.lower().translate(str.maketrans("", "", string.punctuation)).strip()


# -----

# Temperature Fetcher

# -----


import requests


API_KEY = "7b3b0c3ccdf5f72ec472153dc330dc61"

CITY = "Chennai,IN"

```

```

def get_potheri_temperature():

    url =
    f"http://api.openweathermap.org/data/2.5/weather?q={CITY}&units=metric&appid={A
    PI_KEY}"

    response = requests.get(url)

    if response.status_code == 200:

        data = response.json()

        temp = data['main']['temp']

        description = data['weather'][0]['description']

        return f"The current temperature in SRM Potheri, Chennai is {temp}°C with
        {description}."

    else:

        return "Sorry, I couldn't fetch the weather for Potheri, Chennai."

# -----

# RAG Query

# -----

def rag_query(question, top_k=5):

    q_norm = normalize(question)

    # 1 Extract seat info

    reg_match = re.search(r'RA\d{13}', question.upper())

```

```

reg_no = reg_match.group(0) if reg_match else None

session = parse_session(question)

date = parse_date(question)


# If all present → check seat

if reg_no and session and date:

    return check_seat_from_message(question)


if is_gibberish(question):

    return "Hmm... I cannot understand that! Could you rephrase that?"


def is_timetable_query(q):

    # Year pattern

    year_match = re.search(r"\b(d)(?:st|nd|rd|th)? year\b", q.lower())

    # Keywords

    keywords = ["exam", "timetable", "schedule", "subject", "paper", "date", "when is"]

    keyword_match = any(k in q.lower() for k in keywords)

    # Department or subject code

    dept_match = any(d.lower() in q.lower() for d in timetable['department'].unique())
    if timetable is not None else False

    code_match = any(c.lower() in q.lower() for c in timetable['code'].unique()) if
    timetable is not None else False

```

```
return year_match or keyword_match or dept_match or code_match
```

```
if is_timetable_query(question):
```

```
    return check_timetable(question)
```

```
static_responses = {
```

```
    "hi": "Hi there 🙌! I'm your SRM AI Assistant. How can I help you today?",
```

```
    "hello": "Hello! 😊 I'm your friendly SRM AI Assistant — ready to assist you!",
```

```
    "hey": "Hey! 🙌 What would you like to know about SRM?",
```

```
    "who are you": "I am an AI Assistant 🤖 — built to help you explore SRMIST!",
```

```
    "what is your name": "I'm SRM AI Assistant, your smart campus guide ✨.",
```

```
    "About yourself": "I'm an AI Assistant for SRM, created by MCA Generative AI  
26 students Kaarthik Narain, Vijay, Jagath Ashwin, and Guru Surya under Dr.  
Sivakumar S's guidance.",
```

```
    "help": "Sure! I can assist you with information about SRMIST, exam timetables,  
seat allocations, and more. Just ask your question!",
```

```
    "thanks": "You're welcome! 😊 If you have more questions, feel free to ask",
```

```
    "bye": "Goodbye! 🙌 Have a great day ahead!",
```

```
    "good morning": "Good morning! ☀️ How can I assist you today?",
```

```
    "good afternoon": "Good afternoon! 🌞 What would you like to know about  
SRM?",
```

```
    "good evening": "Good evening! 🌃 How can I help you?",
```

```
    "good night": "Good night! 🌙 If you have any questions, just ask!",
```

```
    "time now": f"The current time is {datetime.now().strftime('%H:%M')}.",
```

```

    "date today": f"Today's date is {datetime.now().strftime('%B %d, %Y')}.",
    "thank you": "You're welcome! 😊 If you have more questions, feel free to ask.",
    "thanks and bye": "You're welcome! 🙌 Have a great day ahead!"

}

```

```

if "temperature" in question.lower() or "weather" in question.lower():
    return get_potheri_temperature()

```

```

for key, value in static_responses.items():
    if re.search(rf"\b{re.escape(key)}\b", q_norm):
        return value

```

```

# --- Step 0: Exact match check first ---

```

```

for entry in data:
    if normalize(entry["prompt"]) == normalize(question):
        return entry["completion"]

```

```

# --- Step 1: Fuzzy matching using RapidFuzz ---

```

```

best_score, best_answer = 0, None

```

```

for entry in data:

```

```
score = fuzz.partial_ratio(q_norm, normalize(entry["prompt"])) # 0-100
```

```
if score > best_score:
```

```
    best_score, best_answer = score, entry["completion"]
```

```
if best_score > 70: # Threshold can be tuned
```

```
    return best_answer
```

```
q_emb = embedder.encode([question], convert_to_numpy=True)
```

```
D, I = index.search(q_emb, top_k)
```

```
retrieved_docs = list(dict.fromkeys([documents[i] for i in I[0]]))
```

```
if len(D[0]) == 0 or D[0][0] > 2.1:
```

```
    return "I'm sorry, I can only answer questions related to SRM."
```

```
context = "\n\n".join(retrieved_docs[:3][:1500])
```

prompt\_text="""You are an AI assistant for SRM students. Answer the user's question using ONLY the context below.

Do NOT copy any sentences verbatim.

Rewrite the information in your own words, concisely and clearly.

Use bullet points if needed.

If the question is unrelated to SRM, reply: "I'm sorry, I can only answer questions related to SRM."



Context:

{context}

User question: {question}

Answer: ""

```
output = generator(prompt_text, max_new_tokens=200, temperature=0.7, top_p=0.95,
do_sample=True,
```

```
pad_token_id=tokenizer.eos_token_id or
tokenizer.pad_token_id)[0]["generated_text"]
```

```
answer = output[len(prompt_text):].strip() if prompt_text in output else output.strip()
```

```
return answer if answer else "I'm sorry, I can only answer questions related to SRM."
```

```
# -----
```

```
# Flask App
```

```
# -----
```

```
app = Flask(__name__)
```

```
DB_FILE = "srm_data.db" # SQLite database file for timetable and seat data
```

```
# Ensure database tables exist
```

```
conn = sqlite3.connect(DB_FILE)
```

```
c = conn.cursor()
```

```
# Create timetable table
```

```
c.execute("""
```

```
CREATE TABLE IF NOT EXISTS timetable (
```

```
    department TEXT,
```

```
    year INTEGER,
```

```
    date TEXT,
```

```
    subject TEXT,
```

```
    code TEXT,
```

```
    time TEXT
```

```
)
```

```
""")
```

```
# Create seats table
```

```
c.execute("""
```

```
CREATE TABLE IF NOT EXISTS seats (
```

```
    register_number TEXT,
```

```
    session TEXT,
```

```
    date TEXT,
```

```
    seat_number TEXT
```

```
)
```

```
""")
```

```

@app.route("/")

def home():

    return render_template("index.html")


@app.route("/ask", methods=["POST"])

def ask():

    data = request.get_json()

    question = data.get("question")

    answer = rag_query(question)

    return jsonify({"answer": answer})


from werkzeug.utils import secure_filename

import os


UPLOAD_FOLDER = "uploads"

os.makedirs(UPLOAD_FOLDER, exist_ok=True)

ALLOWED_EXTENSIONS = {"xlsx"}


DB_FILE = "srm_data.db" # SQLite database file for timetable and seat data


def allowed_file(filename):

```

```
    return "." in filename and filename.rsplit(".", 1)[1].lower() in  
ALLOWED_EXTENSIONS
```

```
@app.route("/upload_timetable", methods=["POST"])
```

```
def upload_timetable():
```

```
    file = request.files.get("file")
```

```
    if not file:
```

```
        return jsonify({"status": "error", "message": "No file uploaded"}), 400
```

```
    if not allowed_file(file.filename):
```

```
        return jsonify({"status": "error", "message": "Only .xlsx files are allowed"}), 400
```

```
    filename = secure_filename(file.filename)
```

```
    filepath = os.path.join(UPLOAD_FOLDER, filename)
```

```
    file.save(filepath)
```

```
    try:
```

```
        df = pd.read_excel(filepath)
```

```
        df.columns = df.columns.str.lower().str.strip()
```

```
        expected_cols = {"department", "year", "date", "subject", "code", "time"}
```

```
        if not expected_cols.issubset(df.columns):
```

```
            raise ValueError(f"Missing columns: {expected_cols - set(df.columns)}")
```

```

# Convert dates to string

df['date'] = pd.to_datetime(df['date'], errors='coerce').dt.strftime('%Y-%m-%d')


conn = sqlite3.connect(DB_FILE)

c = conn.cursor()

c.execute("DELETE FROM timetable") # remove old timetable

for _, row in df.iterrows():

    c.execute("""

        INSERT INTO timetable (department, year, date, subject, code, time)

        VALUES (?, ?, ?, ?, ?, ?)

        """, (row['department'], row['year'], row['date'], row['subject'], row['code'],
row['time']))

    conn.commit()

conn.close()


return jsonify({"status": "success", "message": "Timetable uploaded
successfully!"})

except Exception as e:

    print(f"⚠️ Error uploading timetable: {e}")

    return jsonify({"status": "error", "message": f"Failed to upload timetable: {e}"})
500

```

```

@app.route("/upload_seats", methods=["POST"])

def upload_seats():

    file = request.files.get("file")

    if not file:

        return jsonify({"status": "error", "message": "No file uploaded"}), 400

    if not allowed_file(file.filename):

        return jsonify({"status": "error", "message": "Only .xlsx files are allowed"}), 400

    filename = secure_filename(file.filename)

    filepath = os.path.join(UPLOAD_FOLDER, filename)

    file.save(filepath)

    try:

        df = pd.read_excel(filepath)

        df['Date'] = pd.to_datetime(df['Date']).dt.date

        # Store in SQLite

        conn = sqlite3.connect(DB_FILE)

        c = conn.cursor()

        c.execute("DELETE FROM seats") # remove old seat data

```

```

for _, row in df.iterrows():

    c.execute("""

        INSERT INTO seats (register_number, session, date, seat_number)

        VALUES (?, ?, ?, ?)

        """, (row['Register number'], row['Session'], row['Date'], row['Seat Number']))

    conn.commit()

    conn.close()

    print(f"✅ Seat data uploaded to DB with {len(df)} rows!")

    return jsonify({"status": "success", "message": "Seat data updated successfully!"})

except Exception as e:

    print(f"⚠️ Error uploading seat data: {e}")

    return jsonify({"status": "error", "message": f"Failed to upload seat data: {e}"})
500

```

```

@app.route("/feedback", methods=["POST"])

def feedback():

    data = request.get_json()

    fb = data.get("feedback")

    message = data.get("message")

    with open("feedback_logs.csv", "a") as f:

        f.write(f"{datetime.now()}, {fb}, {message}\n")

```

```

    return jsonify({"status": "ok"})

import csv

from datetime import datetime

FEEDBACK_FILE = "feedback.csv"

try:

    with open(FEEDBACK_FILE, "x", newline="", encoding="utf-8") as f:

        writer = csv.writer(f)

        writer.writerow(["timestamp", "assistant_message", "feedback"])

except FileExistsError:

    pass

@app.route("/feedback", methods=["POST"])

def handle_feedback():

    data = request.get_json()

    message = data.get("message")

    feedback = data.get("feedback")

    if not message or not feedback:

        return jsonify({"status": "error", "message": "Missing data"}), 400

```



```

with open(FEEDBACK_FILE, "a", newline="", encoding="utf-8") as f:

    writer = csv.writer(f)

    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    writer.writerow([timestamp, message, feedback])


return jsonify({"status": "success"})


@app.route("/clear_data", methods=["POST"])
def clear_data():

    data_type = request.form.get("type") # "timetable" or "seats"

    if data_type not in ["timetable", "seats"]:

        return jsonify({"status": "error", "message": "Invalid data type"}), 400

    try:

        conn = sqlite3.connect(DB_FILE)

        c = conn.cursor()

        c.execute(f"DELETE FROM {data_type}") # clears the table

        conn.commit()

        conn.close()

        return jsonify({"status": "success", "message": f"{data_type.capitalize()} data
cleared successfully!"})

```

```

except Exception as e:

    return jsonify({"status": "error", "message": f"Failed to clear data: {e}"}), 500


@app.route('/team')

def team():

    return render_template('team.html')


# Create announcements table

c.execute("""

CREATE TABLE IF NOT EXISTS announcements (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    title TEXT NOT NULL,

    message TEXT NOT NULL,

    date TEXT NOT NULL,

    urgent INTEGER DEFAULT 0

)

""")

conn.commit()

conn.close()


# Fetch all announcements

```

```

@app.route("/get_announcements")

def get_announcements():

    conn = sqlite3.connect(DB_FILE)

    c = conn.cursor()

    c.execute("SELECT id, title, message, date, urgent FROM announcements ORDER
BY id DESC")

    rows = c.fetchall()

    conn.close()

    announcements = [

        {"id": r[0], "title": r[1], "message": r[2], "date": r[3], "urgent": bool(r[4])}

        for r in rows # use the fetched rows, not c.fetchall() again

    ]

    return jsonify(announcements)


# Admin adds a new announcement

@app.route("/admin/add_news", methods=["POST"])

def add_news():

    data = request.get_json()

    title = data.get("title")

    message = data.get("message")

    date = data.get("date")

    urgent = 1 if data.get("urgent") else 0

```

```

if not title or not message or not date:

    return jsonify({"status": "error", "message": "All fields are required!"}), 400


conn = sqlite3.connect(DB_FILE)

c = conn.cursor()

c.execute("INSERT INTO announcements (title, message, date, urgent) VALUES (?,
?, ?, ?)",

        (title, message, date, urgent))

conn.commit()

conn.close()

return jsonify({"status": "success", "message": "Announcement added!"})


# Delete a news item

@app.route("/admin/delete_news/<int:news_id>", methods=["DELETE"])

def delete_news(news_id):

    conn = sqlite3.connect(DB_FILE)

    c = conn.cursor()

    c.execute("DELETE FROM announcements WHERE id = ?", (news_id,))

    conn.commit()

    conn.close()

    return jsonify({"status": "success", "message": "Announcement deleted!"})

```

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5001, debug=True)
```

### 11.3.5 JSON FOR CHATBOT

```
[
  {
    "prompt": "about SRM",
    "completion": "SRMIST 🏠, established in 1985, is a premier private deemed-to-be university located in Kattankulathur, near Chennai, Tamil Nadu. Spanning over 250 acres, the campus houses multiple colleges, including the College of Science & Humanities, and offers a diverse range of undergraduate and postgraduate programs."
  },
  {
    "prompt": "What is FSH at SRM?",
    "completion": "The Faculty of Science & Humanities at SRMIST, established in 2014, provides exceptional learning opportunities across various disciplines. It has over 50,000 students and more than 3,200 faculty members across its campuses."
  },
  {
    "prompt": "follow updates and campus life at FSH",
    "completion": "Stay connected with FSH campus life and updates through their official Instagram handle: @srm_fsh_ktr."
  },
  {
    "prompt": "Books and resources",
```

"completion": "You can access SRM free resources at  
<https://www.srmist.edu.in/library/online-resources/>"

},

{

"prompt": "Library ",

"completion": "You can access SRM free resources at  
<https://www.srmist.edu.in/library/online-resources/>"

},

{

"prompt": "Bus routes",

"completion": "You can access bus routes at  
[https://webstor.srmist.edu.in/web\\_assets/downloads/2025/science-law-management-bus-route-2025-26.pdf](https://webstor.srmist.edu.in/web_assets/downloads/2025/science-law-management-bus-route-2025-26.pdf) with Arrival at 7:50 AM and Departure at 4:00 PM"

},

{

"prompt": "vision of the FSH SRM?",

"completion": "The vision of FSH at SRMIST is to emerge as a world-class university in creating and disseminating knowledge and providing students with a unique learning experience in science, technology, medicine, management, and other areas of scholarship that serve the world and better humanity."

},

{

"prompt": "mission of FSH SRM?",

"completion": "FSH SRMIST's mission includes achieving global excellence through international alliances, advancing knowledge in a rigorous academic and research environment, and building a rewarding and inspiring community fostering freedom, creativity, and innovation."

},

{

"prompt": "programs FSH SRM offer?",

"completion": "FSH offers undergraduate programs, postgraduate programs with core courses and electives, and PhD and advanced research-oriented programs. Programs are designed to emphasize critical thinking, collaboration, problem-solving, and interdisciplinary learning."

},

{

"prompt": "What are the key aspects of teaching and learning at FSH SRMIST?",

"completion": "FSH focuses on evolving curriculum, faculty development, a resource-rich environment with labs and libraries, collaborative learning, and strong industry connections for internships and placements."

},

{

"prompt": "How many departments and faculty staff are there at FSH SRM",

"completion": "FSH has 24 departments, a student body of over 6,000, 200 dedicated faculty members, and 180 research scholars."

},

{

"prompt": "contact SRM for general inquiries?",

"completion": "You can contact SRMIST at +91 44 27417000 / +91 44 27417777 / +91 80 69087000 or via email at infodesk@srmist.edu.in."

},

{

```

    "prompt": "Dean of FSH at SRM?",

    "completion": "Dr. A. Duraisamy is the Dean of the Faculty of Science and
    Humanities at SRMIST, Kattankulathur."

  },

  {

    "prompt": "What is the vision of the Faculty of Science & Humanities (FSH) at
    SRMIST?",

    "completion": "The vision of FSH is to emerge as a world-class university in creating
    and disseminating knowledge and providing students with a unique learning experience
    in science, technology, medicine, management, and other areas of scholarship that serve
    the world and better humanity."

  },

  {

    "prompt": "What is the mission of FSH SRMIST?",

    "completion": "The mission includes achieving global excellence through
    international alliances, advancing knowledge in a rigorous academic and research
    environment, and building a community fostering freedom, creativity, and innovation."

  },

  {

    "prompt": "Which departments are part of FSH SRMIST?",

    "completion": "FSH SRMIST has the following departments: Biochemistry,
    Biotechnology, Career Development Centre, Commerce, Computer Applications,
    Computer Science, Corporate Secretaryship and Accounting & Finance, Defence and
    Strategic Studies, School of Education, Economics, English, Fashion Designing, French,
    Hindi, Hotel and Catering Management, Journalism and Mass Communication,
    Mathematics and Statistics, Physical Education & Sports Sciences, Psychology, Social
    Work, Tamil, Visual Communication, and Yoga."

  },

```



{

"prompt": "What programs are offered by FSH SRMIST?",

"completion": "FSH offers undergraduate programs, postgraduate programs including core courses and electives, PhD and advanced research-oriented programs, and Self Learning Mode (SLM) programs for flexible learning. Specialized courses include Commerce, Computer Applications, Media, Biotechnology, Economics, Psychology, Fashion Designing, Literature, Social Work, Defence and Strategic Studies, and Disaster Management."

},

]