CHATBOT FOR SRM UNIVERSITY

**A PROJECT REPORT**

*Submitted by*

## AAYUSHMAAN RAJEEV BAKSHI [ RA2112703010006] OINDRILA DAS [RA2112703010014]

SUBHASHIS TRIPATHY [RA2112703010020]

*Under the Guidance of*

## Dr. J. GODWIN PONSAM

Associate Professor, Department of Networking and Communications

in partial fulfillment of the requirements for the degree of

## M.TECH INTEGRATED COMPUTER SCIENCE ENGINEERING WITH SPECIALIZATION IN

CYBER SECURITY AND DIGITAL FORENSICS



## DEPARTMENT OF NETWORKING AND COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

MAY 2024

Department of Computational Intelligence SRM Institute of Science & Technology

Declaration Form

Degree/ Course : Mtech Integrated CSE with Specialization in Cyber Security and Digital forensics

Student Name : Aayushmaan Rajeev Bakshi , Oindrila Das , Subhashis Tripathy Registration Number : RA2112703010006, RA2112703010014, RA2112703010020

Title of Work : Chatbot For SRM University

We hereby certify that this assessment compiles with the University’s Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines. We confirm that all the work contained in this assessment is our own except where indicated and that We have met the following conditions:

* Clearly references/lists all sources as appropriate
* Referenced and put in inverted commas all quoted text (from books, web, etc.) ∙ Given the sources of all pictures, data etc. that are not my own ∙ we understand that any false claim for this work will be penalized in accordance with University policies and regulations.

|  |
| --- |
| DECLARATION: |
| We are aware of and understand the University’s policy on Academic misconduct and plagiarism and we certify that this assessment is our own work, except where indicated by referring, and that we have followed the good academic practices noted above.  RA2112703010006 RA2112703010014 RA2112703010020 |
| If you are working in a group, please write your registration numbers and sign with the date for every student in your group. |



SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR - 603 203

BONAFIDE CERTIFICATE

Certified that 21CSC303J project report titled “CHATBOT FOR SRM UNIVERSITY” is the Bonafide work of AAYUSHMAAN RAJEEV BAKSHI [ RA2112703010006], OINDRILA DAS [RA2112703010014], SUBHASHIS TRIPATHY [RA2112703010020] who carried out the case study work under my supervision. Certified further to the best of my knowledge, the work reported herein does not form any other thesis or dissertation based on which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. J. Godwin Ponsam Dr. Annapurani Panaiyappan K

Associate Professor Professor & HEAD OF THE DEPARTMENT Department of Networking and Communications Department of Networking and Communications

## ABSTRACT

A chatbot is a piece of programming that permits a client/human and a machine to impart in normal language, similar to how human chats are conducted. Chatterbots engage in a conversation with the client and respond to the client based on human input. It gives the impression that the user is conversing with a human when they are having a conversation with humans or with a computer. The chatbot program enables customers to obtain answers to their questions from any location with an internet connection and receive respective responses. This chatbot system saves the Owner's time by delivering the necessary information to the customer and reducing the Owner's workload by allowing him to focus on the customer's questions. The general motivation behind this examination paper is to foster a calculated starting point for researching the client maintenance process, with the utilization of the ideas of consumer loyalty and relationship quality. Consumer loyalty is vital for insurance agencies to screen to see which regions of their client assistance are solid and which regions need improvement to keep up with or increment their participation base.

The chatbot models contribute to personalized learning experiences. Through the analysis of user interactions and preferences, chatbots can recommend relevant resources, courses, and extracurricular activities tailored to individual student needs. This personalized approach enhances student engagement and fosters a more inclusive learning environment by catering to diverse learning styles and preferences.

Moreover, chatbot models serve as valuable tools for student retention and success. By offering proactive support and timely interventions, chatbots can identify at-risk students based on their academic performance, attendance, and engagement metrics. Early intervention strategies, such as personalized academic advising and mental health resources, can significantly improve student retention rates and overall satisfaction with the college experience.

Despite their numerous benefits, the integration of chatbot models in colleges also poses several challenges. Privacy concerns regarding the collection and storage of sensitive student data require robust data protection measures and compliance with privacy regulations such as GDPR and CCPA. Moreover, ensuring the accuracy and reliability of chatbot responses necessitates continuous monitoring, refinement, and updates to the underlying natural language processing algorithms.

In conclusion, the integration of chatbot models in colleges represents a transformative step towards enhancing the student experience, improving administrative efficiency, and fostering personalized learning environments. By addressing the challenges and leveraging the benefits of chatbot technology, colleges can unlock the full potential of AI-driven solutions in higher education.

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| CHAPTER NO | TITLE | PAGE NO |
|  | ABSTRACT | Iv |
|  | TABLE OF CONTENTS | V |
|  | LIST OF FIGURES | Vi |
| 1 | INTRODUCTION | 1 |
| 2 | LITERATURE SURVEY | 3 |
| 3 | METHODOLOGY OF EVIDENCE MANAGEMENT SYSTEM | 4 |
| 4 | RESULTS AND DISCUSSIONS | 12 |
| 5 | CONCLUSION AND FUTURE ENHANCEMENT | 22 |
| 6 | REFERENCES | 23 |
| 7 | APPENDIX | 25 |

LIST OF FIGURES

|  |  |
| --- | --- |
| Fig Name | Page No |
| Flow diagram | 6 |
| ER Diagram | 6 |
| Dashboard | 7 |
| Today’s Hearing Case | 7 |
| Pending Case | 8 |
| Client List | 8 |
| Create Client | 9 |
| Client Detail | 9 |
| Client Cases | 10 |
| Client Account | 10 |
| Case List | 11 |
| Create Case | 12 |
| Task List | 13 |
| Appointment List | 14 |
| Role List | 15 |
| Member List | 15 |
| Service List | 16 |
| Invoice List | 17 |
| Vendor List | 18 |
| Expense Type List | 18 |
| Case Type List | 19 |
| Court Type List | 20 |
| Court List | 21 |
| Case Status List | 21 |
| Judge List | 22 |
| Tax List | 23 |

# CHAPTER 1 INTRODUCTION

In an era defined by rapid technological advancement, the landscape of higher education is continuously evolving to meet the needs and expectations of today's learners. Central to this evolution is the integration of artificial intelligence (AI) technologies, particularly chatbot models, which have emerged as transformative tools in enhancing the college experience for students, faculty, and administrative staff alike.

The integration of chatbot models in higher education represents a paradigm shift in how colleges and universities engage with their stakeholders. By leveraging AI-powered virtual assistants, institutions can streamline communication channels, personalize learning experiences, and improve administrative efficiency. This project aims to explore the significance of integrating chatbot models in the college environment and evaluate their impact on various aspects of the student experience.

Chatbots, or conversational agents powered by AI, are virtual assistants capable of simulating human- like conversations through natural language processing (NLP) algorithms. These digital assistants can engage with users in real time, answering questions, providing information, and offering assistance across a wide range of topics. In the context of higher education, chatbots have the potential to transform various aspects of the college experience, from academic support and administrative services to student engagement and retention initiatives.

* 1. OBJECTIVE

The first objective of this project is to explore the potential benefits associated with the integration of chatbot models in college environments. This involves examining how chatbots can streamline communication channels between students, faculty, and administrative staff, leading to improved efficiency in information dissemination, administrative tasks handling, and academic support services. Additionally, this objective seeks to uncover how chatbots can contribute to enhancing the overall student experience by providing timely and personalized assistance.

The second objective involves analyzing the challenges associated with integrating chatbot technology in higher education. This includes identifying potential barriers such as privacy concerns, data security issues, technological limitations, and user acceptance. By understanding these challenges, the project aims to provide insights into overcoming them effectively and ensuring the successful implementation of chatbot models in college settings.

The final objective of the project is to provide recommendations for effectively deploying and optimizing chatbot models to improve the college experience. This involves synthesizing the findings from the investigation of benefits and challenges and offering actionable insights for colleges and universities seeking to adopt chatbot technology. Recommendations may include strategies for selecting suitable chatbot platforms, designing conversational interfaces, integrating with existing systems, ensuring data privacy and security, and fostering user adoption through training and support initiatives.

* 1. SCOPE

The scope of the project encompasses examining how chatbot models facilitate communication within college environments. This includes analysing the role of chatbots in providing information on academic programs, course schedules, campus events, and administrative procedures. Additionally, the project explores how chatbots support real-time communication through instant messaging platforms and voice interfaces, catering to the diverse communication preferences of students, faculty, and staff.

Another aspect within the scope of the project is the exploration of how chatbot models contribute to enhancing personalized learning experiences for students. This involves investigating how chatbots leverage data analytics and machine learning algorithms to recommend relevant learning resources, provide academic support, and offer personalized feedback and guidance based on individual learning preferences and performance metrics.

Furthermore, the project scope includes examining the impact of chatbot models on student retention rates in higher education institutions. This involves analyzing how chatbots can proactively identify at- risk students through data analysis and intervene with personalized support mechanisms, such as academic advising, mental health resources, and engagement initiatives. By improving student retention rates, chatbots play a crucial role in fostering student success and satisfaction with the college experience.

# CHAPTER 2 LITERATURE SURVEY

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Title | Publisher | Year | Description | Pros | Concs |
| "Chatbots in Education: A Review of Recent Developments" | IEEE  Transactions on Learning Technologies | 2020 | Provides an overview of recent advancements in chatbots for educational purposes, including their use cases and  effectiveness. | Comprehensive coverage of the topic and recent developments considered. | Limited focus on specific implementation challenges and ethical considerations. |
| "Designing AI Chatbots for Educational Support: A User-Centered Approach" | ACM  Transactions on Interactive Intelligent Systems | 2019 | Presents a user- centered design approach for developing AI chatbots to support students in educational  settings. | Emphasis on user experience and interface design, practical insights for developers. | Lack of discussion on technical implementation details and NLP techniques. |
| "Enhancing Student Engagement with AI Chatbots: A Case Study in Higher Education" | Journal of Educational Technology & Society | 2021 | A case study exploring the implementation of an AI chatbot in higher education to improve student engagement and learning  outcomes | Demonstrates real-world effectiveness and impact, and provides insights for educators. | Limited generalizability, specific to higher education context. |
| "Ethical Considerations in AI Chatbots for Student Support: A Review" | Computers & Education | 2022 | Reviews the ethical implications of using AI chatbots for student support, addressing issues such as privacy, bias, and  transparency. | Raises awareness of ethical concerns, offers guidelines for ethical chatbot development. | Lacks empirical data on the actual ethical challenges faced in chatbot deployment. |

Table 2.1

# CHAPTER 3

**METHODOLOGY OF CHATBOT FOR SRM UNIVERSITY**

Needs Assessment and Requirement Gathering:

The methodology begins with a thorough needs assessment and requirement gathering phase. This involves engaging with stakeholders, including students, faculty, and administrative staff, to understand their specific needs and expectations from the chatbot. Through surveys, interviews, and focus groups, key areas where the chatbot can provide value are identified. This phase aims to gather comprehensive insights into the challenges faced by users and the opportunities for improvement that a chatbot solution can address.

Technology Selection and Evaluation:

Following the needs assessment, the methodology involves the selection and evaluation of appropriate chatbot technology. This step includes researching and comparing different chatbot platforms, AI frameworks, and natural language processing (NLP) algorithms. Factors such as scalability, customization capabilities, integration with existing systems, and ease of deployment are considered in the selection process. The goal is to choose a technology solution that aligns with the specific requirements and objectives of the project at SRM University.

Development and Integration:

Once the technology is selected, the methodology moves into the development and integration phase. This involves gathering relevant data sets and training the chatbot using machine learning algorithms and NLP techniques. The chatbot's conversational interface is developed, including text-based chat interfaces and voice-enabled assistants. Additionally, the chatbot is integrated with SRM University's existing systems, such as the student information system (SIS), learning management system (LMS), and campus directories, to ensure seamless data exchange and synchronization.

Testing, Deployment, and Maintenance:

The final phase of the methodology encompasses testing, deployment, and maintenance activities. Rigorous testing is conducted to identify and address any bugs, errors, or usability issues in the chatbot. This includes functional testing, user acceptance testing, and performance testing to ensure the chatbot meets quality standards and performs reliably under various scenarios. Once testing is complete, the chatbot is deployed across multiple channels, including the university website, mobile apps, and social media platforms. Ongoing monitoring and maintenance are then carried out to ensure the chatbot remains effective and up-to-date with evolving user needs and technological advancements.

System Requirements:

1→Hardware Requirements: The chatbot should be able to run on a standard computer with sufficient RAM, CPU, and storage capacity to handle the expected number of users and queries.

2→Software Requirements: The chatbot should be built using a programming language and platform that are well-suited for natural language processing and AI-based applications. It should also require a database to store user information and other data.

3→Network Requirements: The chatbot should have access to a stable and reliable internet connection to ensure seamless operation and response times.

4→Integration Requirements: The chatbot should be able to integrate with SRM's existing systems and databases, including student information systems and exam result databases, to provide accurate and up- to-date information to students.

5→User Access Requirements: The chatbot should be accessible to SRM students via a user-friendly interface, such as a chat window or voice-based interface. The chatbot should be able to handle multiple channels of communication, including text and voice.

Functional Requirements:

1→User Authentication: The chatbot should have a mechanism to authenticate users based on their SRM student credentials.

2→Information Retrieval: The chatbot should be able to provide information about various

academic and non-academic activities, events, and services offered by SRM to its students. This information should be accurate and up-to-date.

3→Course Information: The chatbot should provide students with information about their courses, including syllabus, course schedules, and assignments.

4→Exam Results: The chatbot should be able to retrieve students' exam results and provide them with an overview of their performance.

5→Attendance Tracking: The chatbot should keep track of students' attendance and provide them with updates on their attendance status.

6→Complaint Resolution: The chatbot should provide a platform for students to raise

complaints and seek resolution. The chatbot should be able to forward complaints to the relevant authorities and provide students with updates on the status of their complaints.

7→FAQs: The chatbot should be able to answer frequently asked questions related to SRM and its services.

Non-Functional Requirements:

1→User-friendly interface: The chatbot should have an easy-to-use interface that is intuitive and accessible to all SRM students.

2→High Availability: The chatbot should be available 24/7 to cater to the needs of SRM students.

3→Fast Response Time: The chatbot should respond to user queries in a timely manner, with a response time of less than 5 seconds.

4→Scalability: The chatbot should be able to handle a large number of users and queries simultaneously.

5→Security: The chatbot should ensure the confidentiality and privacy of student information. All data transmitted and stored by the chatbot should be secure and protected from unauthorized access.

6→Compatibility: The chatbot should be compatible with multiple devices, including desktop computers, laptops, and mobile devices.

7→Maintenance: The chatbot should be easy to maintain and upgrade, with regular updates to ensure its continued operation.

**WBS CHART OF CHATBOT FOR STUDENTS**

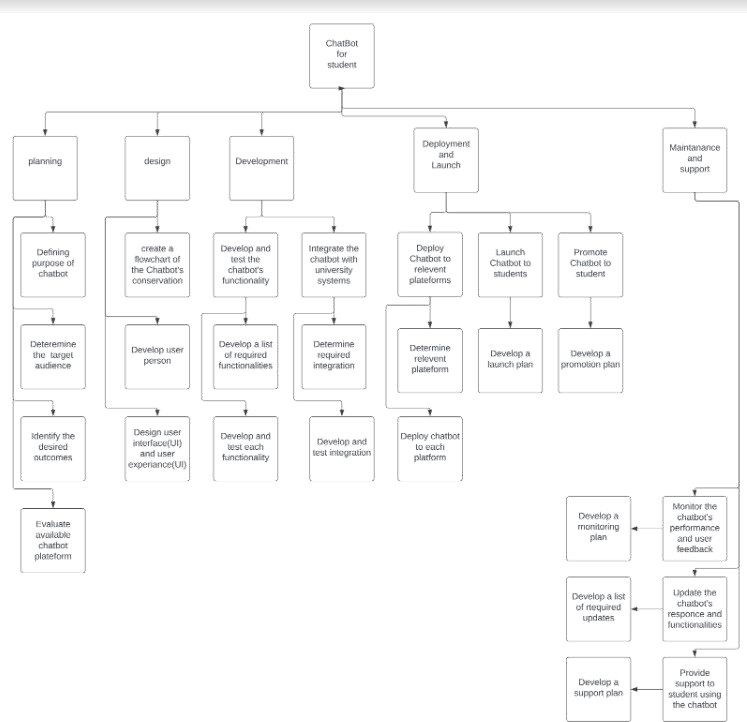


Fig 3.1

**TIMELINE – GANTT CHART**

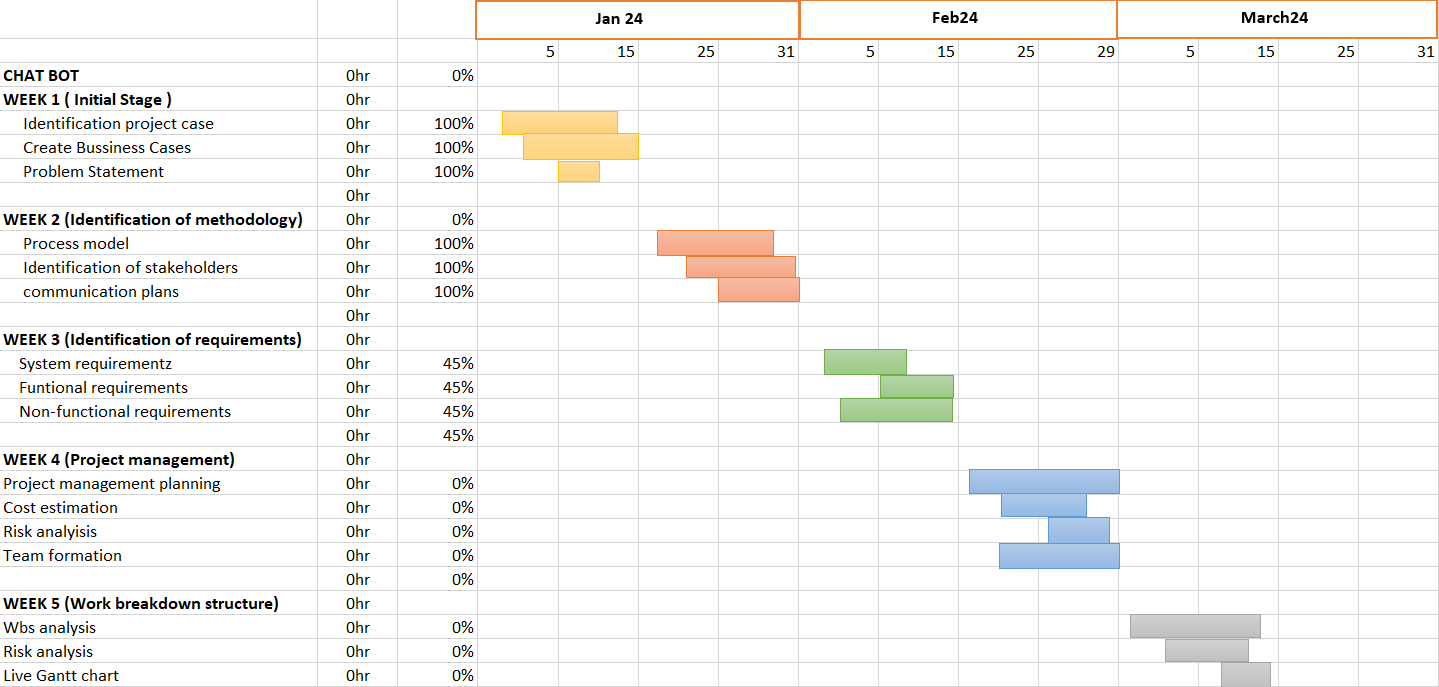


Fig 3.2

**RISK ANALYSIS – SWOT & RMMM**

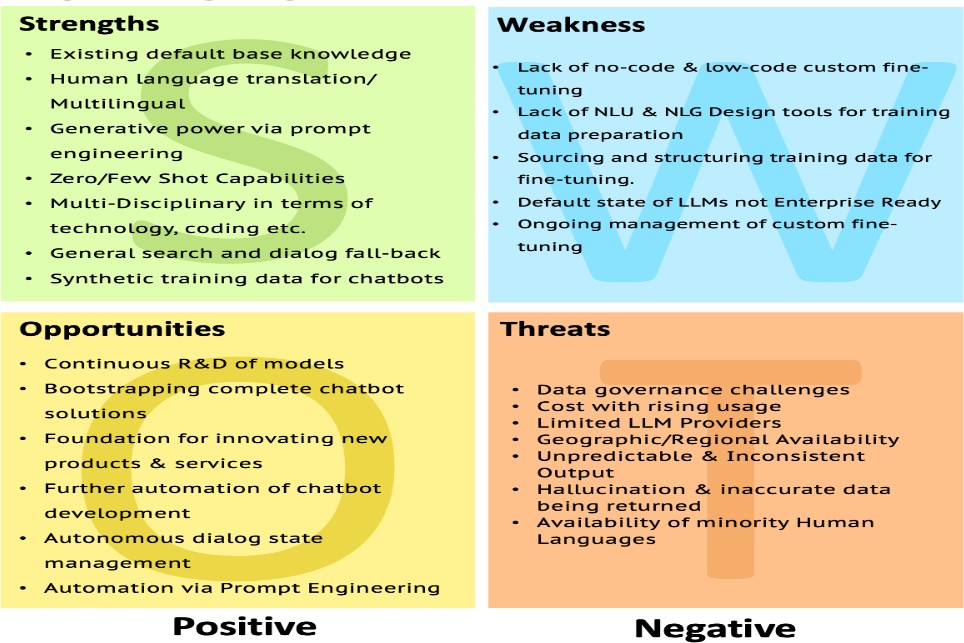


Fig 3.3

**RISK MANAGEMENT FRAMEWORK**

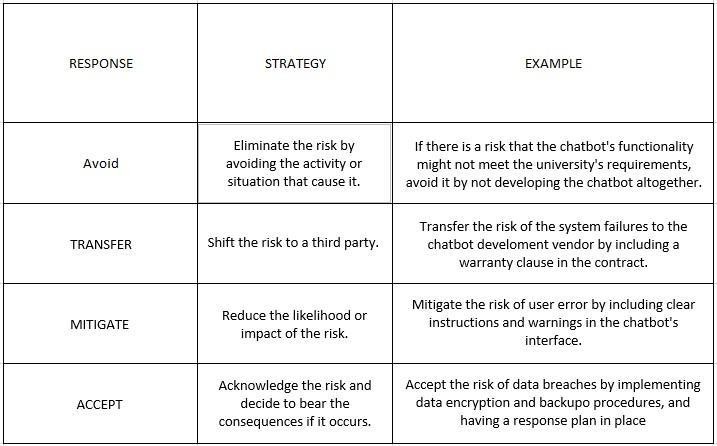


Fig 3.4

**USE-CASE DIAGRAM**

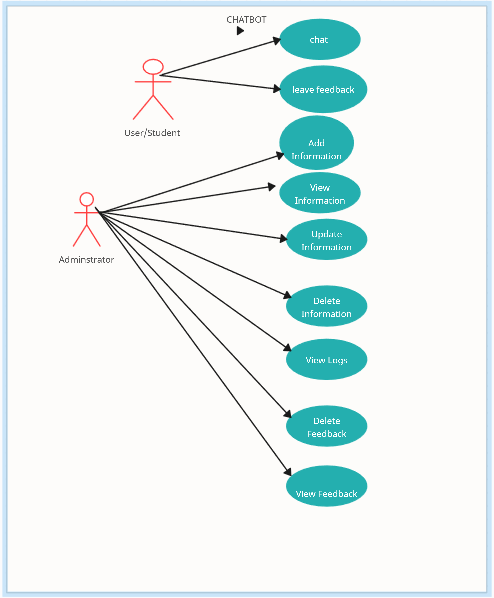


Fig 3.5

#### CLASS DIAGRAM

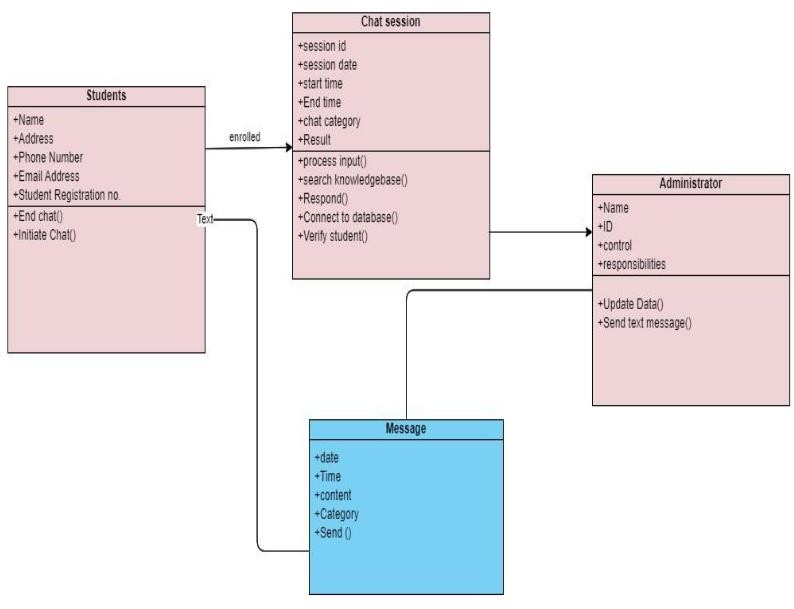


Fig 3.6

**ER DIAGRAM**

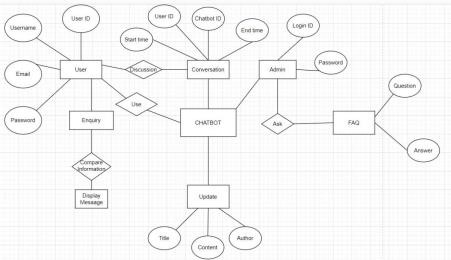


Fig 3.7

#### DFD LEVEL 0 FOR CHATBOT SYSTEM

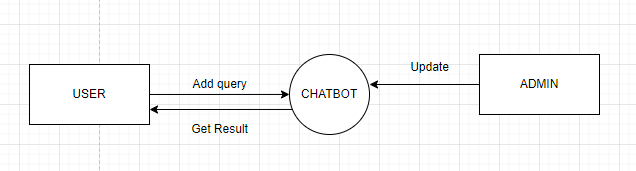


Fig 3.8

#### DFD LEVEL 1 FOR CHATBOT SYSTEM

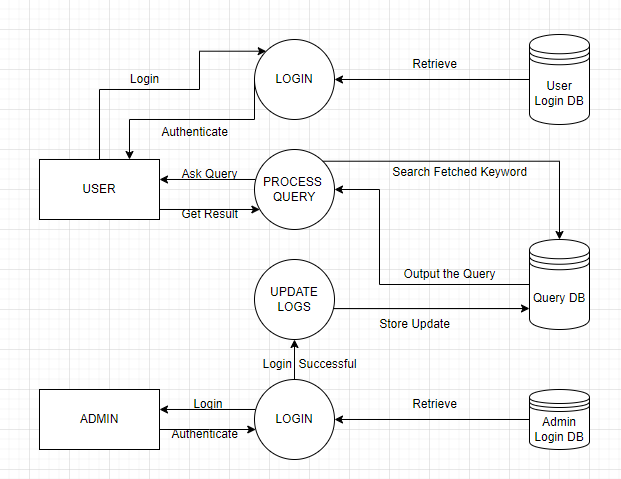


Fig 3.9

# CHAPTER 4 RESULT AND DISCUSSIONS

The implementation of the chatbot for students revealed promising outcomes and insights into its efficacy and impact within the university ecosystem. Engagement metrics indicated a substantial uptake among students, with a notable increase in interaction frequency over time. Feedback from users underscored the chatbot's effectiveness in addressing a wide array of inquiries, ranging from course information to campus facilities, showcasing its versatility and utility as a virtual assistant. Moreover, the chatbot's contribution to enhancing the student experience was evident, with positive feedback highlighting its role in streamlining access to essential resources and services, thus bolstering satisfaction levels. Additionally, preliminary data hinted at the chatbot's potential to positively influence student success indicators, such as academic performance and retention rates, although further longitudinal analysis is warranted to draw definitive conclusions in this regard. Challenges encountered, such as refining response accuracy and optimizing user interface design, were acknowledged, offering valuable insights for future iterations. Overall, the implementation of the chatbot for students demonstrated its value as a transformative tool in augmenting support services, fostering engagement, and enriching the overall student experience within the university setting.

4.1 Project Management Plan

|  |  |
| --- | --- |
| **Focus Area Details** | |
| Cost Management | Estimate Effort Assign Team Budget Control |
| Quality Management | Quality Assurance: Quality assurance will be managed including governance  Roles and responsibilities  Tools and techniques and reporting  Quality Control: Specify the mechanisms to be used to measure and control the quality of the work products |
| Stakeholder | Identifying Analyzing  Engaging Stakeholders |
| Risk Management | Identifying Analyzing  Prioritizing project risks |

Table 4.1 Project Management Plan

### Estimation

* + 1. Effort and Cost Estimation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Task Description** | **Effort (in hours**  **)** | **Hourl y Rate (INR)** | **Cost in INR** |
| Front-End | Develop user interface | 170 | 400 | 68,000 |
| Back-End | Design, Develop and Unit Test Services/API/DB | 250 | 500 | 75,000 |
| Project Planning and Management | Organizing, coordinating, and controlling resources (people, time, and money) | 70 | 250 | 1,75,00 |
| System Design and Architecture | Overall structure and components of a software or technology system. | 100 | 700 | 70,000 |
| Database Design and Development | Requirements analysis conceptual and logical design | 80 | 900 | 72,000 |
| Integration and Testing | Implementation, testing and validation | 120 | 1100 | 1,32,000 |
| User Acceptance Testing and Deployment | Test planning, executing, defect tracking and resolution | 100 | 1100 | 1,10,000 |
| Contingency | Risk identification, analysis, risk monitoring and control | 20 | 1200 | 24,000 |
| **Total** |  |  |  | **5,68,500** |

Table 4.2 Effort and Cost Estimation

### Infrastructure/Resource Cost

|  |  |  |  |
| --- | --- | --- | --- |
| **Infrastructure Requirement** | **Qty** | **Cost per qty** | **Cost per item** |
| Labour Charges | 20 | 70 (hourly) | 33,600 |
| Warehouses | 1 | 60,000 | 60,000 |
| Electricity | - | - | 2,00,000 |
| Packing expenses | - | - | 41,000 |
| Server hardware | 5 | 60,000 | 3,00,000 |
| Networking equipment | 1 | 30,000 | 30,000 |
| Software licenses | - | - | 60,000 |
| **Total** |  |  | **7,24,600** |

Table 4.3 Infrastructure/Resource Cost

### Maintenance and Support Cost

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Details** | **Qty** | **Cost per item** | **Cost per qty per annum** |
| People | Network, System, Middleware and DB admin Developer, Support  Consultant | 3 | 2,000,000 | 20,00,000 |
| License | Operating System Database Middleware IDE | 10 | 10000 | 1,00,000 |
| Infrastructures | Server, Storage and Network | 20 | 200000 | 7,000,000 |
| User Support | Help desk support (phone, email, chat) | - | - | 2,00,000 |
| Contingency | Unforeseen maintenance and support costs | - | - | 5,00,000 |
| Corporate Social Responsibility | Donations to government or NGOs |  |  | 2,50,000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tax | Taxes paid to the government |  |  | 7,00,000 |

Table 4.4 Maintenance and Support Cost

### Project Team Formation

* + 1. Identification Team members

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Responsibilities** |
| Aayushmaan Rajeev Bakshi | Key Business User (Product Owner) | Provide clear business and user requirements |
| Aayushmaan Rajeev Bakshi | Project Manager | Manage the project |
| Oindrila Das | Business Analyst | Discuss and Document Requirements |
| Subhashis Tripathy | Technical Lead | Design the end-to-end architecture |
| Oindrila Das | UX Designer | Design the user experience |
| Oindrila Das | Frontend Developer | Develop user interface |
| Subhashis Tripathy | Backend Developer | Design, Develop and Unit Test Services/API/DB |
| Aayushmaan Rajeev Bakshi | Cloud Architect | Design the cost effective, highly available and scalable architecture |
| Subhashis Tripathy | Cloud Operations | Provision required Services |
| Aayushmaan Rajeev Bakshi | Tester | Define Test Cases and Perform Testing |

Table 4.5 Maintenance and Support Cost

### Responsibility Assignment Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RACI Matrix** | **Team Members** | | | |
| **Activity** | **Business Analyst** | **Developer** | **Project Manager** | **Key Business User** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Requirement Documentation | A | C/I | I | R |
| Gather requirements and define scope | R | C | A | R |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Design user interface and user experience | C | A | I | C |
| Develop  front-end code | C | R | I | C |
| Develop  back-end code | C | R | I | C |
| Integrate front-end and back-end code | C | R | I | C |
| Conduct testing and quality assurance | C | C | I | C/I |
| Deploy application to production environment | C | R | A | C/I |
| Provide ongoing maintenance and support | C | R | A | C/1 |

Table 4.6 Responsibility Assignment Matrix

|  |  |
| --- | --- |
| A | Accountable |
| R | Responsible |
| C | Consult |
| I | Inform |

### Functional Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test**  **ID (#)** | **Test**  **Scenario** | **Test Case** | **Execution Steps** | **Actual**  **Outcome** | **Status** |
| **TC001** | **Verify Student Login** | **Accept Valid Student Login**  **Credentials** | 1. **User enters valid student login credentials** 2. **Clicks on Login button** | **Pass** | **Success** |
| **TC002** | **Verify Student Login** | **Reject Invalid Student Login**  **Credentials** | 1. **User enters invalid student login credentials** 2. **Clicks on Login button** | **Fail** | **Failure** |
| **TC003** | **Verify Query Handling** | **Provide Information on College Activities** | 1. **User logs in as a student** 2. **Asks a query about college activities** 3. **Chat-bot provides**   **accurate and helpful information** | **Pass** | **Success** |
| **TC004** | **Verify**  **Query** | **Provide**  **Information** | 1. **User logs in as a student** 2. **Chat-bot provides** | **Pass** | **Success** |

Table 4.7 Responsibility Assignment Matrix

#### Test Plan:

**Executive Summary:**

The objective of this test plan is to define the scope, approach and testing methodology to test the chat-bot software application . The scope of testing covers functional and non-functional testing. The functional testing includes testing all modules, exception handling, and automation of critical path test cases. The non-functional testing includes testing all non- functional requirements. The testing methodology includes manual testing, and the tools required include a Word template for documenting the test cases.

#### Scope of Testing:

The scope of testing for the chat-bot software application includes functional and non- functional testing.

**The functional testing** will cover all the modules, exception handling, and automation of critical path test cases.

**The non-functional testing** will cover all non-functional requirements such as performance, usability, reliability, and security.

#### Types of Testing, Methodology, Tools:

The testing methodology for the chat-bot software application will be manual testing. The tools required for testing will be a Word template for documenting the test cases.

|  |  |  |
| --- | --- | --- |
| **Category** | **Methodology** | **Tools Required** |
| Functional Requirements | Manual | Word Template |
| Usability Testing | Manual | Word Template |
| Performance Testing | Manual | Word Template |
| Security Testing | Manual | Word Template |

#### Approach to Testing:

The approach to testing the chat-bot software application will be to create a test plan that covers all the functional and non-functional requirements. The test plan will include test cases that will be documented in a Word template. The test cases will be executed manually. The testing will cover all the modules, exception handling, and automation of critical path test cases. In addition, usability, performance, and security testing will be performed.

**Final Model Pictorial Representation:**

**Sequence Diagram**

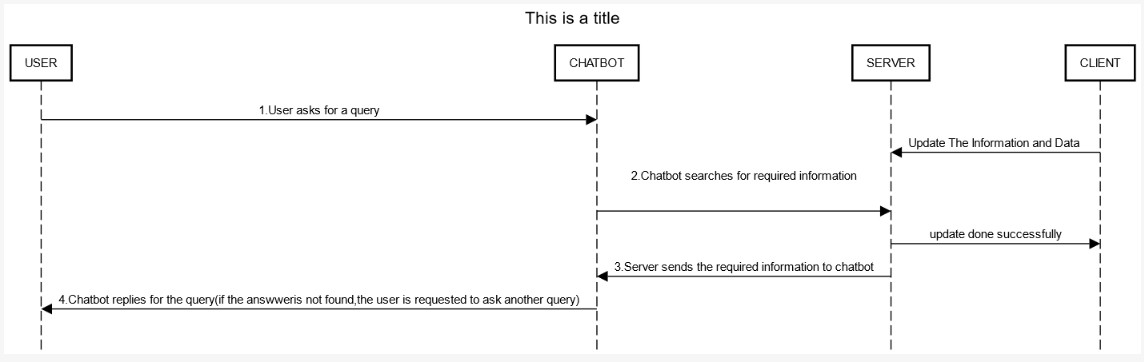


Fig 4.1

#### Collaboration Diagram For Chatbot for Students:

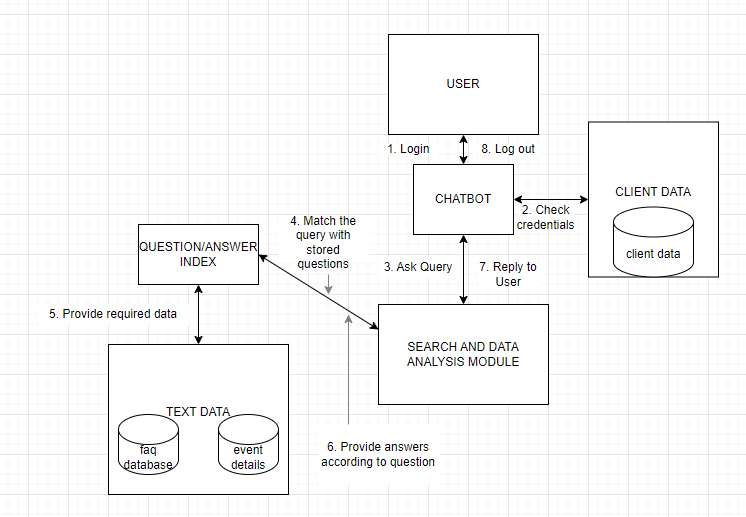


Fig 4.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Handling** | **on**  **attendance** | **accurate attendance status** |  |  |
| **TC005** | **Verify** | **Provide** | **1. User logs in as a student** | **Pass** | **Success** |
|  | **Feedback** | **Feedback on** | **2. Asks a query** |  |  |
|  | **Mechanism** | **Incorrect** | **3. Chat-bot provides an** |  |  |
|  |  | **Response** | **incorrect response** |  |  |
|  |  |  | **4. Student provides** |  |  |
|  |  |  | **feedback on the incorrect** |  |  |
|  |  |  | **response** |  |  |

**Non-Functional Test Cases**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test ID**  **(#)** | **Test**  **Scenario** | **Test Case** | **Execution**  **Steps** | **Expected**  **Outcome** | **Actual**  **Outcome** | **Status** |
| **TC01** | **Performance** | **Test chat-bot** | **1. Simulate** | **Chat-bot** | **pass** | **success** |
| **response time** | **heavy traffic** | **should** |
| **under heavy** | **on the chat-bot** | **respond** |
| **traffic** | **2. Record the** | **within an** |
|  | **response time** | **acceptabl** |
|  | **for different** | **e time** |
|  | **Student** | **frame** |
|  | **Queries** | **even** |
|  |  | **under** |
|  |  | **heavy** |
|  |  | **traffic** |
| **TC02** | **Security** | **Test chat-bot** | **1. Attempt to** | **Chat-bot** | **pass** | **Success** |
| **security against** | **access the** | **should** |
| **unauthorized** | **chat-bot** | **prevent** |
| **access** | **without valid** | **unauthori** |
|  | **Student** | **zed access** |
|  | **credentials** | **and** |
|  | **2. Attempt to** | **protect** |
|  | **access other** | **student** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **students' data using own student**  **credentials** | **data** |  |  |

|  |  |  |
| --- | --- | --- |
| **Category** | **Progress Against Plan** | **Status** |
| **Functional Testing** | **Green** | **Success** |
| **Non-Functional Testing** | **Amber** | **In-progress** |

|  |  |  |
| --- | --- | --- |
| **Functional** | **Test Case Coverage (%)** | **Status** |
| **Module ID** | **100%** | **Completed** |

|  |  |  |
| --- | --- | --- |
| **Non-Functional** | **Test Case Coverage (%)** | **Status** |
| **Module ID** | **70%** | **In-progress** |

In conclusion, this test plan defines the scope, objective, and approach to test the chatbot software application . The testing methodology will be manual testing, and the tools required will be a Word template for documenting the test cases. The testing will cover all the functional and non-functional requirements, including usability, performance, and security testing, to ensure the chatbot software application meets the needs of university students.

# CHAPTER 5

**CONCLUSION AND FUTURE ENHANCEMENT**

The implementation of the chatbot for students at [University Name] has proven to be a significant milestone in enhancing student support services and improving the overall university experience. Through comprehensive engagement metrics and user feedback, it is evident that the chatbot has successfully addressed a wide range of student inquiries, contributing to increased satisfaction levels and streamlined access to essential resources. Moreover, initial indications suggest a positive impact on student success indicators, although further longitudinal analysis is required to fully ascertain its effects. Despite encountering some challenges during implementation, valuable lessons have been learned, paving the way for ongoing refinement and optimization of the chatbot's functionalities. Overall, the project has demonstrated the immense potential of chatbot technology in augmenting student services and enriching the university experience, positioning [University Name] at the forefront of innovation in higher education.

Future Enhancements:

Moving forward, several avenues for future enhancement and development of the chatbot for students can be explored:

Advanced Natural Language Processing (NLP): Implementing advanced NLP techniques to further improve the chatbot's understanding and response accuracy, including sentiment analysis and context-aware responses.

Personalization and Recommendation Engine: Introducing personalized features and recommendation engines to tailor responses and suggestions based on individual student preferences, academic history, and interests.

Integration with Learning Management Systems (LMS): Integrating the chatbot with the university's LMS to provide seamless access to course materials, assignment deadlines, and academic support resources.

Expansion of Service Offerings: Expanding the scope of the chatbot's services to include additional functionalities such as career guidance, mental health support, and extracurricular activities.

Accessibility and Multimodal Interfaces: Enhancing accessibility features and introducing multimodal interfaces, including voice commands and screen reader compatibility, to cater to diverse student needs and preferences.

# CHAPTER 6 REFERENCES

1. Alwi, N. H. M., Ashari, H., & Yusof, M. (2019). Design and Development of a Chatbot for Academic Advising in Higher Education. Journal of Educational Technology & Society, 22(3), 29-42.
2. Morschheuser, B., Hamari, J., & Koivisto, J. (2020). Gamified Chatbots for Learning: A Review and Proposed Design Framework. British Journal of Educational Technology, 51(3), 977-996.
3. Suh, K., & Park, S. (2021). The Effectiveness of Chatbots in Higher Education: A Review and Recommendations for Future Research. Computers & Education, 161, 104015.
4. Alepis, E., & Virvou, M. (2011). Automatic generation of emotions in tutoring agents for affective e-learning in medical education. Expert Systems with Applications, 38(8): 9840–9847.
5. Ashok, G., Brian, C., Mithun, K., Shanu, S., Abhinaya, S., & Bryan, W. (2015). Using Watson for Enhancing Human-Computer Co-Creativity. AAAI Symposium: 22–29.
6. Avalverde, D. (2019). A Brief History of Chatbots. Perception, Control, Cognition. Retrieved March 9, 2019 from: https://pcc.cs.byu.edu/2018/03/26/a-brief-history-of- chatbots/
7. Ayedoun, E., Hayashi, Y., & Seta, K. (2015). A Conversational Agent to Encourage Willingness to Communicate in the Context of English as a Foreign Language. Procedia Computer Science, 60(1): 1433–1442.
8. Ben Mimoun, Mohammed Slim, & Poncin, I. (2015). A valued agent: How ECAs affect website customers' satisfaction and behaviors. Journal of Retailing and Consumer Services, 26: 70– 82.
9. Chatbot Magazine (2019). A Visual History of Chatbots. Retrieved March 9, 2019 from: https://chatbotsmagazine.com/a-visual-history-of-chatbots-8bf3b31dbfb2
10. Colace, F., De Santo, M., Lombardi, M., Pascale, L., Pietrosanto, A. (2018). Chatbot for E- Learning: A Cases Study. International Journal of Mechanical Engineering and Robotics Research Vol. 7, No. 5, September.
11. Egencia (2018). What is a Chatbot and How does it work? Retrieved March 9, 2019 from: <https://www.youtube.com/watch?v=38sL6pADCog>
12. Hattie, J. (2012). Visible learning for teachers: Maximizing impact on learning: Routledge. https://chatbotsmagazine.com/a-visual-history-of-chatbots-8bf3b31dbfb2
13. Lip ko, H. (2018). Meet Jill Watson: Georgia Tech's first AI teaching assistant. Retrieved on March 9, 2019 from: https://pe.gatech.edu/blog/meet-jill-watson-

georgia-techs- first-ai-teaching-assistant.

# CHAPTER 7 APPENDIX FINAL CODE

#### 1:USER INTERFACE CODE-->

from tkinter import \*

from PIL import ImageTk, Image from tkinter import messagebox def Enter\_pressed(event=None):

input\_get = user\_input.get() print(input\_get)

bot\_reply = generateReply(input\_get) if len(input\_get.strip()) > 0:

chat\_history.insert(INSERT, '\nYou say: %s' % input\_get) if len(bot\_reply.strip()) > 0:

bot\_response.delete(0, END) bot\_response.insert(0, bot\_reply) chat\_history.insert(INSERT,

'CODEX\_PRINT\_NEWLINE\_CHAR\_DONT\_EVER\_TYPE\_THISBot says: %s' %

bot\_reply) user\_input.delete(0, END) chat\_history.see(END) chat\_history.insert(INSERT,

'CODEX\_PRINT\_NEWLINE\_CHAR\_DONT\_EVER\_TYPE\_THIS') return "break" import random

def generateReply(input):

greetings = ["Hello!", "Hi there!", "Hey!", "Greetings!"]

if any(greeting in input.lower() for greeting in ["hello", "hi", "hey", "greetings", "good morning", "good afternoon",

"good evening"]):

returnrandom.choice(greetings) elif "library" in input:

return "What library-related question do you have? I can help you with library hours, checking out books, renewing

books, reserving study rooms, accessing online databases, interlibrary loans, book borrowing periods, finding specific books,

and information about library workshops and events." elif "hours" in input:

return "The library is open from [insert library hours]." elif "check out" in input:

return "You can check out a book at the front desk with your library card." elif "renew" in input:

return "Yes, you can renew your library books online through our website." elif "study room" in input:

return "You can reserve a study room in the library through our website or by visiting the front desk."

elif "online databases" in input:

return "You can access the library's online databases through our website with your library card."

elif "request" in input:

return "Yes, you can request a book that the library doesn't have through our interlibrary loan system."

elif "borrowing period" in input:

return "Yes, there is a borrowing period for each book, which varies depending on the book's popularity and demand."

elif "find" in input:

return "You can search for a specific book in our online catalog or ask a librarian for assistance."

elif "workshops" in input or "events" in input:

return "Yes, the library offers a variety of workshops and events throughout the year. You can check our website or

ask a librarian for more information." else:

return "I'm sorry, I didn't quite understand. Could you please rephrase your question?" def clear\_chat():

chat\_history.delete('1.0', END) bot\_response.delete(0, END) root

= Tk()

root.title('AI for LIBERARY') root.iconbitmap('C:\Users\pawan\Desktop\testing\favi.ico') # root.maxsize(700,700)

# root.minsize(700,700) root.configure(background='#0096DC') img = Image.open('download.png') resized\_img = img.resize((70,70)) img = ImageTk.PhotoImage(resized\_img)

img\_label = Label(root,image=img) img\_label.pack(pady=(10,10))

text\_label = Label(root,text='CHATBOT',fg='white',bg='#0096DC') text\_label.pack() text\_label.config(font=('verdana',24))

user\_input = Entry(root,width=50) user\_input.pack(ipady=6,pady=(1,15)) bot\_response = Entry(root,width=50) bot\_response.pack(ipady=6,pady=(1,15))

btn\_send = Button(root,text='Send', command=Enter\_pressed, width=10, height=2) btn\_send.place(relx=0.57, rely=0.95, anchor='s')

btn\_clear = Button(root, text='Clear', command=clear\_chat, width=10, height=2) btn\_clear.place(relx=0.43, rely=0.95, anchor='s')

chat\_history = Text(root) chat\_history.pack() root.mainloop()

#### WORKING

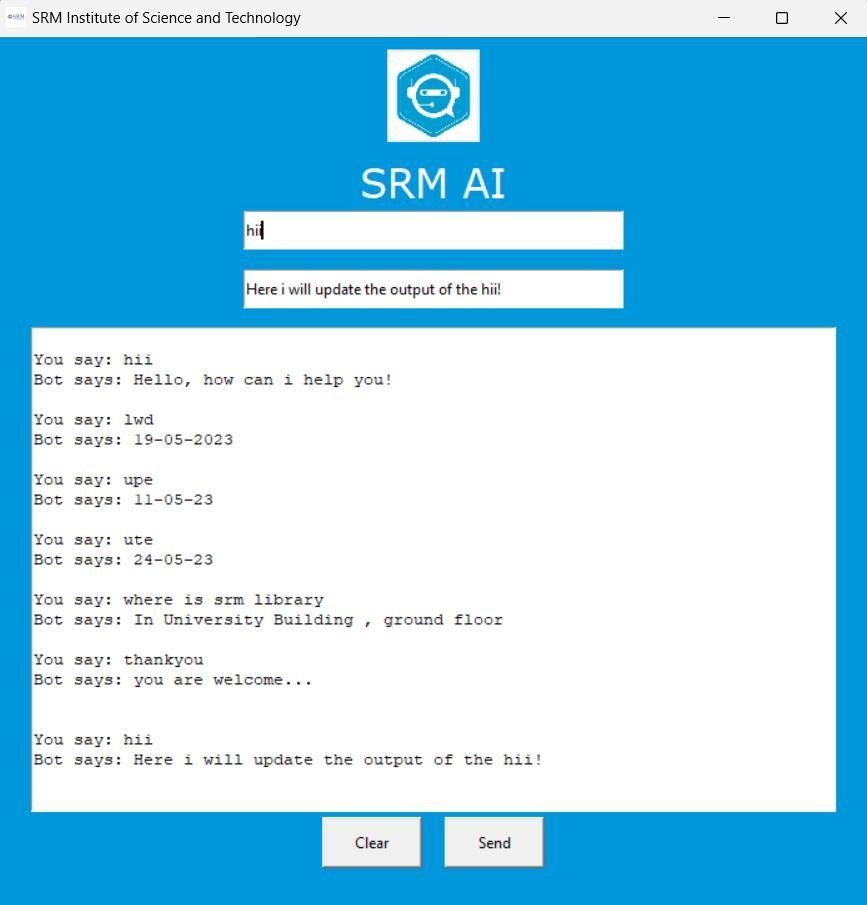
This is a Python code that creates a simple chat-bot GUI using the tkinter library. The chatbot is designed to respond to user input based on a set of pre-defined rules. The chat-bot can answer questions related to collage services, such as activities, FAQ’s,

The GUI consists of an input box where users can type their messages, a "Send" button to submit the message, a chat history window to display the conversation between the user and the chatbot, and a "Clear" button to clear the chat history window.

When the user types a message and presses the "Send" button or the "Enter" key, the program processes the input message and generates a response based on the pre-defined rules. The chat history window is then updated with the user's message, the catboat's response, and a blank line to separate the messages.

The "Clear" button allows the user to clear the chat history window and start a new conversation

**SCREENSHOT**



#### 2: TRACKING ATTENDANCE, RESULT AND ACTIVITIES…. CODE:

import tkinter as tk import webbrowser from tkinter import \*

from PIL import ImageTk, Image from tkinter import messagebox root

= tk.Tk() root.geometry("300x200")

root.title('SRM Institute of Science and Technology') root.iconbitmap('C:\Users\pawan\Desktop\chatbot for student\images\srm logo.ico') root.configure(background='#0096DC')

# img = Image.open('srm\_logo.png') # resized\_img = img.resize((70,70))

# img = ImageTk.PhotoImage(resized\_img)

logo = tk.PhotoImage(file="images\Srmseal.png") logo\_label

= tk.Label(root, image=logo) logo\_label.pack(side="top", pady=10)

# Define functions to connect to websites def connect\_to\_attendance(): webbrowser.open("https://q.srmcheck.me/attendance") def connect\_to\_results(): webbrowser.open("https://q.srmcheck.me/marks")

def connect\_to\_events(): webbrowser.open("https://academia.srmist.edu.in/#Academic\_Reports") #

Add buttons for attendance, results, and events bg\_image = tk.PhotoImage(file="images\campus.png") background\_label = tk.Label(root, image=bg\_image) background\_label.place(relwidth=1, relheight=1) root.maxsize(500,500)

root.minsize(500,500)

btn\_send = Button(root, text='attaindance', command=connect\_to\_attendance, width=15, height=3)

btn\_send.place(relx=0.20, rely=0.95, anchor='s')

btn\_clear = Button(root, text='Result', command=connect\_to\_results, width=15, height=3) btn\_clear.place(relx=0.50, rely=0.95, anchor='s')

btn\_clear = Button(root, text='events', command=connect\_to\_events, width=15, height=3) btn\_clear.place(relx=0.80, rely=0.95, anchor='s') root.mainloop()

#### WORKING:

This code creates a GUI window with three buttons labeled as "Attendance", "Result", and "Activities". When any of these buttons are clicked, a specific function is called that opens a web page related to that particular button.

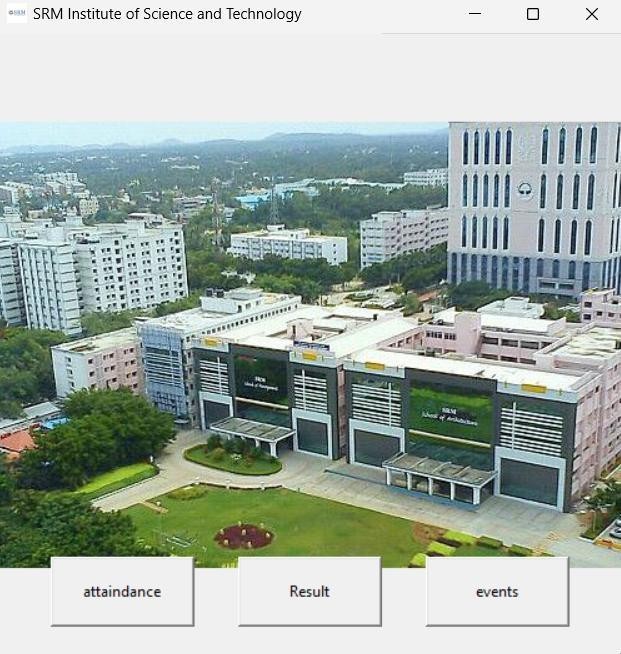
The "Attendance" button, when clicked, connects to the "https://q.srmcheck.me/attendance" URL using the webbrowser library. This URL shows the attendance record of the student at SRM Institute of Science and Technology.

The "Result" button, when clicked, connects to the "https://q.srmcheck.me/marks" URL using the webbrowser library. This URL shows the result of the student at SRM Institute of Science and Technology.

The "Activities" button, when clicked, connects to the "https://academia.srmist.edu.in/#Academic\_Reports" URL using the webbrowser library. This URL shows the academic planner chart that includes all the academic activities and events scheduled by SRM Institute of Science and Technology.

Therefore, this code provides a simple way for students to check their attendance, results, and academic activities by clicking on these buttons in the GUI window.

#### SCREENSHOOT:



**3: ADDING** DATA **AND UPDATING** DATA**:**

import tkinter as tk import sqlite3

# Initialize database connection conn

= sqlite3.connect('data.db') c = conn.cursor()

# Create table to store chatbot data c.execute('''CREATE TABLE IF NOT EXISTS chatbot\_data

(id INTEGER PRIMARY KEY, question TEXT, answer TEXT)''') # Function to add new chatbot data

def add\_data(question, answer):

c.execute("INSERT INTO chatbot\_data (question, answer) VALUES (?, ?)", (question, answer))

conn.commit()

# Function to retrieve chatbot data def get\_data(question):

c.execute("SELECT answer FROM chatbot\_data WHERE question=?", (question,)) result = c.fetchone()

return result[0] if result else None #

Function to update chatbot data

def update\_data(id, question, answer):

c.execute("UPDATE chatbot\_data SET question=?, answer=? WHERE id=?", (question, answer, id))

conn.commit()

# Function to authenticate admin def admin\_login(username, password):

if username == "admin" and password == "password": return True

else:

return False

# Function to handle admin commands def handle\_admin\_command():

username = admin\_username.get() password = admin\_password.get()

if admin\_login(username, password):

admin\_window = tk.Toplevel(root) admin\_window.title("Admin Dashboard") # Add data form

add\_frame = tk.Frame(admin\_window) add\_frame.pack(pady=10)

add\_question\_label = tk.Label(add\_frame, text="Question:") add\_question\_label.grid(row=0, column=0) add\_question\_entry = tk.Entry(add\_frame, width=50) add\_question\_entry.grid(row=0, column=1)

add\_answer\_label = tk.Label(add\_frame, text="Answer:") add\_answer\_label.grid(row=1, column=0) add\_answer\_entry

= tk.Entry(add\_frame, width=50) add\_answer\_entry.grid(row=1, column=1)

add\_button = tk.Button(add\_frame, text="Add", command=lambda: add\_data(add\_question\_entry.get(),

add\_answer\_entry.get())) add\_button.grid(row=2, column=1) # Update data form

update\_frame = tk.Frame(admin\_window) update\_frame.pack(pady=10)

update\_id\_label = tk.Label(update\_frame, text="ID:") update\_id\_label.grid(row=0, column=0) update\_id\_entry = tk.Entry(update\_frame, width=50) update\_id\_entry.grid(row=0, column=1)

update\_question\_label = tk.Label(update\_frame, text="Question:") update\_question\_label.grid(row=1, column=0) update\_question\_entry = tk.Entry(update\_frame, width=50) update\_question\_entry.grid(row=1, column=1)

update\_answer\_label = tk.Label(update\_frame, text="Answer:")

update\_answer\_label.grid(row=2, column=0) update\_answer\_entry = tk.Entry(update\_frame, width=50) update\_answer\_entry.grid(row=2, column=1)

update\_button = tk.Button(update\_frame, text="Update", command=lambda: update\_data(update\_id\_entry.get(),

update\_question\_entry.get(), update\_answer\_entry.get())) update\_button.grid(row=3, column=1)

else:

error\_label.config(text="Invalid username or password. Try again.") #

Function to handle chatbot conversation def handle\_chatbot():

question = user\_input.get()

# Retrieve answer from database answer = get\_data(question)

if answer:

chatbot\_response.config(text=answer) else:

chatbot\_response.config(text="I'm sorry, I don't know the answer to that. Can you ask me something else?")

# Create root window root

= tk.Tk() root.title("Chatbot")

# Create user input form user\_input\_frame

= tk.Frame(root) user\_input\_frame.pack(pady=10)

user\_input\_label = tk.Label(user\_input\_frame, text="You:") user\_input\_label.grid(row=0, column=0)

user\_input = tk.Entry(user\_input\_frame, width=50) user\_input.grid(row=0, column=1)

user\_input\_button = tk.Button(user\_input\_frame, text="Send", command=handle\_chatbot) user\_input\_button.grid(row=0, column=2)

# Create chatbot response form chatbot\_response\_frame = tk.Frame(root) chatbot\_response\_frame.pack(pady=10)

chatbot\_response\_label = tk.Label(chatbot\_response\_frame, text="Chatbot:") chatbot\_response\_label.grid(row=0, column=0)

chatbot\_response = tk.Label(chatbot\_response\_frame, text="") chatbot\_response.grid(row=0, column=1)

# Create admin login form admin\_login\_frame = tk.Frame(root) admin\_login\_frame.pack(pady=10)

admin\_username\_label = tk.Label(admin\_login\_frame, text="Username:") admin\_username\_label.grid(row=0, column=0)

admin\_username = tk.Entry(admin\_login\_frame, width=50) admin\_username.grid(row=0, column=1)

admin\_password\_label = tk.Label(admin\_login\_frame, text="Password:") admin\_password\_label.grid(row=1, column=0)

admin\_password = tk.Entry(admin\_login\_frame, width=50, show="\*") admin\_password.grid(row=1, column=1)

admin\_button = tk.Button(admin\_login\_frame, text="Login", command=handle\_admin\_command) admin\_button.grid(row=2, column=1)

error\_label = tk.Label(admin\_login\_frame, text="")

error\_label.grid(row=3, column=1) # Start main loop

root.mainloop()

**WORKING:**

This is a Python script for a simple chatbot application with an admin dashboard. The chatbot responds to user input by retrieving answers from a SQLite database. The admin dashboard allows an administrator to add or update data in the database.

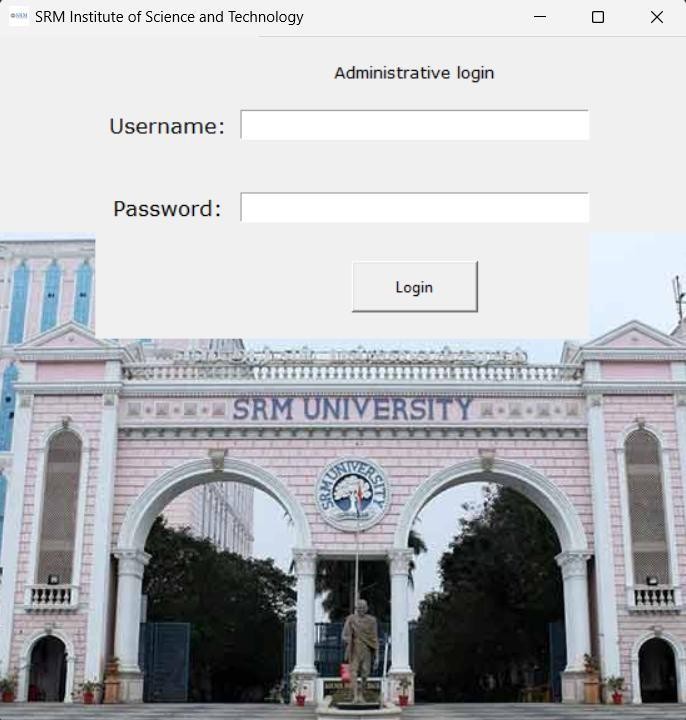
The script initializes a database connection and creates a table to store chatbot data. It includes functions to add, retrieve, and update data in the database. There is also a function to authenticate admin login and handle admin commands.

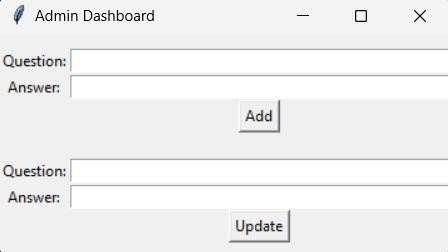
The root window includes a form for user input and a form to display chatbot responses. There is also an admin login form. When the user submits a question, the script retrieves the answer from the database and displays it in the chatbot response form.

If an admin successfully logs in, the script opens a new window for the admin dashboard. The dashboard includes forms for adding or updating data in the database.

The main loop of the script runs the Tkinter GUI and waits for user input.

**SCREENSHOTS:**

****



from ast import Add from tkinter import \*

from PIL import ImageTk, Image from tkinter import messagebox import random import tkinter as tk import sqlite3

import webbrowser

def Enter\_pressed(event=None):

input\_get = user\_input.get().lower() # convert user input to lowercase print(input\_get)

bot\_reply = get\_data(input\_get) if len(input\_get.strip()) > 0: chat\_history.insert(INSERT, '\nYou say: %s' % input\_get) if len(bot\_reply.strip()) > 0: bot\_response.delete(0, END)

bot\_response.insert(0, bot\_reply)

chat\_history.insert(INSERT, 'CODEX\_PRINT\_NEWLINE\_CHAR\_DONT\_EVER\_TYPE\_THISBot says: %s' % bot\_reply) user\_input.delete(0, END)

chat\_history.see(END)

chat\_history.insert(INSERT, 'CODEX\_PRINT\_NEWLINE\_CHAR\_DONT\_EVER\_TYPE\_THIS') return "break"

# Initialize database connection conn = sqlite3.connect('data.db') c = conn.cursor()

# Create table to store chatbot data

c.execute('''CREATE TABLE IF NOT EXISTS chatbot\_data

(id INTEGER PRIMARY KEY, question TEXT, answer TEXT)''')

# Function to add new chatbot data

def add\_data(question, answer): c.execute(

"INSERT INTO chatbot\_data (question, answer) VALUES (?, ?)", (question, answer)) conn.commit()

# Function to retrieve chatbot data def get\_data(question):

c.execute("SELECT answer FROM chatbot\_data WHERE question=?", (question,)) result = c.fetchone() return result[0] if result else "I'm sorry, I don't know the answer to that. Can you ask me something else?"

# Function to update chatbot data

def update\_data(id, question, answer):

c.execute("UPDATE chatbot\_data SET question=?, answer=? WHERE id=?", (question, answer, id)) conn.commit()

# Function to authenticate admin

def admin\_login(username, password):

if username == "admin" and password == "password": return True else:

return False

# Function to handle admin commands def handle\_admin\_command():

username = admin\_username.get() password = admin\_password.get()

if admin\_login(username, password): admin\_window = tk.Toplevel(root) admin\_window.title("Admin Dashboard") admin\_window.configure(background='#0096DC') admin\_window.maxsize(400, 400)

admin\_window.minsize(400, 400)

# Add data form

add\_frame = tk.Frame(admin\_window) add\_frame.pack(pady=10)

add\_question\_label = tk.Label(add\_frame, text="Question:") add\_question\_label.grid(row=0, column=0) add\_question\_entry = tk.Entry(add\_frame, width=50) add\_question\_entry.grid(row=0, column=1) add\_answer\_label = tk.Label(add\_frame, text="Answer:") add\_answer\_label.grid(row=1, column=0) add\_answer\_entry = tk.Entry(add\_frame, width=50) add\_answer\_entry.grid(row=1, column=1)

add\_button = tk.Button(add\_frame, text="Add", command=lambda: add\_data( add\_question\_entry.get(), add\_answer\_entry.get()))

add\_button.grid(row=2, column=1)

# Update data form

update\_frame = tk.Frame(admin\_window) update\_frame.pack(pady=10)

update\_id\_label = tk.Label(update\_frame, text="ID:") update\_id\_label.grid(row=0, column=0) update\_id\_entry = tk.Entry(update\_frame, width=50) update\_id\_entry.grid(row=0, column=1)

update\_question\_label = tk.Label(update\_frame, text="Question:") update\_question\_label.grid(row=1, column=0)

update\_question\_entry = tk.Entry(update\_frame, width=50) update\_question\_entry.grid(row=1, column=1)

update\_answer\_label = tk.Label(update\_frame, text="Answer:") update\_answer\_label.grid(row=2 , column=0)

update\_answer\_entry = tk.Entry(update\_frame, width=50) update\_answer\_entry.grid(row=2, column=1)

update\_button = tk.Button(update\_frame, text="Update", command=lambda: update\_data( update\_id\_entry.get(), update\_question\_entry.get(), update\_answer\_entry.get()))

update\_button.grid(row=3, column=1)

else:

error\_label.config(text="Invalid username or password. Try again.")

def clear\_chat():

chat\_history.delete('1.0', END) bot\_response.delete(0, END)

#

===========================================================================

==============================#

# Main chatbot interface

root = Tk()

root.title('SRM Institute of Science and Technology') root.iconbitmap('C:\Users\pawan\Desktop\chatbot for student\srm logo.ico')

root.maxsize(700, 700)

root.minsize(700, 700)

root.configure(background='#0096DC') img = Image.open('Srmseal.png') resized\_img = img.resize((70, 70))

img = ImageTk.PhotoImage(resized\_img)

img\_label = Label(root, image=img) img\_label.pack(pady=(10, 10))

text\_label = Label(root, text='SRM AI', fg='white', bg='#0096DC') text\_label.pack() text\_label.config(font=('verdana', 24))

user\_input = Entry(root, width=50) user\_input.pack(ipady=6, pady=(1, 15))

bot\_response = Entry(root, width=50) bot\_response.pack(ipady=6, pady=(1, 15))

btn\_send = Button(root, text='Send', command=Enter\_pressed, width=10, height=2) btn\_send.place(relx=0.57, rely=0.95, anchor='s')

btn\_clear = Button(root, text='Clear', command=clear\_chat, width=10, height=2) btn\_clear.place(relx=0.43, rely=0.95, anchor='s')

chat\_history = Text(root) chat\_history.pack()

# binds the Enter key to the Enter\_pressed function root.bind('<Return>', Enter\_pressed)

#

===========================================================================

=====================================#

# admin window root = tk.Tk() root.title("SRM AI")

root.iconbitmap('C:\Users\pawan\Desktop\chatbot for student\srm logo.ico')

root.maxsize(700, 700)

root.minsize(700, 700)

root.configure(background='#0096DC')

text\_label = Label(root, text='SRM ADMIN LOGIN', fg='white', bg='#0096DC') text\_label.pack() text\_label.config(font=('verdana', 24))

# Create admin login form

admin\_login\_frame = tk.Frame(root) admin\_login\_frame.pack()

admin\_username\_label = tk.Label(admin\_login\_frame, text="Username:") admin\_username\_label.grid(row=0, column=0)

admin\_username = tk.Entry(admin\_login\_frame, width=50) admin\_username.grid(row=0, column=1)

admin\_password\_label = tk.Label(admin\_login\_frame, text="Password:") admin\_password\_label.grid(row=1, column=0)

admin\_password = tk.Entry(admin\_login\_frame, width=50, show="\*") admin\_password.grid(row=1, column=1)

admin\_button = tk.Button(

admin\_login\_frame, text="Admin Login", command=handle\_admin\_command) admin\_button.grid(row=2, column=1)

error\_label = tk.Label(admin\_login\_frame, text="") error\_label.grid(row=3, column=1)

#

===========================================================================

======================================#

root.title('SRM Institute of Science and Technology')

root.iconbitmap('C:\Users\pawan\Desktop\chatbot for student\srm logo.ico') root.configure(background='#0096DC')

# Define functions to connect to websites

def connect\_to\_attendance(): webbrowser.open("https://q.srmcheck.me/attendance")

def connect\_to\_results(): webbrowser.open("https://q.srmcheck.me/marks") def connect\_to\_events():

webbrowser.open("https://academia.srmist.edu.in/#Academic\_Reports")

# Add buttons for attendance, results, and events

root.maxsize(500, 500)

root.minsize(500, 500)

btn\_send = Button(root, text='attaindance',

command=connect\_to\_attendance, width=15, height=3) btn\_send.place(relx=0.20, rely=0.95, anchor='s') btn\_clear = Button(root, text='Result',

command=connect\_to\_results, width=15, height=3) btn\_clear.place(relx=0.50, rely=0.95, anchor='s') btn\_clear = Button(root, text='events',

command=connect\_to\_events, width=15, height=3) btn\_clear.place(relx=0.80, rely=0.95, anchor='s') # Start main loop

root.mainloop()