

A MINI PROJECT REPORT

On

Student Monitoring System Using Chatbot

Submitted in partial fulfilment of the Requirement for

the award of the degree of

Bachelor of Engineering

in

Computer Science and Engineering

By

Yerukala Vamshi (1608-20-733-074)

Mullapudi Venkata krishna sai (1608-20-733-307)

Thakur Harsh Raj Singh (1608-20-733-308)

Under the guidance of

Mrs. Swapna Mudrakola

Assistant Professor



Department of Computer Science and Engineering

Matrusri Engineering College

Accredited by NBA & NAAC

(Affiliated to Osmania University, Approved by AICTE)

Saidabad, Hyderabad-500059 (2022-2023)

Department of Computer Science and Engineering

Matrusri Engineering College

Accredited by NBA & NAAC

(Affiliated to Osmania University, Approved by AICTE)

Saidabad, Hyderabad-500059

(2022-2023)



CERTIFICATE

This is to Certify that A Mini Project report entitled “**Student Monitoring System Using Chatbot**” is being submitted by Yerukala Vamshi (1608-20-733-074), Mullapudi Venkata Krishna Sai (1608-20-733-307), Thakur Harsh Raj Singh (1608-20733-308) in partial fulfilment of the requirement of the award for the degree of Bachelor of Engineering in “Computer Science and Engineering” O.U., Hyderabad during the year 2022-2023 is a record of bonafide work carried out by him/her under my guidance. The results presented in this thesis have been verified and are found to be satisfactory.

Project Guide

H.O.D.

Mrs. Swapna Mudrakola

Assistant Professor,

Dept. of CSE

Dr. P. Vijayapal Reddy

Professor & Head,

Dept. of CSE

External Examiner(s)

II

Department of Computer Science and Engineering

Matrusri Engineering College

Accredited by NBA & NAAC

(Affiliated to Osmania University, Approved by AICTE)

Saidabad, Hyderabad-500059

(2022-2023)



DECLARATION

We, Yerukala Vamshi (1608-20-733-074), Mullapudi Venkata Krishna Sai (1608-20-733-307), Thakur Harsh Raj Singh (1608-20-733-308), hereby certify that the project report entitled “Student Monitoring System Using Chatbot” is submitted in the partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Computer Science and Engineering.

This is a record of the bonafide work carried out by us under the guidance of Mrs. Swapna Mudrakola, Associative Professor, Matrusri Engineering College, Saidabad, Hyderabad. The Results embodied in this report have not been reproduced/copied from any source. The results embodied in this report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Yerukala Vamshi (1608-20-733-074)

Mullapudi Venkata Krishna Sai (1608-20-733-307)

Thakur Harsh Raj Singh (1608-20-733-308)

CSE Dept. of MECS

ACKNOWLEDGEMENT

This project consumed huge amount of work, research and dedication. Still implementation would not have been possible if we did not have support of my Project Guide, Project Coordinator, Head of the Department and Principal. Therefore, we like to extend our sincere gratitude to all of them.

We are grateful to our project guide **Mrs. Swapna Mudrakola**, Assistant Professor, for provision of expertise, technical support and guidance in the implementation.

We wish to express our gratitude to project coordinators for their indefatigable inspiration, constructive criticisms and encouragement throughout this dissertation work.

We would like to express our sincere thanks to the Professor and Head of the Department, **Dr. P. Vijaya Pal Reddy**, for permitting us to do this project.

We would like to express our gratitude to **Dr. D. Hanumantha Rao**, principal of Matrusri Engineering College who permitted to carry out this project as per the academics.

We would like to thank CSE Department for providing us this opportunity to share and contribute our part of work to accomplish the project in time and all the teaching and support staff for their steadfast support and encouragement.

Nevertheless, we express our gratitude towards our families and colleagues for their kind cooperation and encouragement which helped us in completion of this project.

Yerukala Vamshi (1608-20-733-074)
Mullapudi Venkata Krishna Sai (1608-20-733-307)
Thakur Harsh Raj Singh (1608-20-733-308)

ABSTRACT

Student monitoring system using a chatbot project can be used to monitor the overall student activities and academic performance by providing personalized support and assistance. All the schools, colleges and the universities all over the globe will be having the work of keeping the track of the student details, information related to health. and other requirements that a student requires. But managing this information through the pen paper mode will be a tedious job.

The student monitoring system project will help the staffs to keep track of the student performance and other requirements that the student needs. The Existing system is a manual entry for the students. Here the monitoring of students will be carried out in the handwritten registers. It will be a difficult task to maintain the record for the user. The human effort is more here. The retrieval of the information is not as easy as the records are maintained in the handwritten registers.

The proposed methodology for implementing a student monitoring system using chatbot- progress tracking, internship checker, complaints register. Based on the requirements, design the chatbot's conversational flow, user interface, and integration with backend systems. Consider using natural language processing (NLP) techniques to make the chatbot more human-like and improve its understanding of students queries. Student monitoring using chatbot reduces the time of students as they don't have to search for their attendance or grades manually. All the manual difficulties in monitoring the student details in a college have been rectified by implementing computerization.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	1
ABSTRACT	2
LIST OF FIGURES.....	3
1. INTRODUCTION	1
1.1. Existing Systems.....	2
1.1.1 Disadvantages of Existing systems	2
1.2. Objectives.....	2
1.3. Proposed Methods.....	3
2.LITERATURE SURVEY	4
2.1 Design and Implementation of a Chatbot for Student Support Services	4
2.2 Automating Student Management System Using Chatbot &RPA Technology ...	5
2.3 Chatbot for Academic Record Monitoring in Education Institution.....	5
3. ANALYSIS	7
3.1 Gathering Data	8
3.2 Data preparation	8
3.3 Data Wrangling	9
3.4 DataAnalysis	9
3.5 TestModel	9
3.6 Test Model.....	10
3.7 Deployment	10
4. DESIGN OF THE PROJECT	11
4.1. Use Case Diagram	13
4.2. Class Diagram	14
4.3 Activity Diagram	15
4.4 Sequence Diagram.....	16
4.5 Collaboration Diagram.....	17
4.6 Component Diagram.....	18
4.7 Deployment Diagram.....	19

4.8 Architecture Diagram of the Application	19
5. IMPLEMENTATION	21
5.1. Registration Of Students and Faculty	21
5.2. Deletion And Retrieve Details of Students and Faculty	24
5.3. Login And Logout Sessions	24
5.4. Implementation of Attendance Module.....	26
5.5. Marks Module	28
5.5.1 Uploaded File Marks	29
5.5.2 Add and Save Marks	29
5.6. Implementation Of Percentage Action	30
5.7. Implementation Of Internship Module.....	33
5.8 Implementation of Chatbot Module.....	34
6. TESTING	37
6.1 Test Cases.....	38
6.2 Table of Test Cases For User Queries	39
7. RESULTS.....	37
7.1 Home Page	37
7.2 Login Page	37
7.3 Student Page.....	37
7.3.1 Percentage Page	38
7.3.2 Internships Page	38
7.4 Faculty Page	39
7.4.1 Attendance Actions	39
7.4.2 Internship Actions	40
7.4.3 Percentage Page	41
7.5 Admin Page	42
7.6 Chatbot Feature.....	44
7.6.1 Student Chatbot Feature.....	44
7.6.2 Admin Chatbot	45
8. CONCLUSION.....	46
9. LIMITATIONS OF THE PROJECT	49

10. SCOPE OF THE PROJECT50
11. REFERENCES.....

LIST OF FIGURES

S.No	Fig No.	Name of the Figure	Page No
1	4.1	Use case Diagram	13
2	4.2	Class Diagram	14
3	4.3	Activity Diagram	15
4	4.4	Sequence Diagram	16
5	4.5	Collaboration Diagram	17
6	4.6	Component Diagram	18
7	4.7	Deployment Diagram	19
8	4.8	Architecture Diagram	20
9	7.9	Homepage of Project	37
10	7.10	Login Page of role	37
11	7.11	Student login Page	38
12	7.12	Student Percentage Page	38
13	7.13	Student Internship Page	39
14	7.14	Add Attendance Page	39
15	7.15	View Attendance Page	40
16	7.16	Add internship page	40
17	7.17	View Internship page	41
18	7.18	Upload Marks	41
19	7.19	View Percentage in Faculty Page	42
20	7.20	Add Faculty Details	42
21	7.21	Add Student Details	43
22	7.22	View Faculty	43
23	7.23	View Student	44
24	7.24	Student Chatbot	44
25	7.25	Admin chatbot	45
26	7.26	Output of Marks in Chatbot	45
27	7.27	Academic Percentage in chatbot	46
28	7.28	Output of Internship in Chatbot	46

1. INTRODUCTION

The improvements in the fields of inter-networking and information technology have been intricate in executing Artificial Intelligent (AI) systems. These systems are drawing nearer of human activities, for example, choice emotionally supportive networks, robotics, natural language processing, and so forth. Indeed, even in the artificial intelligent fields, there are some hybrid strategies and adaptive techniques that make increasingly complex techniques. That, yet these days there are additionally several Natural Language Processing (NLP) and intelligent systems that could comprehend human language. Artificial intelligent systems learn themselves and retrieve insight by perusing required electronic articles that have been existed on the web.

A chatbot (otherwise called a chatterbox, Bot, or Artificial Conversational Entity) is an AI program that copies human discussions including content and communication in natural language utilizing artificial intelligence methods, for example, Natural Language Processing (NLP), picture and video processing, and voice analysis. Chatbot for college management system has been created utilizing artificial intelligence algorithms that examine the user queries. This chatbot system is an internet application that gives an answer to the broken down queries of an user. Users simply need to choose the classification for inquiries and afterward ask the question to the bot that utilizes for noting it. Artificial intelligence has been incorporated to respond to the user's inquiries. Then the user can procure the fitting solutions to their inquiries.

The appropriate responses are given utilizing artificial intelligence algorithms. Users won't need to go actually to the college or college website for requests. Users need to enlist to the system and needs to login to the system. After login users can get to the different helping pages. There will be different helping pages through which users can chat by asking questions related with college activities. The system answers to users' queries with the assistance of effective Graphical User Interface (GUI). The user can question about the college related activities with the assistance of this web application. College related activities, for example, admissions, academics, Intake, and other social activities. It will support the undergraduates/other user to be refreshed about the college activities. A chatbot is an Artificial Intelligence program that can converse with people in natural language, the manner in which we collaborate with one another. It can trade a human for some undertakings of replying inquiries. A chatbot is a specialist that assists users in utilizing natural language. It was worked as an endeavor to trick people. A few uses of chatbots, for example, User care, customer support and so on utilizes Artificial Intelligence Markup Language (AIML) [3] to visit with users. One of the foremost objectives of chatbots is to take after a smart human and entangle the recipient of the discussion to comprehend the genuine working along with different designs and abilities for their use has generally widened. These chatbots can demonstrate adequate to trick the user to believe that they are "talking" to an individual, however, they are limited in improving their insight base at runtime, and have typically next to zero methods for keeping track of all the discussion information. Chatbots utilize AI to arrive at counterfeit intelligence helping them to comprehend the user question, what's more, give a suitable reaction. The chatbots are created

utilizing the Artificial Intelligence Markup Language (AIML) for imparting or cooperating with the user. This comprises software that will be made up of utilizing Artificial Intelligence and will assist the user in chatting with a machine. The user can ask the systems like typically did to other humans.

1.1. EXISTING SYSTEMS

Traditionally, the chat bot system is not known to people who are not more into the technology. Even if there exist a chat bot system, it is not much accurate in proving the answer or solutions. Students need to manually visit to the college to get their queries answered by the college help desk. This process consumes lot of time as well as money as the students needed to visit college if its miles away from home. Also, this process may lead to communication gap between student and college.

Here the monitoring of students will be carried out in the hand written registers. It will be a tedious job to maintain the record for the user. The human effort is more here. The retrieval of the information is not as easy as the records are maintained in the handwritten registers. This application requires correct feed on input into the respective field. Suppose the wrong inputs are entered, the application resist to work. so, the user find it difficult to use. This is time consuming and has much cost.

1.1.1 Disadvantages of Existing Systems

These apps have limitations in terms of real-world functionality, accuracy, and user experience. Chatbots may not always provide accurate or reliable responses, which could lead to confusion for students. Like any software, chatbots can experience errors and bugs, leading to potential issues in providing accurate and timely information. A chatbot relies on internet access, which could be problematic in areas with poor connectivity or during network outages.

1.2. OBJECTIVES

To reduce unnecessary paperwork in maintaining students information. To provide a computerized platform that will help the organization to monitor the students daily activity and records. To improve the students time management through smart application features. Keeping students engaged and motivated through interactive and dynamic interactions with the chatbot. Test and refine the app's features to ensure that it meets the standards of security and reliability, and to continually update the app to keep up with the latest cybersecurity threats and trends. The chatbot's NLP capabilities can be utilized to continuously enhance its responses and adapt to student queries better over time.

1.3. PROPOSED METHODS

This College Chatbot System is a web-based application which gives responses to the user queries. The system architecture of the chatbot. Firstly, Chatbot responds to the user by greeting him/her and then asks user to login into the system by providing his/her mail. Then the user finds the buttons in the UI which corresponds to the different categories of the college. After going through the buttons the chatbot system asks the user, is it helpful in giving the response. If the user is not able to find the required response, he/she can continue the chat with the college chatbot system by briefly elaborating their queries. Then chatbot system applies Machine Learning algorithms to the break down the user queries.

1.3.1 Internship Checker

It checks whether the student has done any internship or not. And the names of internships if they done.

1.3.2 Academic Progress

Students can inquire about their grades, assignments, and upcoming exams through the chatbot, enabling them to stay updated on their academic performance.

1.3.3 Attendance Tracking

The chatbot can allow students to check and track their attendance percentage.

1.3.4 Chatbot

Chatbot for Student Monitoring System project will be developed using artificial intelligence algorithms(NLP) that will analyze users queries. The system will be a web application which will provide answers to the analyzed queries of the user.

2.LITERATURE SURVEY

A literature survey for a student monitoring system using a chatbot project involves gathering relevant academic papers, research articles, and other publications related to chatbots, student monitoring systems, and their integration. Below is a sample list of topics and potential sources to include in your literature survey:

2.1 Design and Implementation of a Chatbot for Student Support Services

"Design and Implementation of a Chatbot for Student Support Services" by Chen et al. (2019): This paper presents the design and implementation of a chatbot to provide student support services.

This paper introduces the development and deployment of a chatbot aimed at enhancing student support services. The chatbot is designed to address the diverse needs of students, offering them quick and personalized assistance. Leveraging natural language processing and artificial intelligence, the chatbot provides efficient responses to queries related to academic, administrative, and extracurricular matters. The study delves into the architecture and functionalities of the chatbot, emphasizing the user-centric approach adopted during its design. Through rigorous evaluation, the effectiveness and utility of the chatbot are assessed, showcasing promising results in terms of improved student satisfaction and reduced workload on support personnel. Overall, this research contributes to the growing body of knowledge on the application of chatbots in educational settings, emphasizing the potential to revolutionize student support services. In conclusion, this study successfully demonstrates the design and implementation of a chatbot that significantly enhances student support services. By adopting a user-centric approach, the chatbot provides personalized and timely assistance to students, contributing to their overall academic and administrative experience. The combination of natural language processing and artificial intelligence technologies empowers the chatbot to efficiently handle a wide range of student queries, resulting in increased student satisfaction and reduced workload for support personnel. The positive outcomes from the evaluation indicate the potential of chatbots to revolutionize student support services in educational institutions. However, continuous improvements and updates are required to keep the chatbot relevant and adaptive to changing student needs. Future research could focus on expanding the chatbot's capabilities and exploring its integration with other educational systems to further enhance the student experience.

2.2 Automating Student Management System Using Chatbot & RPA

Technology

*Automating Student Management System Using ChatBot and RPA Technology
Proceedings of the 3rd International Conference on Advances in Science & Technology
(ICAST) 2020*

The implementation of an automated student management system using ChatBot and Robotic Process Automation (RPA) technology presents a significant advancement in the realm of educational administration. This innovative approach leverages the capabilities of AI-driven Chatbots and RPA to streamline and enhance various aspects of student management, ultimately leading to increased efficiency, improved user experiences, and optimized resource allocation. In conclusion, the implementation of an automated student management system through the convergence of Chatbot and RPA technologies heralds a new era of educational administration. This amalgamation fosters a user-centric approach, offering students and administrative personnel enhanced accessibility, efficiency, and accuracy. As institutions continue to embrace digital transformation, this innovative solution stands as a testament to the potential of AI-driven automation in redefining the landscape of student management.

2.3 Chatbot For Academic Record Monitoring in Education Institution

Published under licence by IOP Publishing Ltd IOP Conference Series: Materials Science and Engineering, Volume 879, 3rd International Conference on Informatics, Engineering, Science, and Technology (INCITEST 2020) 11 June 2020, Bandung, Indonesia.

Monitoring academic records at a higher education institution is highly needed by both students and parents of students. Although the system is usually already available in the form of a web site, but it is still considered too complicated because it must involve a troublesome authentication process, especially for parents. Nowadays, chat applications have been very widely used by the community both young people and even the elderly. There are many chat applications that are widely used including WhatsApp, LINE, Telegram, and Facebook Messenger. The chat application provides an Application Programming Interface (API) service for sending or receiving messages. Therefore, the API can be used to create applications (chatbot) that will serve users in the form of chat. In this study, chatbot was built using the services of Telegram. In conclusion, the development of a chatbot for academic record monitoring in higher education institutions represents a significant advancement in enhancing administrative efficiency and student support. Through this project, we have recognized the potential of AI driven technology to streamline the management of academic records, reduce manual intervention, and provide students with timely and personalized assistance. The chatbot's ability to provide real-time access to academic information, answer queries, and offer

guidance on academic matters has the potential to alleviate the administrative burden on staff members and empower students to take control of their educational journey. By automating routine tasks such as grade inquiries, course registration, and program requirements, the institution can allocate resources more effectively and improve overall operational effectiveness.

3.

ANALYSIS

Once the user asks query, the keywords in the query is detected using WordNet Algorithm. As the query description can change from one person to another person. The same query may be asked in a different ways by the users. One user asks a query so simply and clearly while another user may request same query in a completely different manner. So it is required to find what is the exact information user seeks to know and to find a correct response for the corresponding user query. The chatbot system firstly removes the stop words from the user input, if they are present in the queries asked by the user. After removing the stop words from the user queries, tokenization and lemmatization process is done.

Tokenization is a process of taking a set of text or text and breaking it up into its individual words or sentences. Lemmatization is the process of gathering the different inflected forms of a word so they can be dissected as a solitary item and is a variation of stemming. From there spell checker is used to identify and rectify spelling mistakes present in the query, then by using the sentence similarity and WordNet Algorithm a suitable response is explored in the knowledge database. WordNet is a semantic and lexical database for the English language. It is used to group English words into the set of synonyms called synsets, it gives short definitions and utilization models, and records various relations among these synonym sets or their individuals. If the response is found in the database it is displayed to the user, else the system notifies the admin about missing response in the database and gives a predefined response to the user. Admin can write the missing response into the database by logging into the admin block in website so that if the user asks the same query next time, he/she may get the suitable response. At the end of conversation the college chatbot system collects the feedback from users to improve the system efficiency.

The functions of the user are to ask queries, provide feedback and so on. All the functions to be performed by the user are outlined below.

After clicking on the chatbot provided in the college website. The chatbot system greets the user and requests the user to provide the mail id. After which the chatbot starts chatting with the user. When the user proceeds to choose chatbot to get an answer to his/her query, the chatbot displays a page to select few options regarding college and identifies his/her category of query. If the user gets his query cleared then the task of chatbot is completed. If the user is not satisfied with rule based response, then the chatbot system requests to enter his/her query in words and the suitable response is given by the chatbot. User's query is first checked in database. If the query is valid then suitable response is given to the user. If the query is invalid then chatbot requests user to ask queries regarding the college.

At the other end, admin who is responsible for maintaining the college chatbot system up to date has several functions to perform such as add the query to the database, modify the data, delete the data, and view feedback given by user and so on. All the functions to be performed by the admin are outlined below.

A Student bot project is built using artificial algorithms that analyzes user's queries and understand user's message. This System is a web application which provides answer to the query of the student. Students just have to query through the bot which is used for chatting. Students can chat using any format there is no specific format the user has to follow. The System uses built in artificial intelligence to answer the query. The answers are appropriate what the user queries. The User can query any college related activities through the system. The user does not have to personally go to the college for enquiry. The System analyzes the question and than answers to the user. The system answers to the query as if it is answered by the person. With the help of artificial intelligence, the system answers the query asked by the students. The system replies using an effective Graphical user interface which implies that as if a real person is talking to the user. The user just has to register himself to the system and has to login to the system. After login user can access to the various helping pages. Various helping pages has the bot through which the user can chat by asking queries related to college activities. The system replies to the user with the help of effective graphical user interface. The user can query about the college related activities through online with the help of this web application. The user can query college related activities such as date and timing of annual day, sports day, and other cultural activities. This system helps the student to be updated about the college activities.

3.1. Gathering Data

Data Gathering is the first step of the machine learning life cycle. The goal of this step is to identify and obtain all data-related problems.

In this step, we need to identify the different data sources, as data can be collected from kaggle such as csv files. It is one of the most important steps of the life cycle. The quantity and quality of the collected data will determine the efficiency of the output. The more will be the data, the more accurate will be the prediction. This step includes the below tasks:

Identify various data sources

Collect data

Integrate the data obtained from different sources

By performing the above task, we get a coherent set of data, also called as a dataset. It will be used in further steps.

3.2. Data preparation

After collecting the data, we need to prepare it for further steps. Data preparation is a step where we put our data into a suitable place and prepare it to use in our machine learning training.

In this step, first, we put all data together, and then randomize the ordering of data.

This step can be further divided into two processes:

□ Data exploration:

It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data.

A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.

□ Data pre-processing:

Now the next step is preprocessing of data for its analysis.

3.3. Data Wrangling

Data wrangling is the process of cleaning and converting raw data into a useable format. It is the process of cleaning the data, selecting the variable to use, and transforming the data in a proper format to make it more suitable for analysis in the next step. It is one of the most important steps of the complete process. Cleaning of data is required to address the quality issues.

It is not necessary that data we have collected is always of our use as some of the data may not be useful. In real-world applications, collected data may have various issues, including:

Missing Values

Duplicate data

Invalid data

Noise

So, we use various filtering techniques to clean the data.

It is mandatory to detect and remove the above issues because it can negatively affect the quality of the outcome.

3.4. Data Analysis

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

Selection of analytical techniques

Building models

Review the result

The aim of this step is to build a machine learning model to analyze the data using various analytical techniques and review the outcome. It starts with the determination of the type of the problems, where we select the machine learning techniques such as Classification. then build the model using prepared data, and evaluate the model.

Hence, in this step, we take the data and use machine learning algorithms to build the model.

3.5. Train Model

Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.

We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

3.6. Test Model

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.

Testing the model determines the percentage accuracy of the model as per the requirement of project or problem

.

3.7. Deployment

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.

If the above-prepared model is producing an accurate result as per our requirement with acceptable speed, then we deploy the model in the real system. But before deploying the project, we will check whether it is improving its performance using available data or not. The deployment phase is similar to making the final report for a project.

4. DESIGN OF THE PROJECT

The Unified Modeling Language (UML) is a standard language for writing software blueprints. The UML is a language for Visualizing, Specifying, Constructing, Documenting the artifacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system.

There are two broad categories of diagrams, and they are again divided into structural diagrams and behavioral diagrams. The structural diagrams represent the static aspect of the system. The four structural diagrams are class diagram, object diagram, component diagram, deployment diagram. Behavioral diagrams basically capture the dynamic aspect of a system. Types of behavioral diagrams are use case diagram, sequence diagram, collaboration diagram, state chart diagram, activity diagram. Some of the frequently used use case diagrams in software development are:

Use Case diagram: Use case is a description of set of sequence of actions that a system performs that yields an observable result of value to actor. Actors are the entities that interact with a system. Although in most cases, actors used to represent the users of system, actors can be anything that needs to exchange information with the system. So, an actor may be people, computer hardware, other systems, etc.

Activity diagram: An activity diagram is a special case of state diagram. An activity diagram is like a flow Machine showing the flow a control from one activity to another. An activity diagram is used to model dynamic aspects of the system. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

Class diagram: In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram. It describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Sequence diagram: A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. Sequence diagram used lifeline which is a named element which depicts an individual participant in a sequence diagram. Communications happens as the messages appear in a sequential order on the lifeline.

Component diagram: A component diagram in UML illustrates the structure and relationships of software components within a system. Components are depicted as rectangles, and relationships are represented using connectors. It focuses on component organization and dependencies, showcasing how components interact and communicate. Ports and component instances can also be included. The diagram provides a high-level view of the system's component architecture.

Deployment diagram: A deployment diagram in UML illustrates the physical architecture of a system by showing how software components and hardware resources are distributed and interconnected. It uses nodes to represent hardware devices or execution environments and artifacts to represent software components. Communication paths indicate the channels through which nodes interact. The diagram provides a highlevel view of the system's deployment environment, facilitating understanding of software and hardware deployment relationships.

4.1 USE CASE DIAGRAM

In this use case diagram, we have actors like admin, Faculty, chatbot and student. The operations of admin and faculty are similar but there is small difference that the faculty can add the attendance, internship and can view their data, where the operations of student are limited and can view his data only. Chatbot is a feature where can recognize the query or speech that user enter or speaks and retrieve the details based on the query.

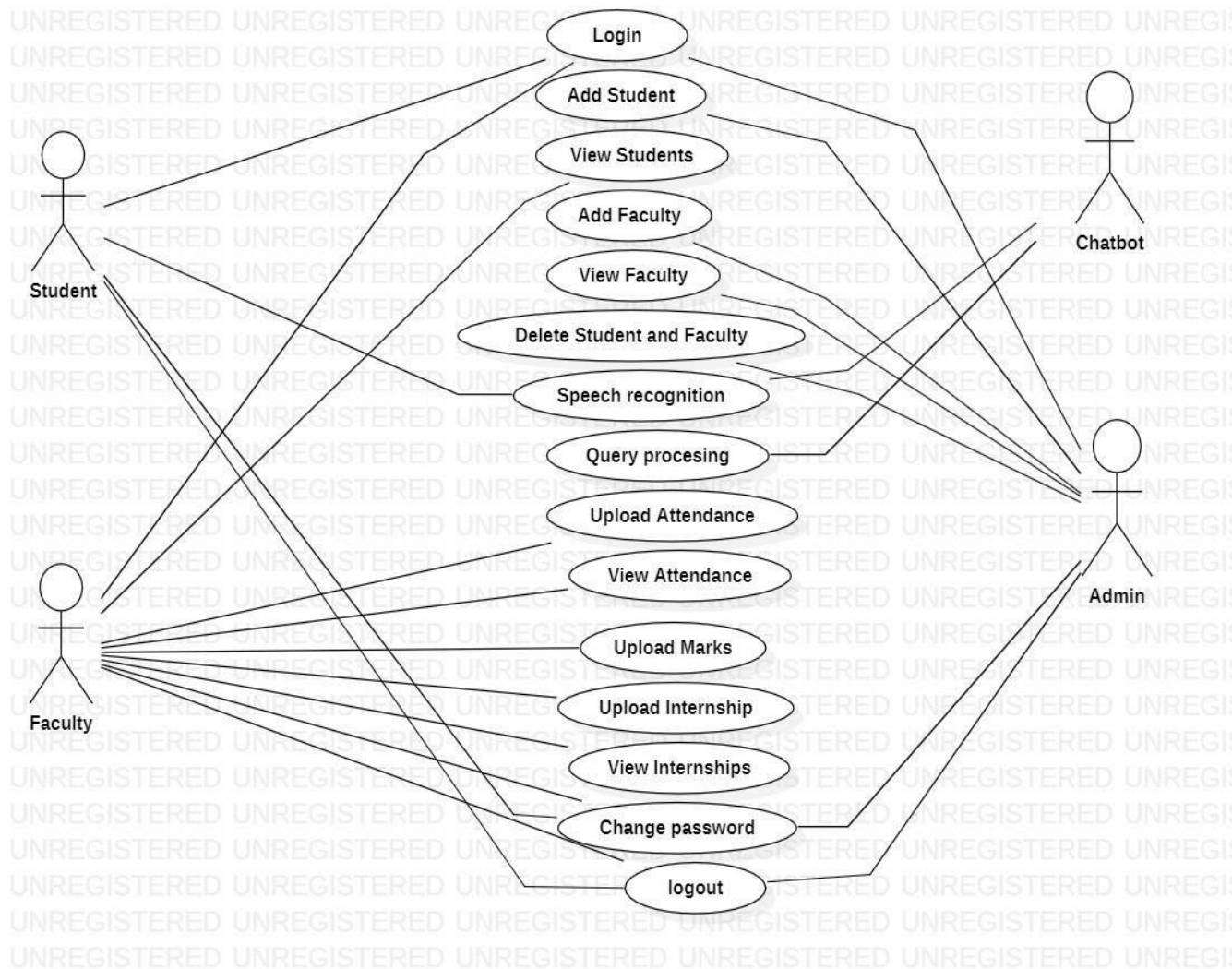


Fig 4.1 Use Case Diagram

4.2 CLASS DIAGRAM

In this class diagram , there are some classes which are dependent on each other and performs the actions sequentially, where the chatbot class recognizes the text or query which is entered by users and it retrieve the details from the database which is related to the query.

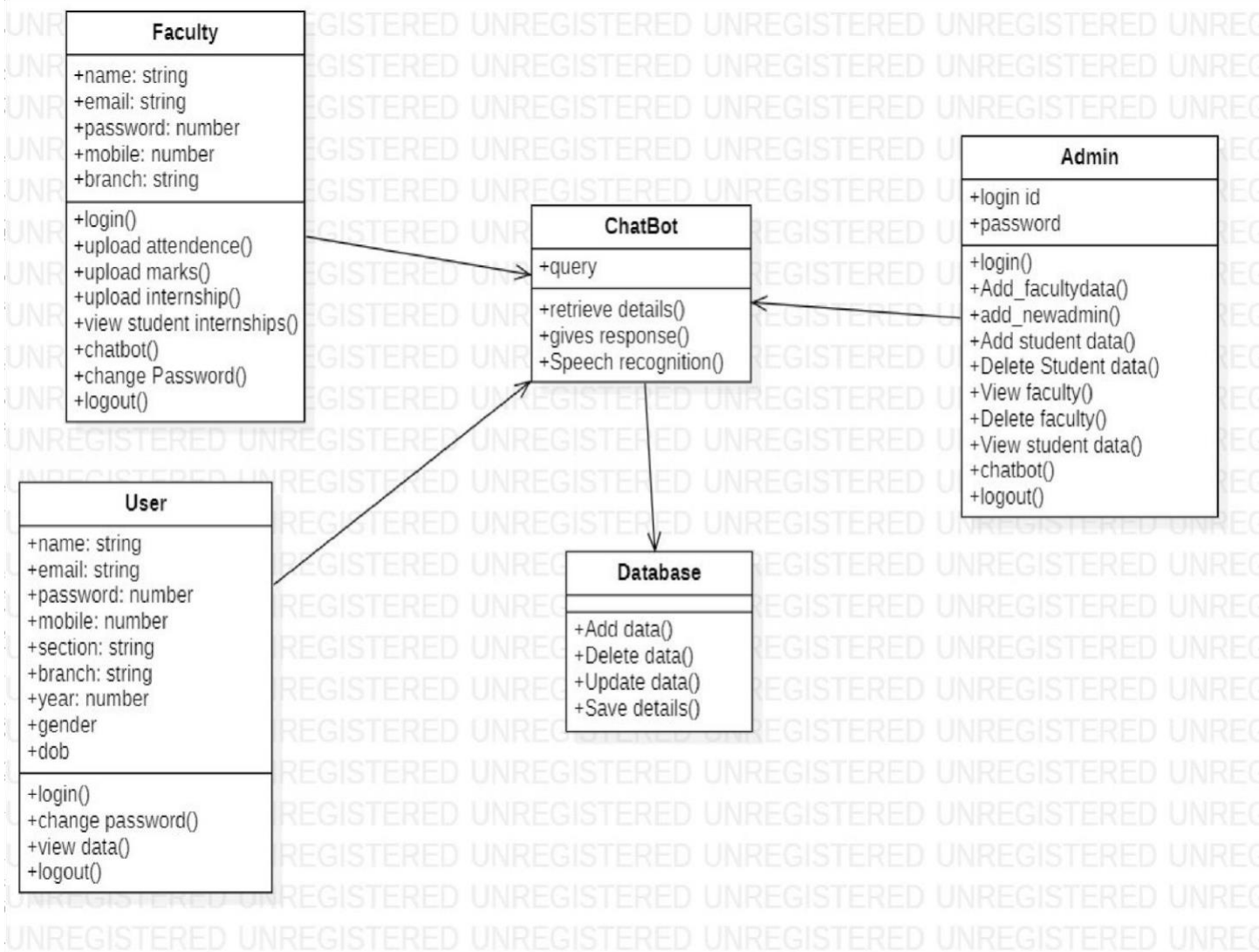
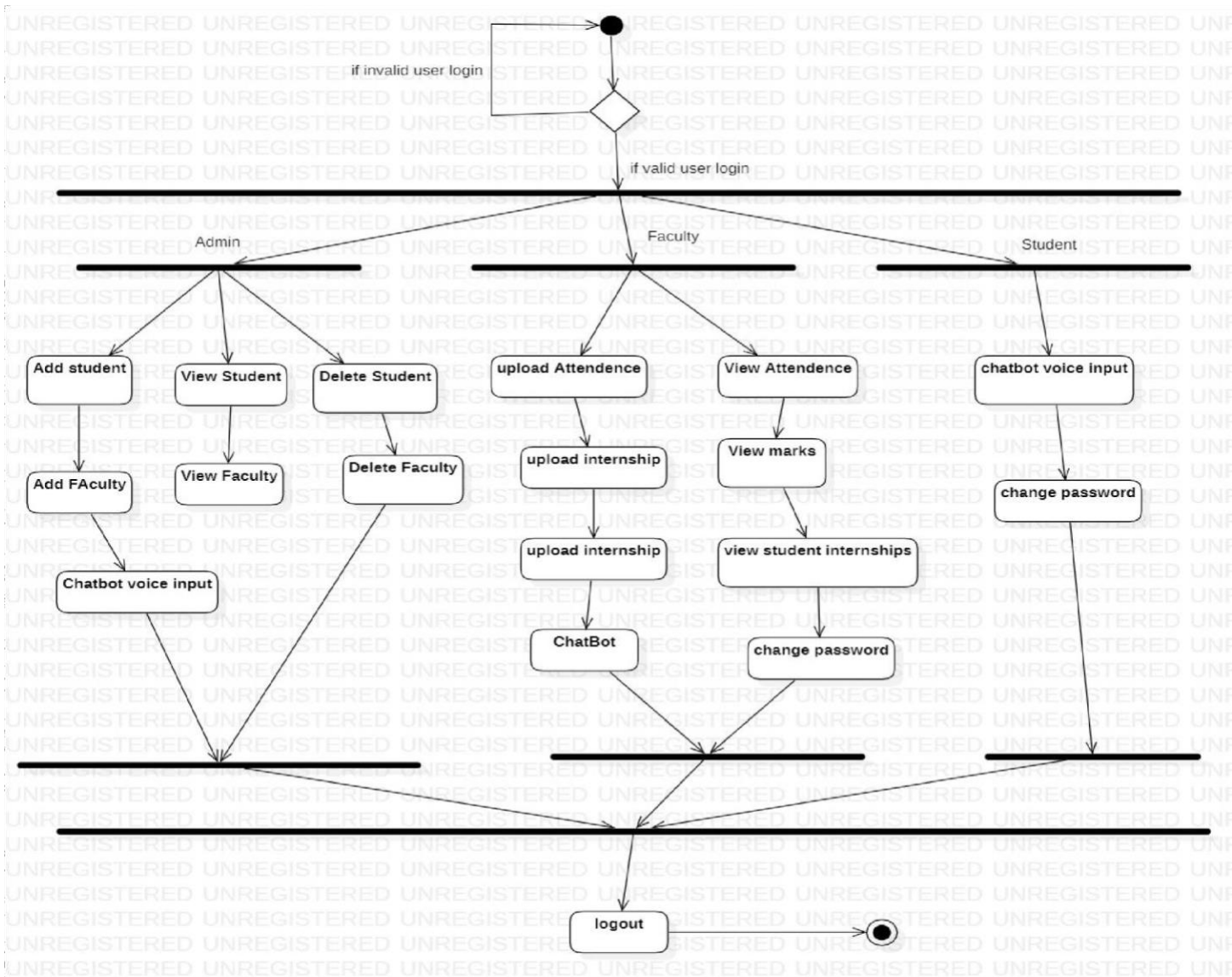


Fig 4.2 Class Diagram

4.3 ACTIVITY DIAGRAM

The activity diagram for the Student Monitoring System using a chatbot project visually depicts the sequence of activities and interactions within the system. Beginning with "User Interaction," where input is provided to the chatbot, the diagram follows a dynamic path encompassing "Process User Input" for understanding intent. Based on the user's aim, the system branches into routes like "Retrieve Student Data" for fetching information from the database or "Update Student Records" for making changes. This is complemented by the potential for generating "Reports/Analytics" if insights are sought. The diagram culminates with "End Interaction," marking the user's session conclusion and preparing the system for the next



input. Overall, the diagram illustrates how the system manages user interactions, providing a comprehensive perspective on the intricate processes of student information handling and enhancing the monitoring experience.

Fig 4.3 Activity Diagram

4.4 SEQUENCE DIAGRAM

The sequence diagram for the Student Monitoring System using a chatbot with NLP shows how users, the chatbot, and the system work together. NLP helps the chatbot understand user messages and intents. Users start by talking to the chatbot, which uses NLP to figure out what they want. The system then uses NLP insights to decide if users need student info or updates. If it's info, the system gets data from the database and shows it through the chatbot. NLP also handles follow-up questions. For updates, users guide changes, which the system checks and saves in the database. NLP is key for clear communication. Reports can be requested too. When users finish, the system waits for the next chat, showing how NLP makes student monitoring smooth and efficient through the chatbot.

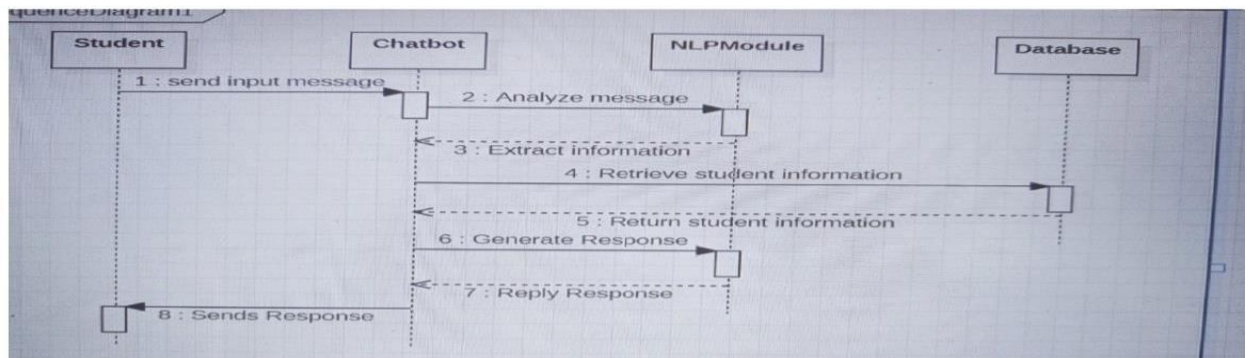


Fig 4.4 Sequence Diagram

4.5 COLLABORATION DIAGRAM

In this collaboration diagram, the entities work cohesively, driven by NLP, to ensure effective student monitoring. Users interact with the Chatbot, which employs NLP to interpret their intent. The System coordinates with various entities to handle data retrieval, updates, and analytics, ensuring a streamlined user experience. This collaboration underscores the integral role of NLP and the orchestrated interactions among components, enabling clear communication and efficient student monitoring within the chatbot project.

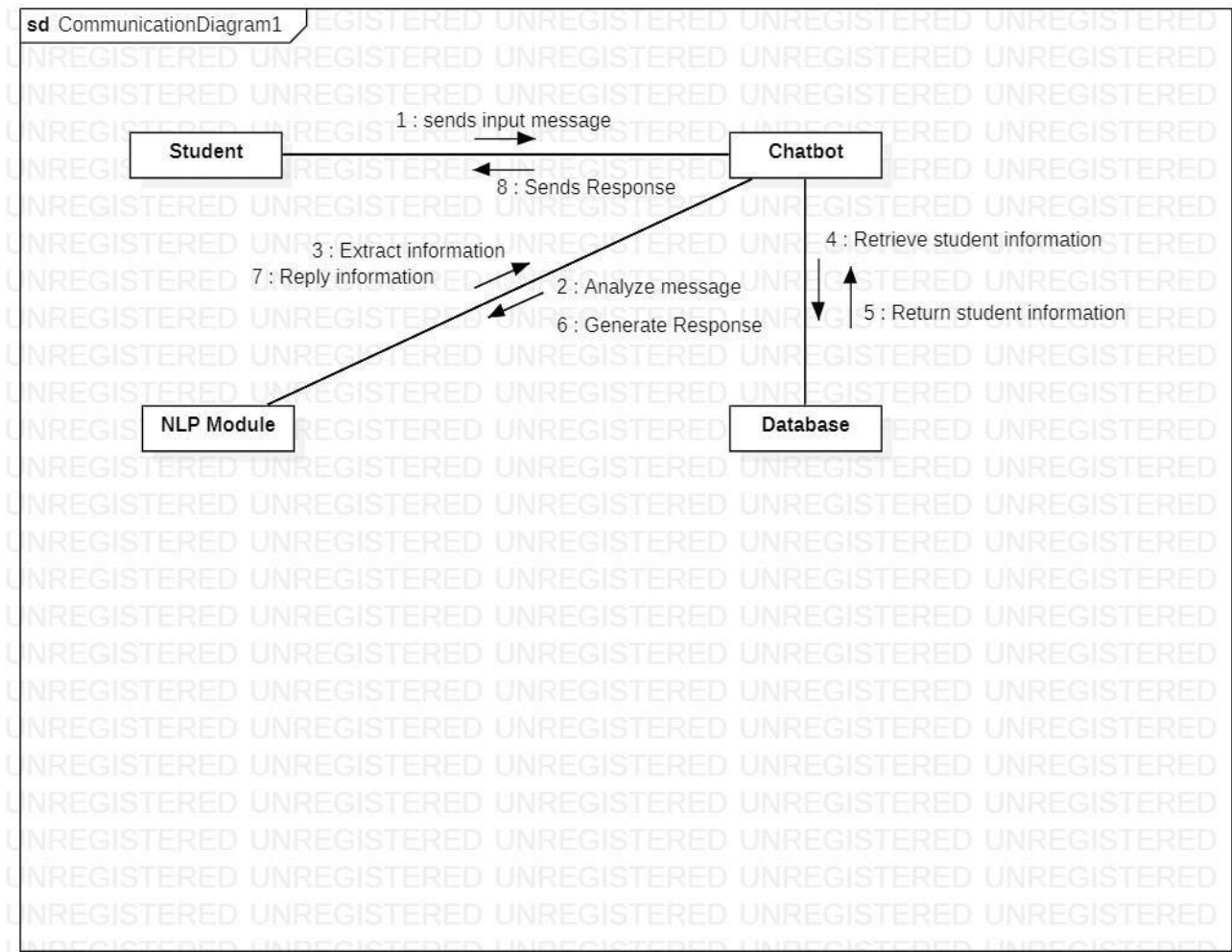


Fig 4.5 Collaboration Diagram

4.6 COMPONENT DIAGRAM

The component diagram for the Student Monitoring System utilizing a chatbot with Natural Language Processing (NLP) portrays the structural arrangement of distinct system components and their interactions. It illustrates the interconnection of elements such as the "User Interface" that facilitates user engagement, the "NLP Engine" responsible for processing and understanding user input, and the "Database Interface" managing student data storage and retrieval

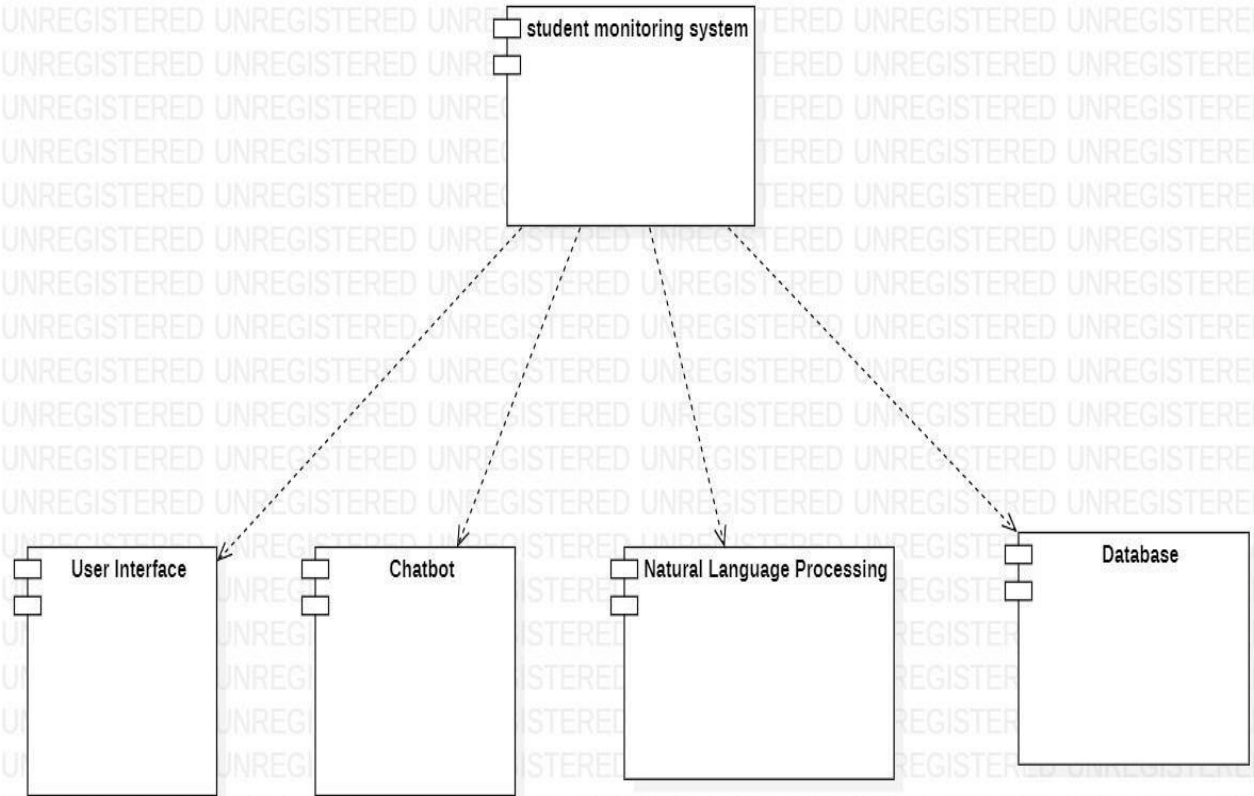


Fig 4.6 Component Diagram

4.7 DEPLOYMENT DIAGRAM

The deployment diagram provides a clear picture of how different parts of the Student Monitoring System are set up and connected, either on real-world hardware or in the cloud. It shows how everything works together, from the user's devices to the chatbot's logic, database, and other services. This diagram helps us see how the system is organized, how data moves between components, and how the system can handle more users as needed.

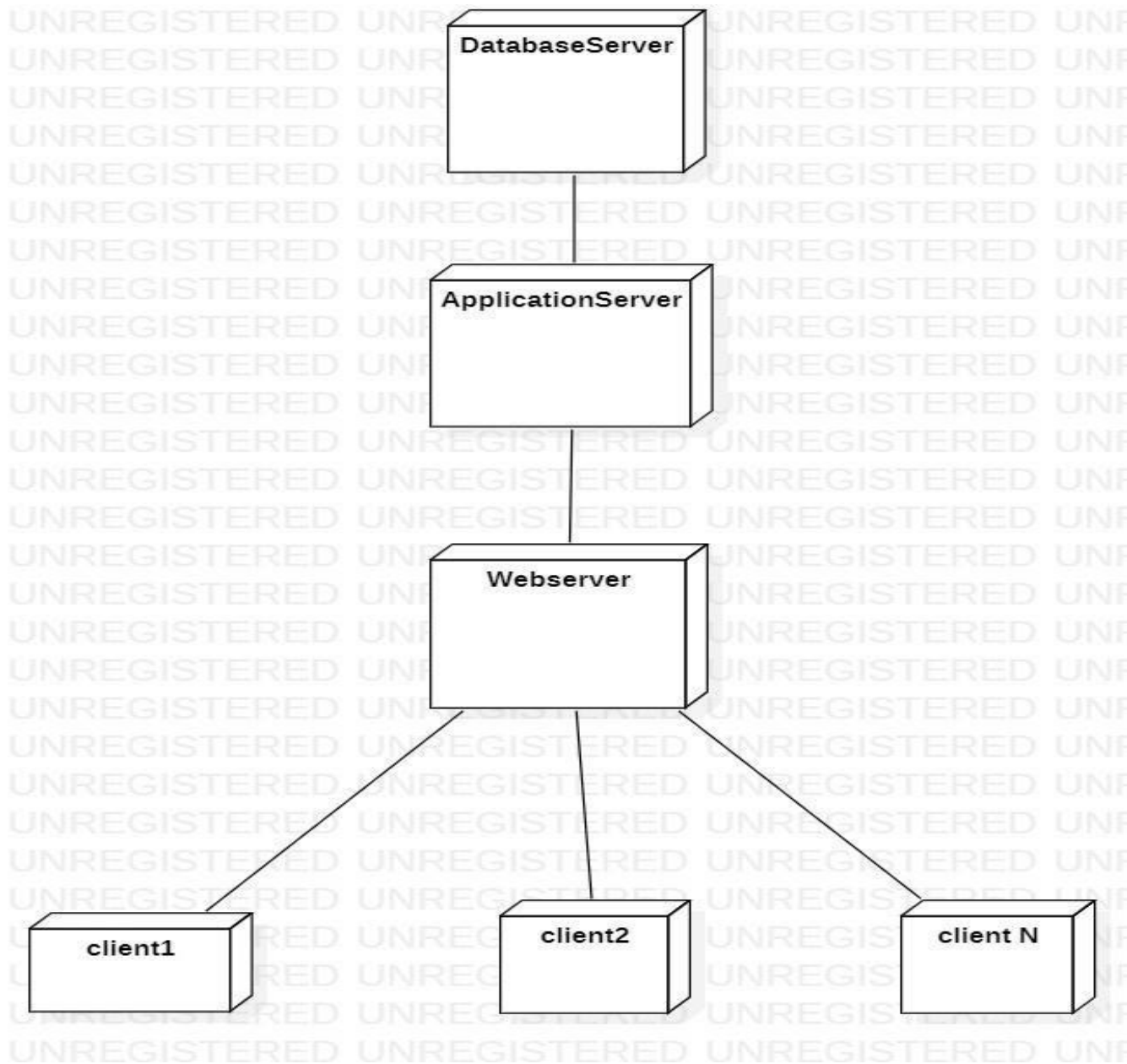


Fig 4.7 Deployment Diagram

4.8 ARCHITECTURE DIAGRAM OF THE APPLICATION

The diagram represents the high-level view of the system's architecture, illustrating the interactions, dependencies, and communication channels between different components.

User interaction begins when the user engages with a chat interface, inputting queries or requests. As the user provides input, the system processes the information, analyzing the context and discerning the user's intent. Once the intent and context are identified, if the user's query pertains to student information, the system extracts the relevant data from its database. Subsequently, the requested student information is presented to the user in a clear format.

The system's capabilities extend beyond basic data presentation; it adeptly manages additional user requests, catering to inquiries for further details or updates. In cases where modifications to student records are requested, the system efficiently processes the changes and ensures that the database reflects the updated information accurately. Furthermore, the system can offer insightful reports and analytics based on the student data, providing valuable insights such as attendance patterns, grades, and more.

As the user's needs are addressed, the interaction culminates, marking the conclusion of the session. The system remains poised for subsequent inputs, prepared to engage with users in a seamless and responsive manner.

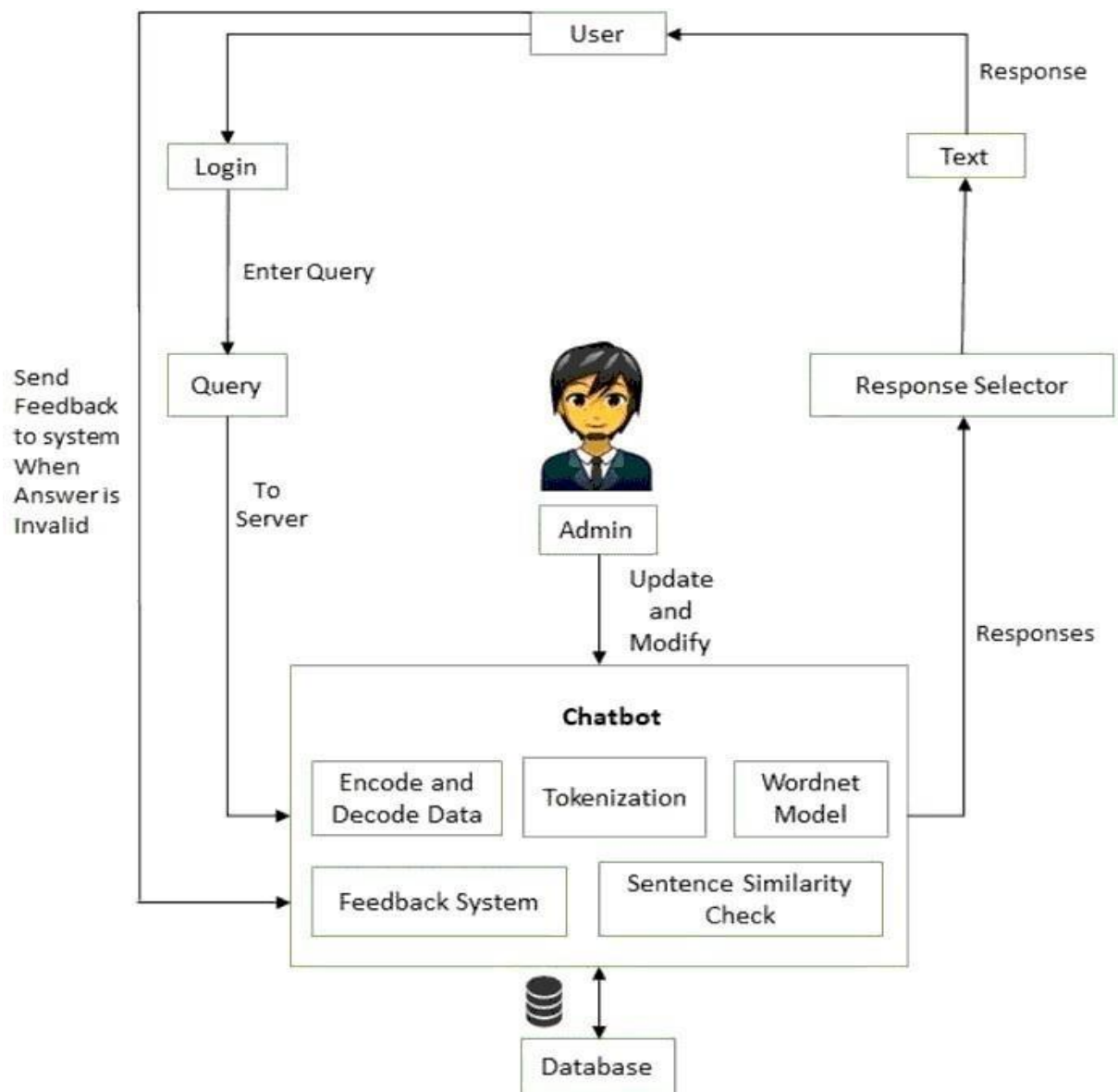


Fig 4.8 Architecture Diagram of the Application

5.

IMPLEMENTATION

The implementation of the project contains various technologies and several dependencies. The project is developed in the form of modules. These modules are integrated to give a successful feature of the application. However, we will discuss only the core features and its module/code/method that is responsible for the functioning of that particular feature.

5.1 REGISTRATION OF STUDENTS AND FACULTY

```
# Import
necessary
libraries and
modulesimport
csv
import datetime
from django.shortcuts import render

# Import forms and
models from the chatbot
appfrom chatbot.forms
import FacultyForm
from
chatbot.models
import
FacultyModel#
Import smtplib for
sending emails
# Import os for file and directory operations

# Import functions from the
chatbot service modulefrom
```

```

# Check if the form data is valid
if registrationForm.is_valid():

    # Create a new instance of the FacultyModel to store registration data
    regModel = FacultyModel()

    # Populate the model fields with data from the form
    regModel.name = registrationForm.cleaned_data["name"]
    regModel.email = registrationForm.cleaned_data["email"]
    regModel.mobile = registrationForm.cleaned_data["mobile"]
    regModel.department = registrationForm.cleaned_data["department"]
    regModel.username = registrationForm.cleaned_data["username"]
    regModel.password = registrationForm.cleaned_data["password"]

    # Check if a user with the same username already exists
    user = FacultyModel.objects.filter(username=regModel.username).first()

    # If user already exists, provide an appropriate message
    if user is not None:
        return render(request, 'facultyregistration.html', {"message": "User Already Exists"})
    else:
        try:
            # Save the registration data to the database
            regModel.save()
            return render(request, 'facultyregistration.html', {"message": "Faculty Added Successfully"})
        except:
            return render(request, 'facultyregistration.html', {"message": "Registration Failed"})
    else:
        return render(request, 'facultyregistration.html', {"message": "Invalid Form"})

    # If the request method is not POST, render the registration form page with a message
    return render(request, 'facultyregistration.html', {"message": "Invalid Request"})
# Define the view function for student registration
def studentregistration(request):

    # Check if the HTTP request method is POST

```



```

if request.method == "POST":

    # Create an instance of the StudentForm using the data from the request
    registrationForm = StudentForm(request.POST)

    # Check if the form data is valid
    if registrationForm.is_valid():

        # Create a new instance of the StudentModel to store registration data
        regModel = StudentModel()

        # Populate the model fields with data from the form
        regModel.name = registrationForm.cleaned_data["name"]
        regModel.email = registrationForm.cleaned_data["email"]
        regModel.mobile = registrationForm.cleaned_data["mobile"]
        regModel.department = registrationForm.cleaned_data["department"]
        regModel.username = registrationForm.cleaned_data["username"]
        regModel.password = registrationForm.cleaned_data["password"]
        regModel.year = registrationForm.cleaned_data["year"]
        regModel.section = registrationForm.cleaned_data["section"]
        regModel.status = "no"
        regModel.regulation = registrationForm.cleaned_data["regulation"]

        # Check if a user with the same username already exists
        user = StudentModel.objects.filter(username=regModel.username).first()

        # If user already exists, provide an appropriate message
        if user is not None:
            return render(request, 'studentregistration.html', {"message":
                "User Already Exists"})
        else:
            try:
                # Save the registration data to the database
                regModel.save()
                return render(request, 'studentregistration.html', {"message":
                    "Student Added Successfully"})
            except:

```

```
        return render(request,
'studentregistration.html', {"message": "Registration
Failed" })
    else:
        return render(request, 'studentregistration.html', {"message": "Invalid
Form" })
```

5.2 Deletion and Retrieve Details of Student and Faculty

```
# Define view functions for deleting student and faculty records
def deletestudent(request):
    # Get the student ID from the request parameters
    student = request.GET['studentid']

    # Delete the student record with the specified ID
    StudentModel.objects.get(id=student).delete()

    # Render the students page with the updated list of students
    return render(request, 'students.html', {'students': StudentModel.objects.all()})

def deletefaculty(request):
    # Get the faculty ID from the request parameters
    faculty = request.GET['facultyid']

    # Delete the faculty record with the specified ID
    FacultyModel.objects.get(id=faculty).delete()

    # Render the faculties page with the updated list of faculties
    return render(request, 'facultys.html', {'facultys': FacultyModel.objects.all()})

# Define view functions for retrieving lists of faculties and students
def getfacultys(request):
    # Render the faculties page with the list of all faculties
    return render(request, "facultys.html", {"facultys": FacultyModel.objects.all()})

def getstudents(request):
    # Render the students page with the list of all students
```

```
return render(request, "students.html", {"students":
```

5.3 Login and Logout Sessions

```
# Define the view function for user login
def login(request):

    # Get the username, password, and user type from the request parameters
    uname = request.GET["username"]
    upass = request.GET["password"]
    type = request.GET["type"]

    # Check the user type and perform corresponding authentication
    if type in "admin":
        if uname == "admin" and upass == "admin":
            # Set session variables for admin user and render faculties page
            request.session['username'] = "admin"
            request.session['role'] = "admin"
            return render(request, "facultys.html", {"facultys": FacultyModel.objects.all()})
        else:
            # Render the index page with an invalid credentials message
            return render(request, 'index.html', {"message": "Invalid Credentials"})

    if type in "student":
        # Filter student records based on username, password, and status
        student = StudentModel.objects.filter(username=uname, password=upass,
        status="yes").first()

        if student is not None:
            # Set session variables for student user and render view attendance page
            request.session['username'] = uname
            request.session['role'] = "student"
            return render(request, 'viewattendance.html')
        else:
            # Render the index page with an invalid username and password message
```

```

        return render(request, "index.html", {"message": "Invalid Username and
        Password"})

    if type in "faculty":
        # Filter faculty records based on username and password
        faculty = FacultyModel.objects.filter(username=username, password=password).first()

        if faculty is not None:
            # Set session variables for faculty user and render add attendance page
            request.session['username'] = username
            request.session['role'] = "faculty"
            return render(request, 'addattendance.html')
        else:
            # Render the index page with an invalid username and password message
            return render(request, 'index.html', {"message": "Invalid Username and
            Password"})

# Define the view function to activate or deactivate a student account
def activateAccount(request):

    # Get the username and status from the request parameters
    username = request.GET['username']
    status = request.GET['status']

    # Update the status of the student record with the specified username
    StudentModel.objects.filter(username=username).update(status=status)

    # Render the students page with the updated list of students
    return render(request, "students.html", {"students": StudentModel.objects.all()})

# Define the view function to handle user logout
def logout(request):
    try:
        # Delete the 'username' session variable if it exists
        del request.session['username']
    except:
        pass

    # Render the index page

```

```
return render(request, 'index.html', {})
```

5.4 IMPLEMENTATION OF ATTENDANCE MODULE

```
# Define the view
function to add
attendance
def
addattendance(request):

    # Get the department, year, semester, section, regulation, and subject
    # from the request parameters
    department = request.GET["department"]
    year = request.GET["year"]
    semester = request.GET["semester"]
    section = request.GET["section"]
    regulation = request.GET["regulation"]
    subject = request.GET["subject"]

    # Get the list of students from the database
    students = StudentModel.objects.filter(department=department, year=year, semester=semester, section=section, regulation=regulation, status='yes')

    # Get the list of usernames from the request parameters
    username_list = request.GET.getlist('username')

    # Define session variables for
```

```

section = request.session['section']
regulation = request.session["regulation"]
sem = request.session['sem']
subject = request.session['subject']

# Debugging: Print the retrieved session variables
print(department, year, section, regulation, sem, subject)

# Iterate through students and update attendance records
for student in StudentModel.objects.filter(department=department, year=year, section=section,
regulation=regulation, status='yes'):

    print("in for ", student.username, username_list)

    # Create an AttendanceModel instance
    attendance = AttendanceModel()
    attendance.username = student.username
    attendance.date = datetime.datetime.now()
    attendance.department = department
    attendance.year = year
    attendance.semester = sem
    attendance.section = section
    attendance.subject = subject

    # Check if the student's username is in the submitted list
    if student.username in username_list:
        attendance.isattended = "yes"
    else:
        attendance.isattended = "no"
        # send_email(student.email, "Your Child is Not Attended to day Class")

    # Save the attendance record
    attendance.save()

# Render the addattendance.html template with a success message
return render(request, "addattendance.html", {"message": 'Attendance Submitted Successfully'})

```

5.5 MARKS MODULE

Below code represents the operations that are performed by the user.

5.5.1 Uploaded File Marks

```
# Define a function to
handle an uploaded
filedef
handle_uploaded_file(
f):
    # Construct the destination path for the
    uploaded file destination_path =
    os.path.join(PROJECT_PATH, "uploads",
    f.name)# Open the destination file for writing
    in binary mode ('wb+')
    with
```

5.5.2 Add and Save Marks

```

# Define a function to save marks from a CSV file
def savemarks(filename):
    # Open the CSV file for reading
    with open(PROJECT_PATH + "\\uploads\\" + filename, mode="r") as f:
        reader = csv.reader(f)
        # Iterate through each line in the CSV file
        for line in reader:
            # Create a MarksModel instance
            marks = MarksModel()
            # Assign values from the CSV line to the MarksModel instance
            marks.halticket_number = line[0]
            marks.Subject_Code = line[1]
            marks.Subject_Name = line[2]
            marks.Internal_Marks = line[3]
            marks.ExterNal_Marks = line[4]
            marks.Total = line[5]
            marks.Credits = line[6]
            marks.year_sem = line[7]
            marks.actualmarks = line[8]
            # Save the MarksModel instance to the database
            marks.save()

# Define a view function to handle adding marks from a CSV file
def addmarks(request):
    # Create a MarksForm instance with POST data and uploaded file
    marksForm = MarksForm(request.POST, request.FILES)
    # Check if the form is valid
    if marksForm.is_valid():
        print("in if")
        # Get the uploaded file from the form
        file = marksForm.cleaned_data['file']
        # Call the function to handle the uploaded file
        handle_uploaded_file(file)
        # Call the function to save marks from the CSV file
        savemarks(file.name)
    else:
        print("in else")

    # Render the addmarks.html template with a success message
    return render(request, "addmarks.html", {"message": "uploaded successfully"})

```

5.6 IMPLEMENTATION OF PERCENTAGE ACTION


```
# Function to calculate and display the percentage of marks for  
a specific studentdef viewstudentpercentageaction(request):
```

```
    print("in fun")
```

```
    # Check if the  
    user role is  
    "faculty" if  
    request.session['  
    role'] ==  
    "faculty":
```

```
        print("in faculty")  
        # Get the username parameter  
        from the GET requestusername  
        = request.GET["username"]
```

```
        # Initialize a dictionary to store percentage  
        informationpercentagedict = dict()
```

```
        # Initialize variables to keep track of total and  
        actual markstotal_marks = 0  
        actual_marks = 0
```

```
        # Iterate through the marks associated with the  
        provided username for marks in
```

```

print("in else")

# Get the username from the session
username = request.session["username"]

# Initialize a dictionary to store percentage information
percentagedict = dict()

# Initialize variables to keep track of total and actual marks
total_marks = 0
actual_marks = 0

# Iterate through the marks associated with the student's username
for marks in MarksModel.objects.filter(halticket_number=username):
    total_marks = total_marks + int(marks.Total)
    actual_marks = actual_marks + int(marks.actualmarks)

# Calculate and store the percentage in the dictionary if actual_marks is not zero
if actual_marks != 0:
    percentagedict.update({username: (total_marks / actual_marks) * 100})

# Render the "viewstudentpercentage.html" template with the calculated percentages
return render(request, "viewstudentpercentage.html", {"percentage": percentagedict})
return

# Function to view the percentage of marks for all students in a specific department, year, section,
and regulation
def viewpercentage(request):

    print("in fun")
    # Check if the user role is "faculty"
    if request.session['role'] == "faculty":

        print("in faculty")

        # Get the department, year, section, and regulation parameters from the GET request
        department = request.GET["department"]
        year = request.GET["year"]

```

```

section = request.GET["section"]
regulation = request.GET["regulation"]

# Initialize a dictionary to store student percentages
percentagedict = dict()

print(department, year, section, regulation)

# Iterate through students in the specified criteria
for student in StudentModel.objects.filter(department=department, year=year,
section=section, regulation=regulation):
    total_marks = 0
    actual_marks = 0

    print("in for")

    # Iterate through marks associated with the student's username
    for marks in MarksModel.objects.filter(halticket_number=student.username):

        print("in marks")

total_marks =total_marks+ int(marks.Total)
    actual_marks = actual_marks + int(marks.actualmarks)

    # Calculate the student's percentage and update the dictionary
    percentagedict.update({student.username: (total_marks / actual_marks) * 100})

    # Render the "viewpercentage.html" template with the calculated student percentages
    return render(request, "viewpercentage.html", {"percentage": percentagedict})
else:
    print("in not faculty")

# Function to render the "viewstudentwisepercentage.html" template
def viewstudentpercentage(request):
    return render(request, "viewstudentwisepercentage.html")

```

5.7 IMPLEMENTATION OF INTERNSHIP MODULE

```
# Function to add an internship record for a student
def addinternship(request):
    # Save the internship record with the provided halticket_number (username) and
    Company_Name
    InternshipModel(halticket_number=request.GET['username'],
    Company_Name=request.GET['company']).save()
    # Render the "addinternship.html" template with a success message
    return render(request, "addinternship.html", {"message": "Added Successfully"})
# Function to render the "viewinternship.html" template
def viewstudentinternship(request):
    return render(request, "viewinternship.html")
# Function to view student-wise internship records
def viewstudentinternshipaction(request):
    print("in fun")
    if request.session['role'] == "faculty":
        print("in if")
        username = request.GET["username"]
        print(username)
        # Retrieve and display internship records for the specified student
        internships = InternshipModel.objects.filter(halticket_number=username)
        return render(request, "viewstudentwiseinternship.html", {"internships": internships})
    elif request.session['role'] == "student":
        # Retrieve and display internship records for the logged-in student
        username = request.session["username"]
        internships = InternshipModel.objects.filter(halticket_number=username)
        return render(request, "viewstudentwiseinternship.html", {"internships": internships})
```

5.8 IMPLEMENTATION OF CHATBOT MODULE

```

# Function to handle user input and generate bot responses
def get_bot_response(request):
    try:
        # Get user input from the request
        userText = request.GET['msg']
        print("in bot response", userText)
        # Check if the user is a faculty or admin
        if request.session['role'] in "faculty" or request.session['role'] in "admin":
            print("in bot response faculty ")
            # Handle different user queries
            if "marks" in userText:
                # Extract student roll number and semester from user input
                tokens = userText.split(" ")
                rno = tokens[1]
                sem = tokens[2]
                # Return academic marks information for the specified student and semester
                return render(request, "chat.html", {"academic_marks": getstudentsemmarks(rno, sem),
"type": "marks"})
            elif "sempercentage" in userText:
                # Extract student roll number and semester from user input
                tokens = userText.split(" ")
                rno = tokens[1]
                sem = tokens[2]
                # Return semester percentage for the specified student and semester
                return render(request, "chat.html",
                    {"percentage": getstudentsempercentage(rno, sem), "type": "percentage"})
            # Handle other queries similarly
            else:
                return render(request, "chat.html", {"message": "sorry i didn't understand please try again"})
        # Check if the user is a student
        elif request.session['role'] in 'student':
            print("in bot response student ")
            # Handle different user queries for students
            if "marks" in userText:
                # Extract semester from user input
                tokens = userText.split(" ")
                sem = tokens[1]
                # Return academic marks information for the logged-in student and specified semester

```

```

        return render(request, "chat.html",
                        {"academic_marks":
                         getstudentsemmarks(request.session['username'], sem),
                         "type":
"marks"}})
    elif "sempcentage" in userText:
        #
        Extrac
        t
        semest
        er
        from
        user
        input
        tokens
        =
        userTe
        xt.split
        (" ")
        sem = tokens[1]
        # Return semester percentage for the logged-in student
        and specified semesterreturn render(request,

```

6. TESTING

Testing of a student monitoring system utilizing a chatbot project is a crucial phase in ensuring the system's reliability and effectiveness. This process involves evaluating the chatbot's various functionalities and interactions within the context of student-related tasks. Test cases encompass scenarios such as querying attendance percentages, calculating GPA's, internship details and providing information about campus events. Through a systematic approach of both positive and negative scenarios, testers verify that the chatbot responds accurately and appropriately, delivering the expected information or guidance to users. This testing phase also addresses potential pitfalls, such as handling missing data, gracefully managing non-existent classes, and accurately calculating grades.

6.1 TEST CASES

Test Case Id	Description	Expected Outcome	Actual Outcome	Status
TC001	User login with valid Credentials	User logged in Successfully	User is able to log in	Pass
TC002	User login with invalid credentials	User receives an error message and cannot log in	User is unable to log in	Pass
TC003	User inquires about their current GPA	Chatbot provides the correct GPA information	Chatbot provides outdated GPA information	Fail
TC004	Test the chatbot's ability to retrieve student attendance	Chatbot displays the student's attendance record.	Chatbot displays the student's attendance record.	Pass
TC005	Verify the chatbot's response when a student asks about their grades.	Chatbot displays the student's recent grades.	Chatbot displays the student's recent grades.	Pass

TC006	Test the chatbot's ability to retrieve student internship details.	Chatbot displays the student's Internship details.	Chatbot displays the student's Internship details.	Pass
TC007	Verify that the user able to logout from the website or not.	User Successfully logged out	User able to logout	Pass

6.2 Table of Test Cases for User Queries Format

Test Case Id	User Query	Expected Outcome	Actual Outcome	Status
TC001	Display roll no semester marks of Student?	Chatbot retrieve the marks of student of particular semester.	Chatbot able to retrieve the marks of student.	Pass
TC002	What is marks of roll no semester of student?	Chatbot retrieve the marks of student of particular semester.	Chatbot not able to retrieve the marks of student.	Fail
TC003	Get roll no semester attendance of student?	Chatbot retrieve the attendance of student of particular semester.	Chatbot able to retrieve the attendance of student.	Pass
TC004	Display roll no current attendance of student?	Chatbot retrieve current the attendance of student.	Chatbot able to show outdated attendance.	Fail
TC005	Display roll no internships of student?	Chatbot retrieves the information	Chatbot able to retrieve the details of	Pass

		about the student internship details	student internships.	
TC006	Get details of student roll no internships?	Chatbot retrieves the information about the student internship details	Chatbot not able to show the details of student internships	Fail
TC007	Get roll no academic overallpercentage of student?	Chatbot retrieve the academic percentage of two particular semesters	Chatbot able to retrieve the academic percentage.	Pass

7. RESULTS

7.1 HOME PAGE



Fig 7.9 Homepage of Project

7.2 LOGIN PAGE

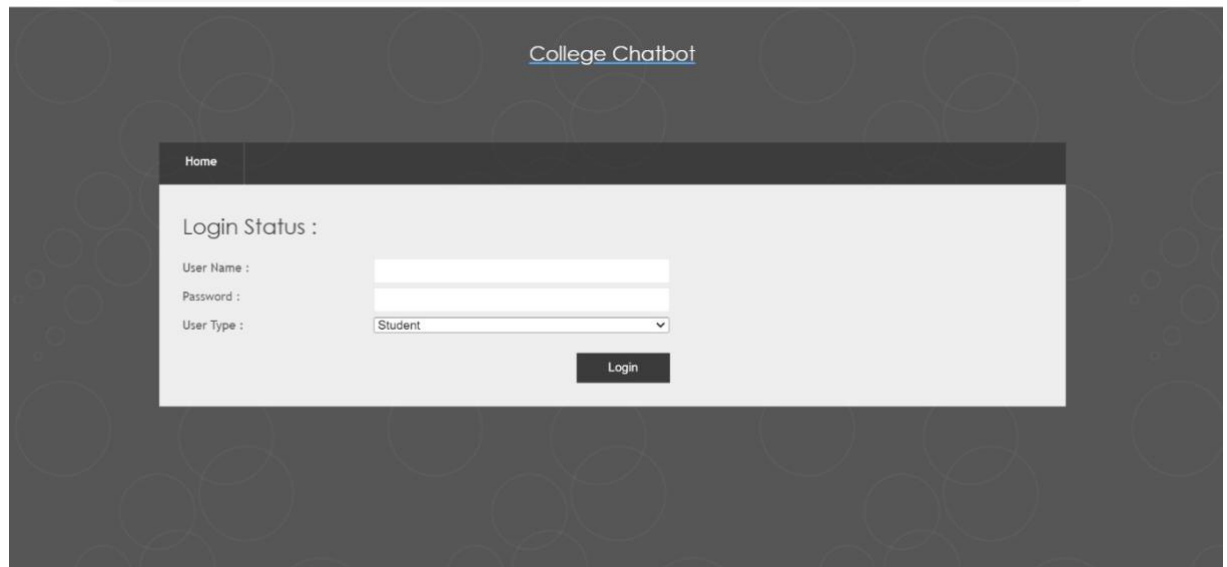


Fig 7.10 Login Page of role

7.3 STUDENT PAGE

The website is tested and all the operations of the website is working fine.

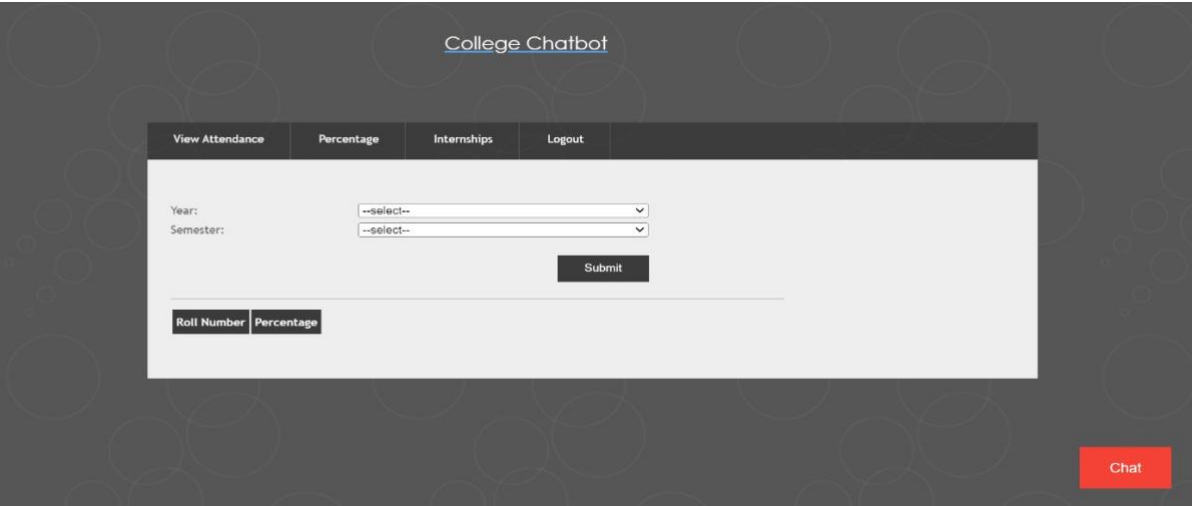


Fig 7.11 Student Page

7.3.1 Percentage Page

When the student logs in through his id and when he clicks on the percentage button it will display his academic percentage on the display.

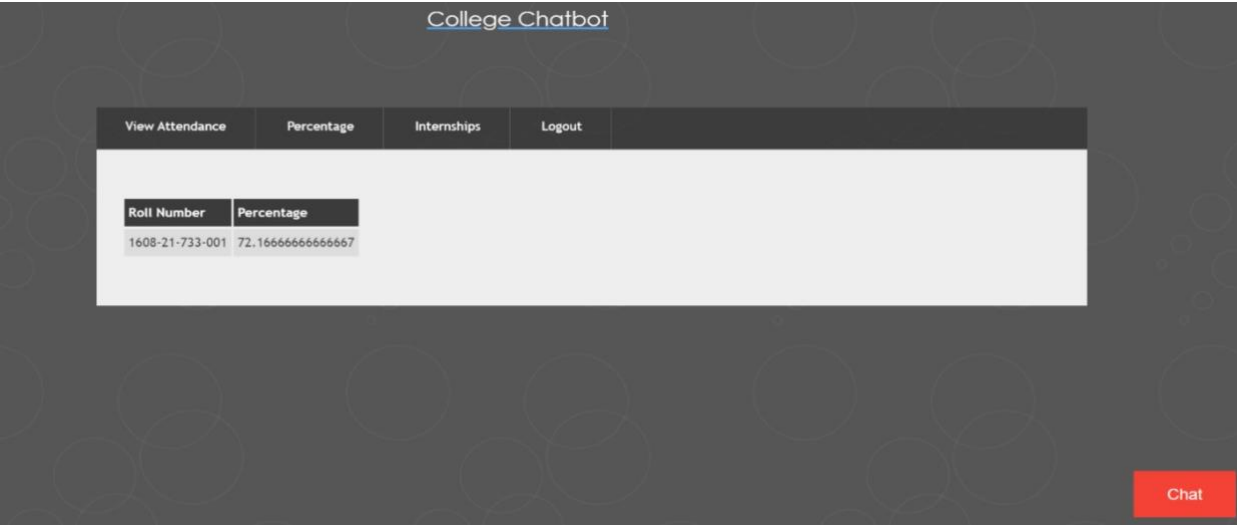


Fig 7.12 Student Percentage Page

7.3.2 Internship Page

When the student logs in through his id and when he clicks on the internships button it will display his internships status on the display.

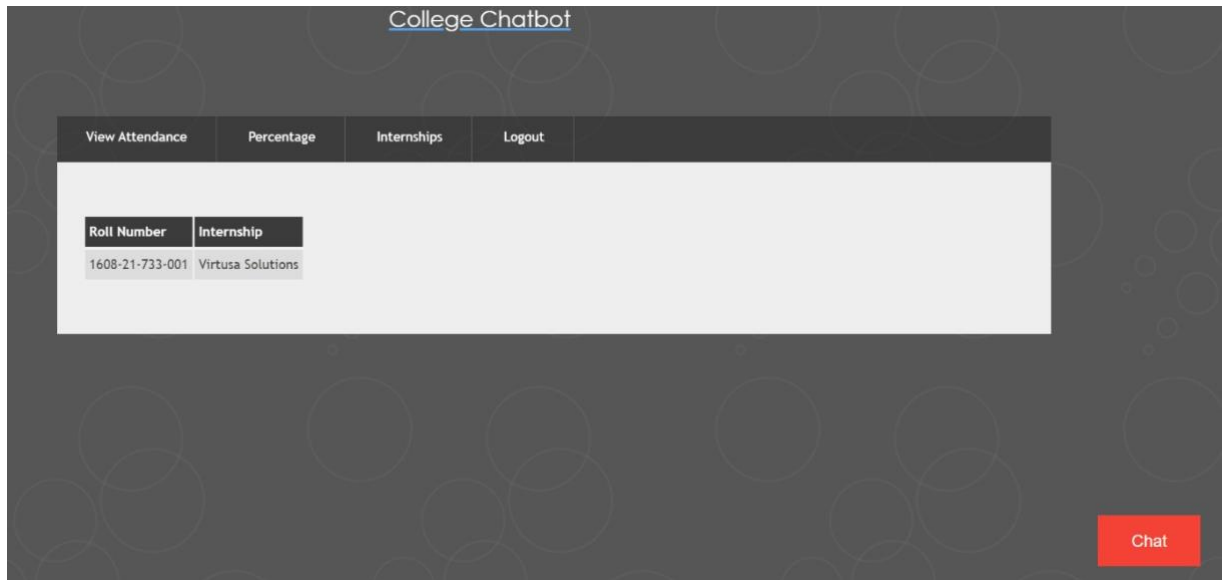


Fig 7.13 Student Internship Page

7.4 FACULTY PAGE

The feature is tested and is working fine. It is successful in validating and generating strong passwords based on the password characteristics given by the user. Below screenshot demonstrates the working of the feature.

7.4.1 Attendance Actions

The Faculty Page is tested and working fine. Faculty can add and view attendance of students.

The screenshot shows the 'Add Attendance' page of the College CRM. At the top, there is a navigation bar with links: Add Attendance, Upload Marks, Percentage, Student Percentage, Add Internship, and Internships. Below this is a secondary bar with View Attendance and Logout. The main content area features a form with five dropdown menus labeled Department, Year, Semester, Section, and Regulation, each with a '--select--' option. A Submit button is positioned below these fields. At the bottom left, there is a table header with 'Roll Number' and 'Percentage' columns. A red Chat button is located in the bottom right corner.

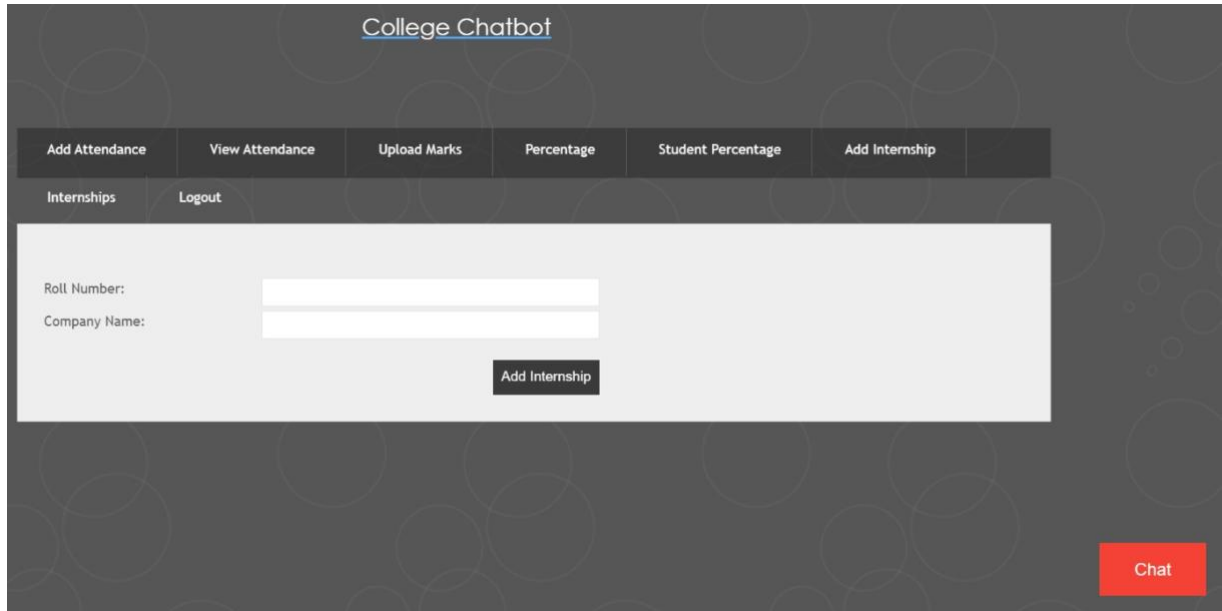
Fig 7.14 Add Attendance Page

The screenshot shows the 'View Attendance' page of the College CRM. The layout is identical to the 'Add Attendance' page, featuring the same navigation bar, secondary bar, and form fields. The table header at the bottom left also shows 'Roll Number' and 'Percentage' columns. A red Chat button is present in the bottom right corner.

Fig 7.15 View Attendance Page

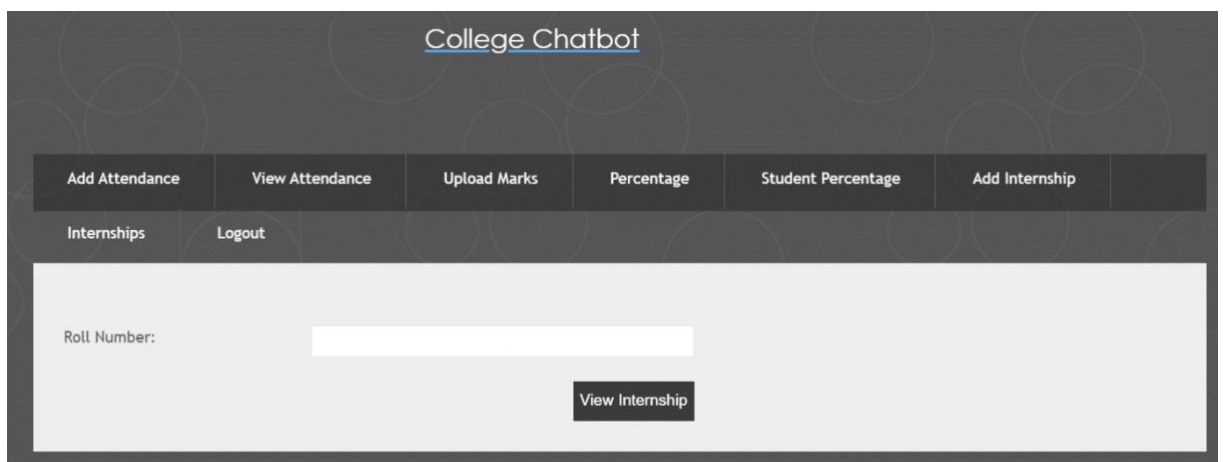
7.4.2 Internship Actions

Faculty can add the internship programs of students and also can view their internships.



The screenshot shows the 'College Chatbot' interface. At the top, the title 'College Chatbot' is displayed. Below it is a navigation bar with buttons: 'Add Attendance', 'View Attendance', 'Upload Marks', 'Percentage', 'Student Percentage', and 'Add Internship'. The 'Add Internship' button is highlighted. Below the navigation bar, there are two tabs: 'Internships' and 'Logout'. The 'Internships' tab is active. The main content area is a light gray box containing two input fields: 'Roll Number:' and 'Company Name:'. Below these fields is a button labeled 'Add Internship'. In the bottom right corner of the interface, there is a red button labeled 'Chat'.

Fig 7.16 Add internship action

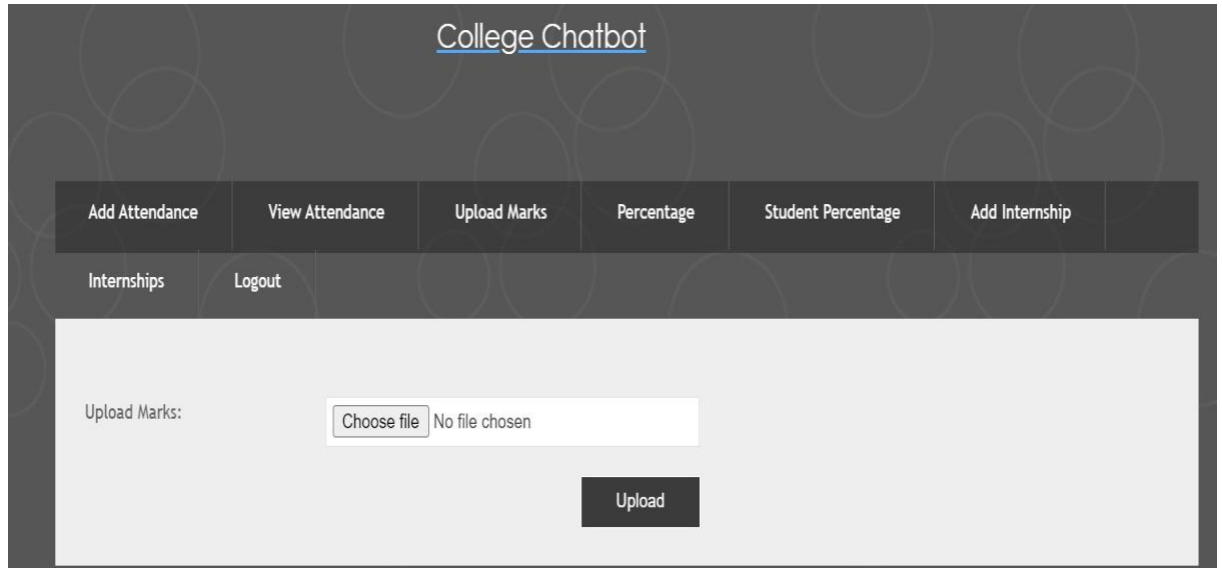


The screenshot shows the 'College Chatbot' interface. At the top, the title 'College Chatbot' is displayed. Below it is a navigation bar with buttons: 'Add Attendance', 'View Attendance', 'Upload Marks', 'Percentage', 'Student Percentage', and 'Add Internship'. The 'Add Internship' button is highlighted. Below the navigation bar, there are two tabs: 'Internships' and 'Logout'. The 'Internships' tab is active. The main content area is a light gray box containing one input field: 'Roll Number:'. Below this field is a button labeled 'View Internship'.

Fig 7.17 View Internship

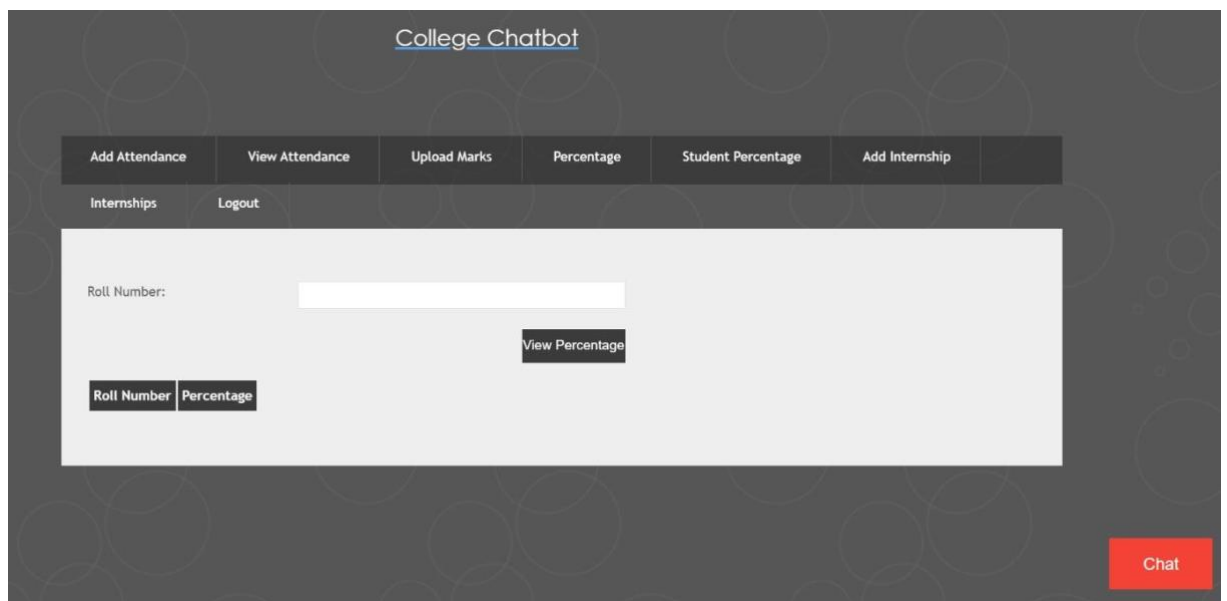
7.4.3 Percentage Page

Faculty can add the marks of students and View their Percentage.



The screenshot shows the 'College Chatbot' interface with a navigation bar containing buttons for 'Add Attendance', 'View Attendance', 'Upload Marks', 'Percentage', 'Student Percentage', and 'Add Internship'. Below the navigation bar, there are buttons for 'Internships' and 'Logout'. The main content area is titled 'Upload Marks:' and features a file upload section with a 'Choose file' button and the text 'No file chosen'. An 'Upload' button is located below the file upload section.

Fig 7.18 Upload Marks



The screenshot shows the 'College Chatbot' interface with a navigation bar containing buttons for 'Add Attendance', 'View Attendance', 'Upload Marks', 'Percentage', 'Student Percentage', and 'Add Internship'. Below the navigation bar, there are buttons for 'Internships' and 'Logout'. The main content area is titled 'Roll Number:' and features a text input field. A 'View Percentage' button is located below the input field. At the bottom left, there is a table with two columns: 'Roll Number' and 'Percentage'. At the bottom right, there is a red 'Chat' button.

Fig 7.19 View Percentage

7.5 Admin Page

Admin Page is tested and working fine. Admin can add the details of Students and faculty and also can view them

The screenshot shows the 'College Chatbot' admin interface. At the top, there is a navigation bar with links: 'Add Faculty' (highlighted in yellow), 'View Faculty', 'Add Student', 'View Students', and 'Logout'. Below the navigation bar is a registration form titled 'Registration Status :'. The form contains the following fields: 'Name :', 'User Name :', 'Password :', 'Email :', 'Mobile :', and 'Department:'. The 'Department:' field is a dropdown menu with '--select--' as the selected option. A 'Register' button is located at the bottom right of the form. In the bottom right corner of the interface, there is a red 'Chat' button.

Fig 7.20 Add Faculty Details

The screenshot shows the 'College Chatbot' admin interface. At the top, there is a navigation bar with links: 'Add Faculty', 'View Faculty', 'Add Student' (highlighted in yellow), 'View Students', and 'Logout'. Below the navigation bar is a registration form titled 'Registration Status :'. The form contains the following fields: 'Name :', 'Roll Number :', 'Password :', 'Email :', 'Mobile :', 'Department:', 'Year:', 'Section:', and 'Regulation:'. The 'Department:', 'Year:', 'Section:', and 'Regulation:' fields are dropdown menus with '--select--' as the selected option. A 'Register' button is located at the bottom right of the form. In the bottom right corner of the interface, there is a red 'Chat' button.

Fig 7.21 Add Student Details

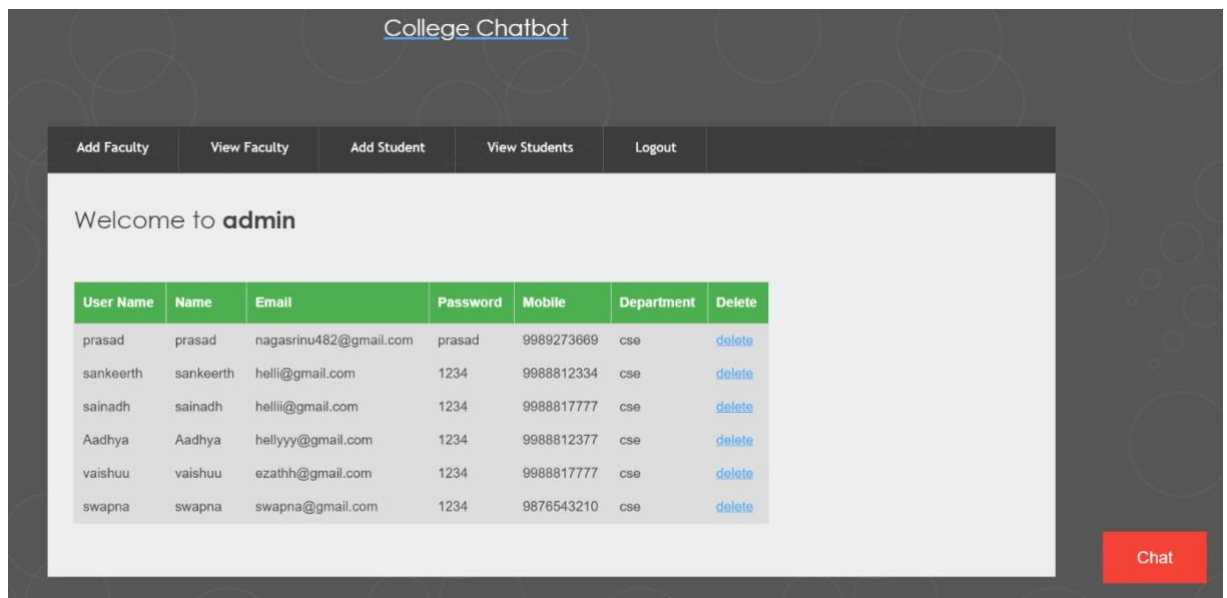


Fig 7.22 View Faculty

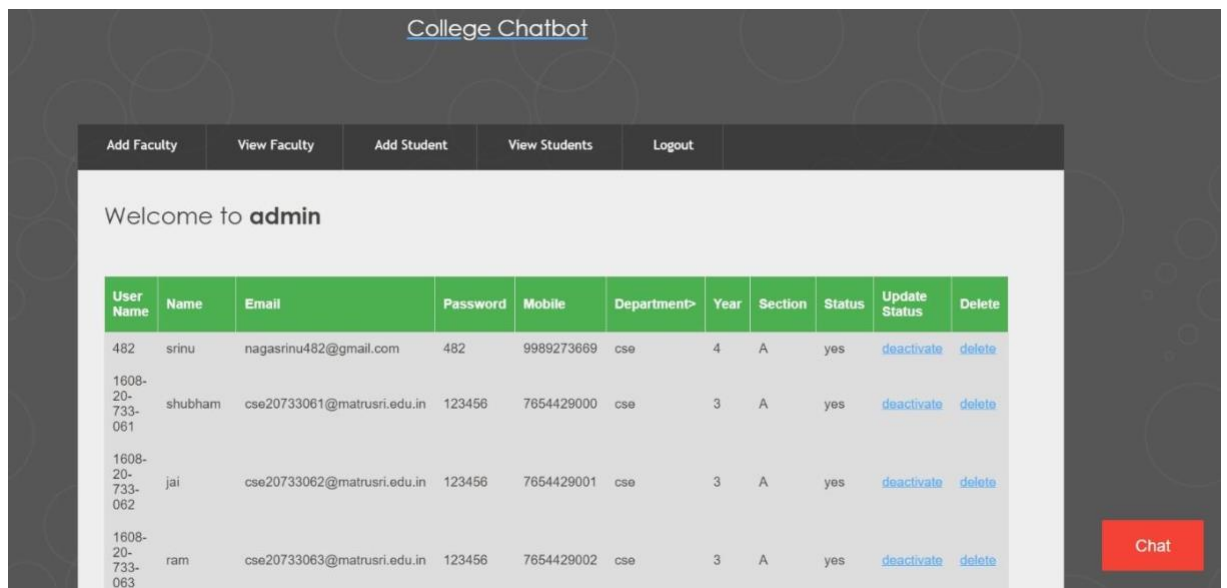
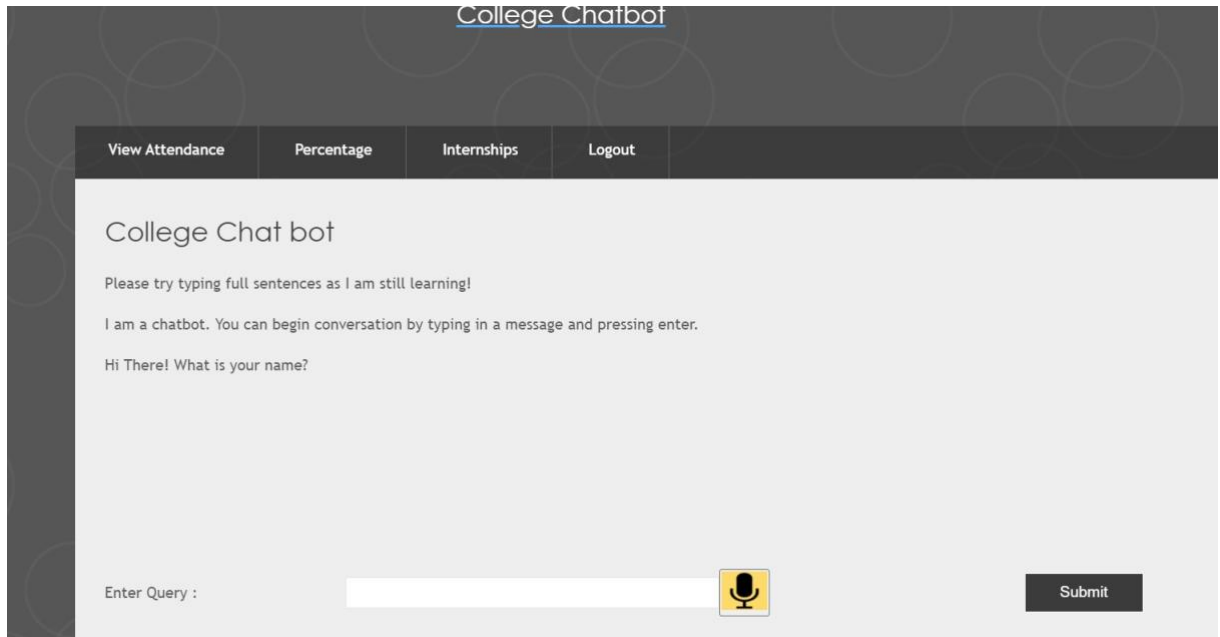


Fig 7.23 View student

7.6 Chatbot Feature

Chatbot Feature is tested and working fine. Chatbot Feature is used in all modules but the differ between Faculty, Admin and student is where the student can access his data only but the faculty and admin which deals with all the students of data.

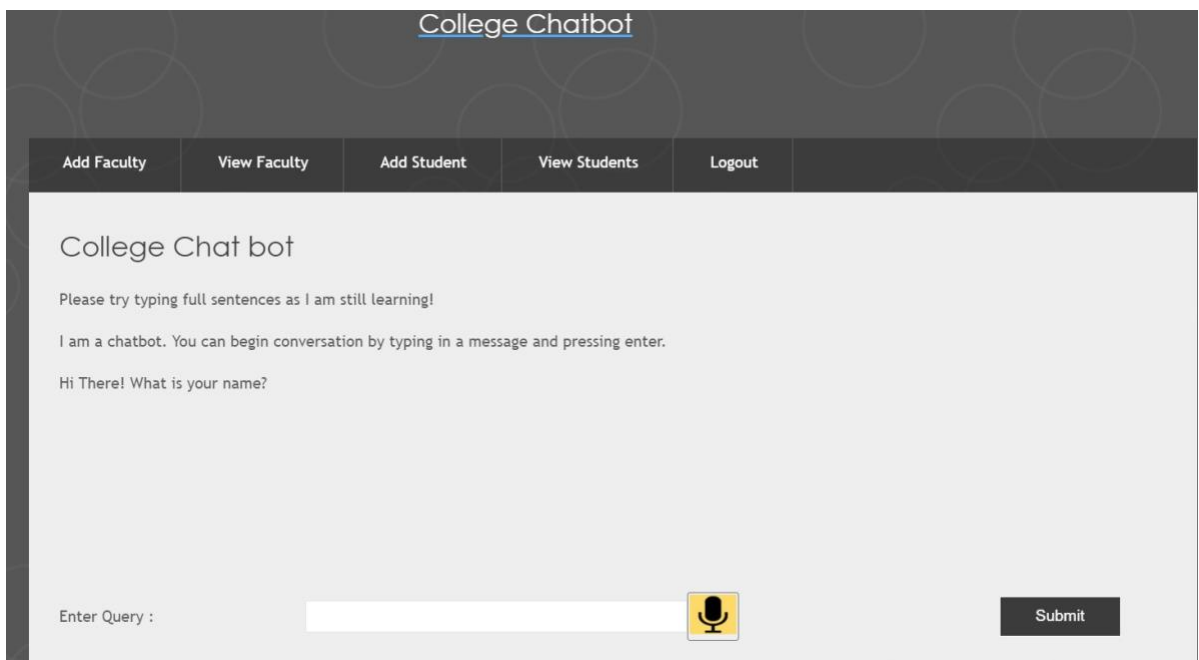
7.6.1 Student Chatbot Feature



The screenshot shows the 'College Chatbot' interface for a student. At the top, there's a header with the title 'College Chatbot' and a navigation bar with links: 'View Attendance', 'Percentage', 'Internships', and 'Logout'. The main chat area contains the following text: 'College Chat bot', 'Please try typing full sentences as I am still learning!', 'I am a chatbot. You can begin conversation by typing in a message and pressing enter.', and 'Hi There! What is your name?'. At the bottom, there is an input field labeled 'Enter Query :', a microphone icon, and a 'Submit' button.

Fig 7.24 student Chabot

7.6.2 Admin Chatbot



The screenshot shows the 'College Chatbot' interface for an admin. At the top, there's a header with the title 'College Chatbot' and a navigation bar with links: 'Add Faculty', 'View Faculty', 'Add Student', 'View Students', and 'Logout'. The main chat area contains the following text: 'College Chat bot', 'Please try typing full sentences as I am still learning!', 'I am a chatbot. You can begin conversation by typing in a message and pressing enter.', and 'Hi There! What is your name?'. At the bottom, there is an input field labeled 'Enter Query :', a microphone icon, and a 'Submit' button.

Fig 7.25 Admin chatbot

College Chat bot


Please try typing full sentences as I am still learning!

I am a chatbot. You can begin conversation by typing in a message and pressing enter.

Hi There! What is your name?

Subject_Code	Subject Name	Internal_Marks	External_Marks	Total	Credits	Actual Marks
1	CD	18	56	74	4	100
2	DAA	12	57	69	4	100
3	ML	13	65	78	4	100
4	CNS	14	66	80	4	100
5	ST	15	45	60	4	100
6	SSIS	16	54	70	4	100

Enter Query :



Submit

Fig 7.26 Marks in Chatbot

Add FacultyView FacultyAdd StudentView StudentsLogout

College Chat bot


Please try typing full sentences as I am still learning!

I am a chatbot. You can begin conversation by typing in a message and pressing enter.

Hi There! What is your name?

Roll Number	Percentage
1608-20-733-061	71.83333333333334

Enter Query :



Submit

Fig 7.27 Academic percentage in Chatbot

Add Faculty	View Faculty	Add Student	View Students	Logout		
-------------	--------------	-------------	---------------	--------	--	--

College Chat bot


Please try typing full sentences as I am still learning!

I am a chatbot. You can begin conversation by typing in a message and pressing enter.

Hi There! What is your name?

Roll Number	Internship
1608-20-733-061	Virtusa Solutions

Enter Query :



Submit

Fig 7.28 Internship in chatbot

8. CONCLUSION

In conclusion, the implementation of a student monitoring system using a chatbot offers a transformative solution to enhance the educational experience and support the holistic development of students. This project leverages the power of technology to provide real-time assistance, engagement, and personalized support to students, ultimately contributing to their academic success and overall well-being. By effectively tracking academic progress, addressing queries, and offering timely interventions, the chatbot fosters a proactive learning environment that empowers students to take ownership of their education.

Furthermore, the student monitoring chatbot has the potential to alleviate the workload of educators by automating routine administrative tasks, allowing them to focus more on providing quality instruction and mentorship. Additionally, the data collected and analyzed by the chatbot can offer valuable insights into student learning patterns, enabling institutions to make data-driven decisions to improve their educational offerings.

However, it's important to recognize that while the chatbot can be a valuable tool, it should not replace the human element in education. It should be viewed as a supportive tool that complements the efforts of teachers and administrators, rather than a substitute for meaningful interactions. Moreover, ethical considerations such as data privacy and security must be rigorously addressed to ensure that student information is safeguarded.

In essence, the student monitoring chatbot project represents a harmonious fusion of technological innovation and educational advancement. By harnessing the capabilities of AI-driven chatbots, educational institutions can create an enriched and responsive learning environment that nurtures students' growth, fosters academic excellence, and prepares them for success in an increasingly digital world.

9. LIMITATIONS OF THE PROJECT

This application is limited only for Web-based. It cannot be installed on OS. Chatbots can have limitations in understanding natural language and may not always accurately interpret the context or intent of the student's queries or responses. This could lead to incorrect or inadequate information being provided. Implementing a student monitoring system involves collecting and storing student data. Ensuring the security and privacy of this data is of utmost importance, and any data breaches or mishandling of information can lead to serious consequences.

Some students may have difficulty expressing themselves through text-based interactions, and a chatbot might struggle to understand and cater to their needs effectively. The machine learning model used to detect the malicious URL needs heavy computational power as the dataset contains large number of entries. It is important to ensure that the model is properly deployed and is scalable and efficient.

A chatbot-based system might not be able to fully replace human interaction and support from teachers, counselors, or other school staff. Certain situations may require face-to-face communication and human intervention, which the chatbot cannot provide. Keeping updated with the real-time location of the user to its contacts is not possible.

10. SCOPE OF THE PROJECT

The scope of a Student Monitoring System using a chatbot can be broad and dynamic, depending on the specific objectives and requirements of the project. However, here are some key components that can be included in the scope:

Chatbot Functionality

Develop a chatbot capable of natural language processing (NLP) to understand and respond to students queries, feedback, and requests. Implement a user-friendly interface to interact with the chatbot.

Academic Performance Tracking

Provide students with quick access to their grades and academic progress. Allow the chatbot to generate reports on students' academic performance.

Attendance Monitoring

Allow students to inquire about their attendance records and provide responses on their attendance status.

Internship Checker

Allows to check whether students are done any internships or not.

FUTURE ENHANCEMENTS

The future scope of a student monitoring system using a chatbot project is promising and can lead to various improvements and innovations. As technology continues to advance and AI capabilities expand, there are several potential areas of growth for such a system:

Enhanced Natural Language Processing (NLP): As NLP techniques improve, chatbots can better understand and respond to students' queries and concerns. Advanced NLP algorithms can help the chatbot comprehend complex questions, leading to more accurate and relevant responses.

Emotional Intelligence: Developing emotional intelligence in chatbots can make them more empathetic and understanding when students express frustration or stress. This can improve the overall emotional well-being of students and foster a positive learning environment.

Multilingual Support: Expanding the chatbot's language capabilities to support multiple languages can make education more accessible to students from diverse linguistic backgrounds.

Voice Interaction: Integrating voice-based chatbot interactions can enhance usability and accessibility, enabling students to engage with the system using voice commands.

Personalization: Future student monitoring chatbots can be designed to provide personalized feedback and recommendations to individual students based on their learning styles, strengths, weaknesses, and progress. This level of personalization can significantly improve the learning experience.

11.REFERENCES

- [1] A guide to Natural Language Processing, Available at https://en.wikipedia.org/wiki/Natural_language_processing
- [2] Chatbot definition, Available at [https://medium.com/@mg/bot-is-ahilariously-over-simplified-buzzword-let-s-fix-thatf1d63abb8ba7#:~:text=A%20chatterbot%20\(also%20known%20as,via%20auditory%20or%20textual%20methods](https://medium.com/@mg/bot-is-ahilariously-over-simplified-buzzword-let-s-fix-thatf1d63abb8ba7#:~:text=A%20chatterbot%20(also%20known%20as,via%20auditory%20or%20textual%20methods).
- [3] Introduction to Artificial Intelligence Markup Language, Available at https://www.tutorialspoint.com/aiml/aiml_introduction.html
- [4] Prof.K.Bala, Mukesh Kumar, SayaliHulawale, SahilPandita,“Chat-Bot For College Management System Using A.I” International Research Journal of Engineering and Technology (IRJET) Volume: 04, Issue: 11, Page no: 2030-2033| Nov 2017.Stemmer Algorithm, Available at <http://snowball.tartarus.org/algorithms/porter/stemmer.html>
- [5] Guruswami Hiremath, AishwaryaHajare, PriyankaBhosale, RasikaNanaware, Dr. K. S. Wagh, “Chatbot for education system” International Journal of Advance Research, Ideas and Innovations in Technology (IJARIIT) ISSN: 2454-132X, Volume: 4, Issue: 3, Page no: 37-43|2018.
- [6] Amey Tiwari, Rahul Talekar, Prof.S.M.Patil, “College Information Chat Bot System” International Journal of Engineering Research and General Science (IJERGS) Volume: 5, Issue: 2, Page no: 131-137| March-April 2017.
- [7] K. Jwala, G.N.V.G Sirisha, G.V. Padma Raju, “Developing a Chatbot using Machine Learning” International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume: 8 Issue: 1S3, Page no: 89-92| June 2019.
- [8] Basics of Natural Language ToolKit, Availilable at <https://www.nltk.org/>
- [9] Naeun Lee, Kirak Kim, Taeseon Yoon, “Implementation of Robot Journalism by Programming Custombot using Tokenization and Custom Tagging” International Conference on Advanced Communications Technology (ICACT) Page no: 566-570| Feb 2017.
- [10] Fundamentals of Natural Language Processing - Tokenization, Lemmatization, Stemming and Sentence Segmentation, Available at https://colab.research.google.com/github/dairai/notebooks/blob/master/_notebooks/2020-03-19-nlp_basics_tokenization_segmentation.ipynb#scrollTo=H7gQFbUxOQt b

[11] Jazzy spell checker Library, Available at <http://jazzy.sourceforge.net/>

[12] WordNet Algorithm, Available at <https://wordnet.princeton.edu/>

[13] Setiaji Bayu, Wibowo Ferry “Chatbot Using a Knowledge in Database: Human-to-Machine Conversation Modeling” 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS) Page no: 72-77| Jan 2016. DOI: 10.1109/ISMS.2016.53.

[14] Synsets for a word in WordNet, Available at <https://www.geeksforgeeks.org/nlp-synsets-for-a-word-inwordnet/#:~:text=WordNet%20is%20the%20lexical%20database,that%20express%20the%20same%20concept.>