A Mini Project Report

On

SMART COLLEGE CHATBOT

Submitted in partial fulfillment of the Requirements for the award of the degree of

Bachelor Of Technology

In

Computer Science and Engineering

By

D.Bhanu	RO200666
G.Anjali	RO200514
CH.Pradhyum	RO200905
CH.Sravanthi	RO201019

Under the Supervision of

Mr. N. Mallikarjuna

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE AND TECHNOLOGIES

ONGOLE CAMPUS

2024-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE AND TECHNOLOGIES

ONGOLE CAMPUS - AP

CERTIFICATE

This is to certify that the project entitled "**SMART COLLEGE CHATBOT** " submitted by D.Bhanu, G.Anjali, CH.Pradhyum, CH.Sravanthi in partial fulfilment of the requirements for the Mini Project Reviews and for award of the degree of the Bachelor of Technology in Computer Science and Engineering to Rajiv Gandhi University of Knowledge Technologies is a record of bonafide work carried out by them under my guidance and supervision from January 2025 to April 2025.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this project report have not been submitted to any other university for the award of any other degree.

Mr. N. Mallikarjuna
Assistant Professor
Head of CSE Department
RGUKT – Ongole

APPROVAL SHEET

This report entitled **Smart College Chatbot** submitted by D.Bhanu, G.Anjali, CH.Pradhyum, CH.Sravanthi, has been approved by **Mr.N.Mallikarjuna**, Assistant Professor and Head of CSE Department, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

Examiner(s)

Supervisor(s)

Date :

Place ;

DECLARATION

We hereby declare that the project work entitled "**Smart College Chatbot**" submitted to **Rajiv Gandhi University Of Knowledge Technologies AP-Ongole** Campus in partial fulfilment of the requirements for the award of the degree in Computer Science and Engineering is a record of an original work done by us under the guidance of Mr. N. Mallikarjuna, Assistant Professor and Head of Computer Science and Engineering Department and this project work have not been submitted to any other university for the award of any other degree.

Signatures of students:

D.Bhanu (RO200666)

G.Anjali (RO200514)

CH.Pradhyum (RO200905)

CH.Sravanthi (RO201019)

Date :

Place ;

ACKNOWLEDGEMENT

It is our privilege to express a profound sense of respect, gratitude and indebtedness to our guide **Mr. N. Mallikarjuna,** Assistant Professor and Head of Computer Science Engineering Department, RGUKT - Ongole, for his indefatigable inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout the dissertation work.

We extend our sincere thanks to Dean **Mr. M Rupas Kumar**, Research and development, RGUKT - Ongole, for his encouragement and constant help.
We extend our sincere thanks to **Dr. Bhaskar Patel**, Director, RGUKT Ongole for his encouragement.

D.Bhanu	RO200666
G.Anjali	RO200514
CH.Pradhyum	RO200905
CH.Sravanthi	RO201019

ABSTRACT

A chatbot, usually referred to as a chatterbot, attempts to have a conversation with a person. When a question is posed, the system has the ability to detect sentences and select the proper answer. The response principle is the matching of the user's input phrase. The current technical project involves building a professional system for a college help desk employing an android-based chatbot, artificial intelligence technology, and virtual assistance (human-machine communication), then sending that natural language to a server. Chatbot systems have become increasingly popular for automating interactions with users and providing information in various domains, including college enquiries. In this paper, we propose a chatbot system for college enquiry using a knowledgeable database. The system utilizes a knowledgeable database that contains relevant information about the college, such as courses, faculty, campus facilities, and admissions procedures. The system employs various algorithms, including rule-based, retrieval-based, natural language processing (NLP), and machine learning algorithms, to understand and respond to user queries in a context-aware manner. The rule-based algorithms provide predefined rules and patterns for handling specific intents or frequently asked questions, while the retrieval-based algorithms search the knowledgeable database for relevant information.

CONTENTS

1. INTRODUCTION

This Application is for college students, staff, and parents. Easy way to interaction.This System is a web application which provides answer to the query of the student. Students just must 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 analyses the question and then 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.

1.1.Motivation

The increasing demand for quick and reliable access to information within educational institutions has highlighted the need for smarter solutions. Students frequently have questions related to admissions, academic schedules, fees, and campus facilities. Relying on notice boards, websites, or waiting for staff responses often leads to delays and confusion. In many cases, administrative staff are overwhelmed with repetitive queries, which could be easily answered through automation. With the rise of digital communication, students expect faster and more interactive solutions. A chatbot system can offer 24/7 assistance, providing instant responses to commonly asked questions. It not only reduces the burden on staff but also improves the overall efficiency of communication within the college. Our motivation for building this Smart College Chatbot System is to enhance student experience by offering a quick, user-friendly, and intelligent support system that bridges the communication gap between students and college services.

1.2.Contribution

The Smart College Chatbot System contributes to the digital transformation of educational support services by providing an intelligent, automated platform for student interaction. This project simplifies the process of obtaining information by offering a responsive chatbot that delivers accurate answers to frequently asked questions. It reduces the dependency on administrative staff for routine queries and ensures students have access to help at any time. Our chatbot is designed with a user-friendly interface and a categorized FAQ system, making it easy for users to navigate and find information quickly. The system also lays the groundwork for future enhancements, such as voice support, multilingual responses, and integration with internal college databases for real-time updates. By introducing this solution, we aim to improve communication, enhance user satisfaction, and contribute to a smarter, more efficient campus environment.

1.3. Problem Statement

The problem addressed by this project is the inefficiency and lack of responsiveness in traditional methods of communication within colleges. Students often rely on notice boards, websites, or waiting in queues to get answers to common queries related to admissions, academics, fees, and campus facilities. These methods are time-consuming, inconvenient, and often fail to provide real-time assistance. Additionally, administrative staff are frequently burdened with repetitive questions, reducing their productivity. To solve this issue, the project proposes a Smart College Chatbot System that uses AI to provide instant and accurate responses to student queries. By offering a user-friendly and accessible platform, the chatbot enhances communication, improves efficiency, and ensures students receive the support they need anytime, anywhere. This solution aims to modernize student support systems and set a new standard for digital interaction in educational institutions.

1.4 Existing System

The existing system in most educational institutions relies on traditional methods such as notice boards, official websites, emails, or in-person communication for delivering information to students. Students are often required to search through multiple web pages or physically visit departments to get answers to common queries related to admissions, course details, exam schedules, fee payments, and campus facilities. While these methods are functional, they are not always efficient or user-friendly. The communication process is slow, prone to human error, and lacks real-time responsiveness. Students may experience delays in receiving important updates or may miss information altogether. Additionally, these systems place a significant burden on administrative staff who must repeatedly address the same questions. The current setup lacks automation, personalization, and adaptability, and it does not leverage modern technologies such as artificial intelligence or machine learning to improve user experience or operational efficiency.

1.5 Drawback of Existing System

Overall, the existing system provides only basic support for student queries and information access but falls short in delivering fast, efficient, and interactive communication. It lacks real-time response capabilities, leading to delays and student frustration, especially during peak periods like admissions or exams. The system does not provide personalized assistance, forcing students to search through static websites or third-party sources, which may contain outdated or incorrect information. Additionally, the reliance on human staff for repetitive queries increases workload and reduces operational efficiency. These limitations highlight the growing need for smarter, more accessible systems that align with modern student expectations. Integrating AI-based chatbot technology presents an opportunity to improve student engagement, provide 24/7 support, and create a more dynamic and responsive college communication platform.

1.6 Proposed System

The proposed system aims to modernize student support services by introducing a Machine Learning and Natural Language Processing (NLP)-based Smart College Chatbot. This intelligent system is designed to understand and respond to student queries in a conversational manner, offering a more interactive and accessible communication platform. By leveraging NLP, the chatbot can process and interpret natural language input, allowing users to ask questions freely without needing specific formats. Machine learning enables the system to improve over time by learning from past interactions and adapting to user needs. The chatbot will provide instant answers to frequently asked questions related to admissions, academics, exams, fees, and campus facilities, reducing the dependency on human staff. This approach ensures 24/7 availability, quick information delivery, and a personalized user experience, setting a new standard for smart communication in educational institutions.

Key Features of the Proposed System

- Natural Language Processing (NLP): Enables the chatbot to understand and respond to user queries in natural, conversational language.
- Machine Learning (ML) Integration: Continuously improves response accuracy by learning from previous interactions and user feedback.
- Categorized Query Handling: Organizes FAQs into categories like Admissions, Academics, Fees, Exams, etc., for faster and easier navigation.
- 24/7 Availability: Provides round-the-clock support to students, ensuring access to information anytime without staff dependency.
- Instant and Accurate Responses: Delivers immediate answers to common queries, reducing wait times and improving user satisfaction.
- User-Friendly Interface: Simple and intuitive design that allows students to interact with the system through buttons or direct text input.
- Scalable and Customizable: Can be easily expanded with new features, categories, or languages to meet future requirements.
- Reduces Administrative Burden: Automates repetitive tasks, allowing staff to focus on more important academic and administrative responsibilities.

- Multilingual Support (Optional): Can be extended to support multiple languages, enhancing accessibility for diverse student populations.
- Secure Data Handling: Ensures student information and query data are processed with privacy and security in mind.

Advantages

- **Centralized Communication Platform:** The chatbot acts as a one-stop solution for students to access all academic and campus-related information, reducing confusion caused by scattered resources.
- **Quick On boarding for New Students:** New admissions often come with lots of questions — the chatbot serves as a virtual guide, helping them adapt to the college environment smoothly.
- **Smart Query Routing** : With additional development, the system can forward complex queries to the appropriate department or staff member, improving resolution efficiency.
- **Low Maintenance Requirements**: After initial training and setup, the chatbot can run with minimal manual intervention, making it easy to maintain.
- Increased Transparency: By making all student-related information instantly available, the chatbot helps ensure transparency between students and administration.
- **Encourages Digital Transformation**: Adoption of such intelligent systems promotes a culture of innovation and smart campus infrastructure.
- **Feedback Collection**: The chatbot can be extended to collect student feedback on services, events, or faculty, helping institutions improve based on real-time insights.
- **Eco-Friendly Solution**: Reduces the need for printed notices, brochures, and flyers by digitizing common information queries, supporting a more sustainable campus.
- **Cost-Effective Solution**: Once developed and deployed, the chatbot reduces the need for continuous human support, making it a cost-efficient option for colleges and institutions.

1.7 Applications of proposed system

The technology behind the Smart College Chatbot System — primarily Machine Learning (ML) and Natural Language Processing (NLP) — is highly flexible and can be adapted to various industries beyond education. These intelligent systems improve user experience by automating information delivery, offering 24/7 assistance, and providing accurate, personalized responses. As a result, similar chatbot-based systems are increasingly being used across different sectors to enhance customer engagement, reduce operational costs, and streamline communication.
Here are some major domains where this system can be applied:

- **Healthcare Sector:** Chatbots can help patients book appointments, provide symptom-based advice, send medication reminders, and answer health-related queries without human intervention.

- **E-Commerce and Online Retail:** These systems assist customers by recommending products, tracking orders, answering FAQs, managing returns, and handling complaints instantly.

- **Banking and Financial Services:** Chatbots can provide real-time assistance with balance inquiries, transaction history, loan details, and investment options, improving customer convenience and support.

- **Travel and Hospitality:** Virtual assistants can help users book tickets, check flight or hotel availability, provide itinerary updates, and give local travel tips — all through conversational interfaces.

- **Public Sector and Government Services:** Citizens can get real-time help with tax filing, accessing welfare schemes, knowing application deadlines, and locating public services through government-integrated chatbots.

2. LITERATURE SURVEY

2.1 Rise of Intelligent Assistants in the Academic World

The integration of AI into educational platforms has brought forward intelligent virtual assistants capable of interacting with students in natural, human-like ways. Using Machine Learning (ML) and Natural Language Processing (NLP), chatbots can now understand questions, provide instant responses, and reduce the dependency on manual support systems. These advancements have encouraged many institutions to adopt chatbot-based services to answer common student queries, helping improve communication, accessibility, and user satisfaction.

2.2 Limitations of Existing Solutions and Scope for Improvement

Despite their growth, many existing educational chatbots rely on predefined rule-based systems, which limits their ability to adapt, personalize, and learn from users. As a result, students often receive generic or inaccurate responses, leading to frustration and inefficiency. The proposed Smart College Chatbot System aims to bridge this gap by utilizing ML and NLP to create a dynamic, self-improving system built in Python. Unlike traditional bots, this chatbot learns from interaction, responds with context, and evolves over time, offering a more intelligent and student-focused experience.

3. REQUIREMENT SPECIFICATION

3.1 System Requirement Analysis

The direct result of requirements analysis is Requirements specification. Hardware requirements specifications list the necessary hardware for the proper functioning of the project. Software requirements specifications is a description of a software system to be developed, laying out functional and non-functional requirements, and may include a set of use cases that describe interactions the users will have the software. In software engineering, a functional requirement defines the function of a system and its components. A function is described as a set of inputs, the behavior, and outputs. A non-functional requirement that specifies the criteria that can be used to judge the operation of a system, rather than specific behaviour.

3.2 Functional Requirements

A function of software system is defined in functional requirement and the behaviour of the system is evaluated when presented with specific inputs or conditions which may include calculations, data manipulation and processing and other specific functionality.

The functional requirements of the project are one of the most important aspects in terms of entire mechanism of modules. After validating our model, it should be able to predict the future stock market price.

3.3. Non - Functional Requirements

Nonfunctional requirements describe how a system must behave and establish constraints of its functionality. This type of requirements is also known as the system's 'quality attributes'. Attributes such as performance, security, usability, compatibility are not the feature of the system, they are a required characteristic.

.

Any attributes required by the customer are described by the specification. We must include only those requirements that are appropriate for our project.

Some Non-Functional Requirements are as follows:

- Reliability

The structure must be reliable and strong in giving the functionalities. The movements must be made unmistakable by the structure when a customer has revealed a couple of enhancements. The progressions made by the Programmer must be Project pioneer and in addition the Test designer.

- Maintainability

The structure must be reliable and strong in giving the functionalities. The movements must be made unmistakable by the structure when a customer has revealed a couple of enhancements. The progressions made by the Programmer must be Project pioneer and in addition the Test designer.

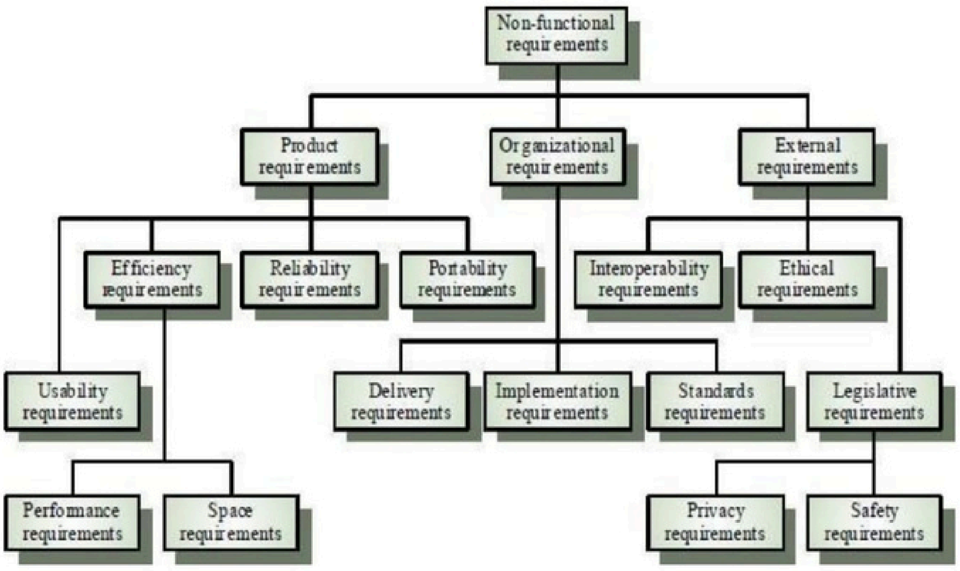


Fig 3.1 Non-functional requirements

- Performance

The framework will be utilized by numerous representatives all the while. Since the system will be encouraged on a single web server with a lone database server outside of anyone's ability to see, execution transforms into a significant concern. The structure should not capitulate when various customers would use everything the while. It should allow brisk accessibility to each and every piece of its customers. For instance, if two test specialists are all the while attempting to report the vicinity of a bug, then there ought not to be any irregularity at the same time.

- Portability

The framework should to be effectively versatile to another framework. This is obliged when the web server, which s facilitating the framework gets adhered because of a few issues, which requires the framework to be taken to another framework.

- Scalability

The framework should be sufficiently adaptable to include new functionalities at a later stage.
There should be a run of the mill channel, which can oblige the new functionalities.

- Flexibility

Flexibility is the capacity of a framework to adjust to changing situations and circumstances, and to adapt to changes to business approaches and rules. An adaptable framework is one that is anything but difficult to reconfigure or adjust because of diverse client and framework prerequisites. The deliberate division of concerns between the trough and motor parts helps adaptability as just a little bit of the framework is influenced when strategies or principles change.

3.4 Tools and Technology Details

Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, a hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL list tested, compatible, and sometimes in compatible hardware devices for a particular operating systems or applications. The CPU is a fundamental system requirement for any software most software running on different kinds of architecture defines processing power as the model and he clock speed of the CPU. In this memory requirements are defined after considering demands of applications, operating system, supporting software and files, and other running process. Hardware requirements specifications list the necessary hardware for the proper functioning of the project.

System Processor : Windows 7 or 7+

Hard Disk : 40 GB.

Ram : 512 MB.

Any desktop / Laptop system with above configuration or higher level.

Software Requirements

Software requirements deal with software resource requirements and prerequisites that need to be installed on the computer to provide optimal functioning of an application. These requirements are prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed. Software requirements specifications is a description of a software system to be developed, laying out functional and non-functional requirements, and may include a set of use cases that describe interactions the users will have the software.

Software's : Python 3.6 or high version
IDLE : PyCharm.
Framework : Flask

4. SYSTEM ANALYSIS

4.1. System Design

The Smart College Chatbot System is designed to interact with users in real-time and provide instant responses to their queries. It uses Machine Learning and Natural Language Processing to understand and process natural language input from students. The system is structured with a user-friendly interface, a backend processing unit, and a knowledge base to fetch accurate answers. It aims to simplify student-college communication by automating responses to frequently asked questions and continuously learning from interactions.

4.2. Algorithms used

4.2.1.NLTK (Natural Language Toolkit)

NLTK is a Python library used for processing and analyzing human language data. It provides tools for text tokenization, stemming, classification, and more. In the Smart College Chatbot System, NLTK helps understand student queries by breaking down text and matching it with relevant answers. Its simplicity and flexibility make it ideal for building efficient, NLP-based chatbot applications.

4.2.2. Knowledge Base Search Algorithm

The Knowledge Base Search Algorithm is used to retrieve accurate and relevant information from a structured repository containing college-related data, such as courses, departments, faculty details, admission procedures, and deadlines. When a user enters a query, the algorithm searches the knowledge base to find matching or related content. It uses techniques like keyword matching, semantic search, or natural language understanding (NLU) to interpret queries and rank results. More advanced implementations may involve vector-based search, knowledge graphs, or tools like Elasticsearch to improve speed and accuracy.

4.2.3. Response Generation Algorithm

The Response Generation Algorithm is responsible for creating appropriate and meaningful replies based on the user's input and detected intent. It can use rule-based approaches (predefined templates and keyword triggers) or AI-driven models like Transformers to generate dynamic responses. In simple chatbots, responses are selected from a fixed set based on matched intents, while advanced systems use natural language generation (NLG) to produce human-like answers. The goal is to ensure the chatbot responds in a way that is relevant, clear, and contextually appropriate.

5. SYSTEM DESIGN

5.1 UML diagrams :

UML is the short form of Unified Modelling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The important goal for UML is to create a common modelling language for the sake of Object Oriented Software engineering. In its current form UML consists of two major components: a Meta- model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization Constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

PURPOSE OF UML DIAGRAMS
• Document: Provides detailed documentation that can be referred to throughout the software development lifecycle.
• Specify: Defines the system's components, interactions, and behavior clearly and unambiguously.
• Construct: Assists in the design and development of the system by defining how components should interact and function.
• Communicate: Provides a common language for various stakeholders (team members, clients, etc.) to discuss and improve system design.
• Visualize: Helps stakeholders (developers, analysts, clients) to understand the system's architecture, processes, and interactions.

GOALS

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language
5. Encourage the growth of the OO tools market.
6. Support higher level development concepts such as collaborations frameworks, patterns and components and Integrate best practices. There are two main categories of UML diagrams

Structural Diagrams – Focus on the static structure of the system (e.g., how the system is built).

Behavioral Diagrams – Focus on the dynamic behavior of the system (e.g., how the system behaves over time).

STRUCTURAL UML DIAGRAMS : Structural diagrams depict a static view of astructure of a system. It is widely used in the Documentation of software architecture. It involves 7 diagrams They are:

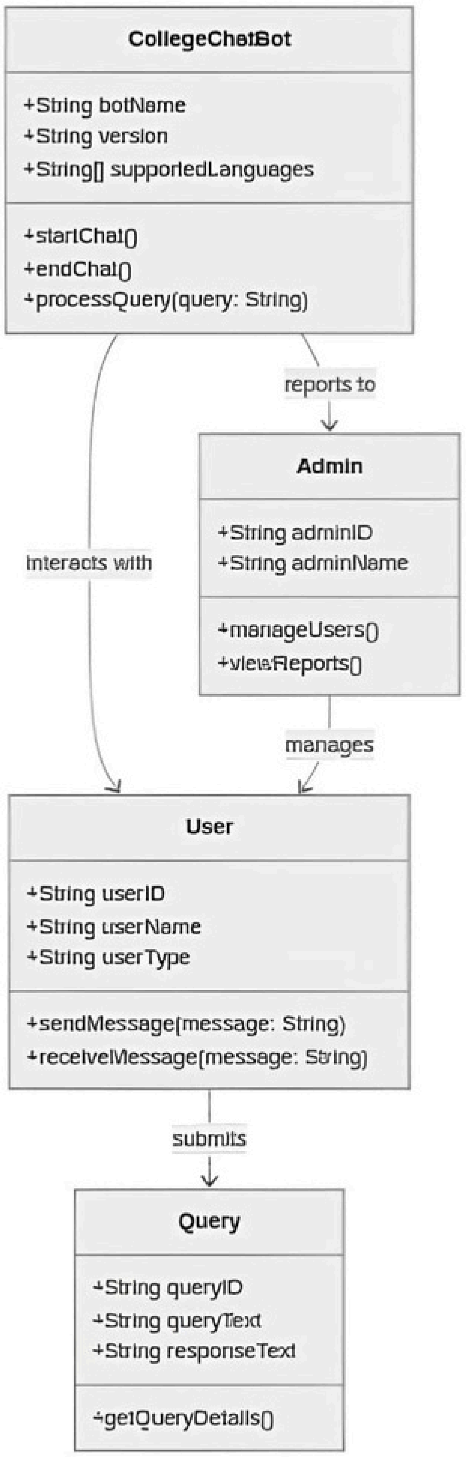
- Class Diagram
- Object Diagram
- Component Diagram
- Composite Structure Diagram
- Deployment Diagram
- Package Diagram
- Profile Diagram

BEHAVIOURAL UML DIAGRAMS: Behavioral diagrams portray a dynamic view of a system or the behaviour of a system, which describes the functioning the system. It involves 7 diagrams They are:

- Use case Diagram
- Sequence Diagram
- Activity Diagram
- State Machine Diagram
- Interaction Overview Diagram
- Communication Diagram
- Timing Diagram

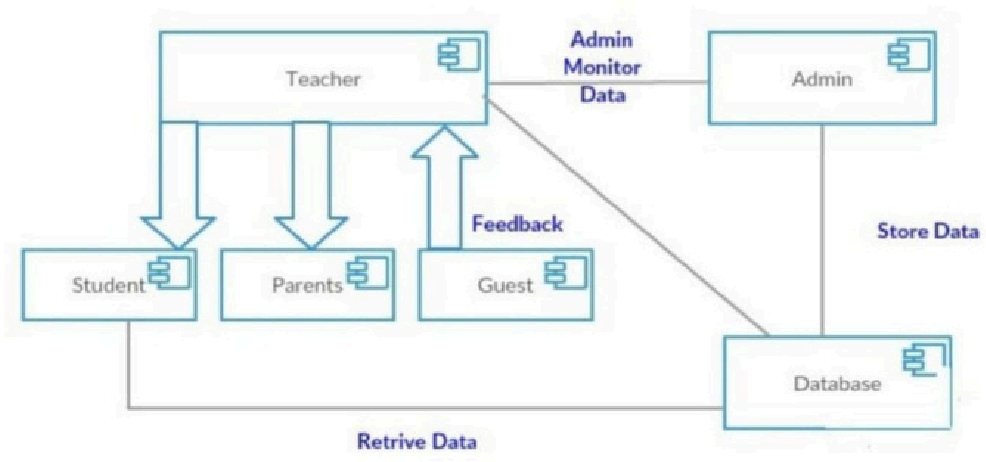
CLASS DIAGRAM

A class diagram is a type of static structure diagram used in UML (Unified Modeling Language) to describe the structure of a system. It shows the system's classes, their attributes (properties), methods (operations), and the relationships between the classes.



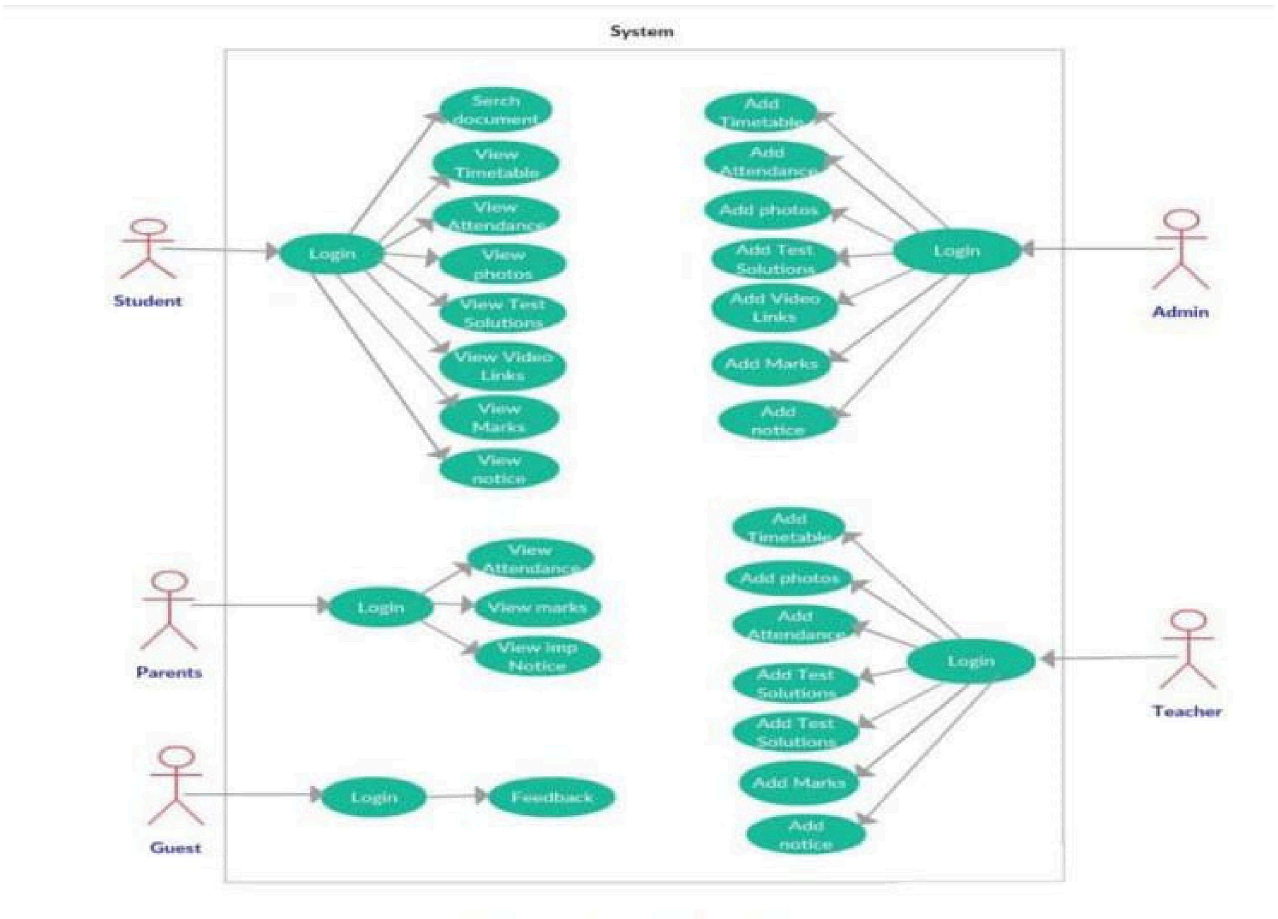
COMPONENT DIAGRAM

A Component Diagram is a type of UML (Unified Modeling Language) diagram used to model the physical components in a system and their interactions. These components represent modular parts of a system that can be independently developed, replaced, and deployed. In software design, a component typically represents a part of a system that encapsulates a set of related classes, packages, or subsystems.



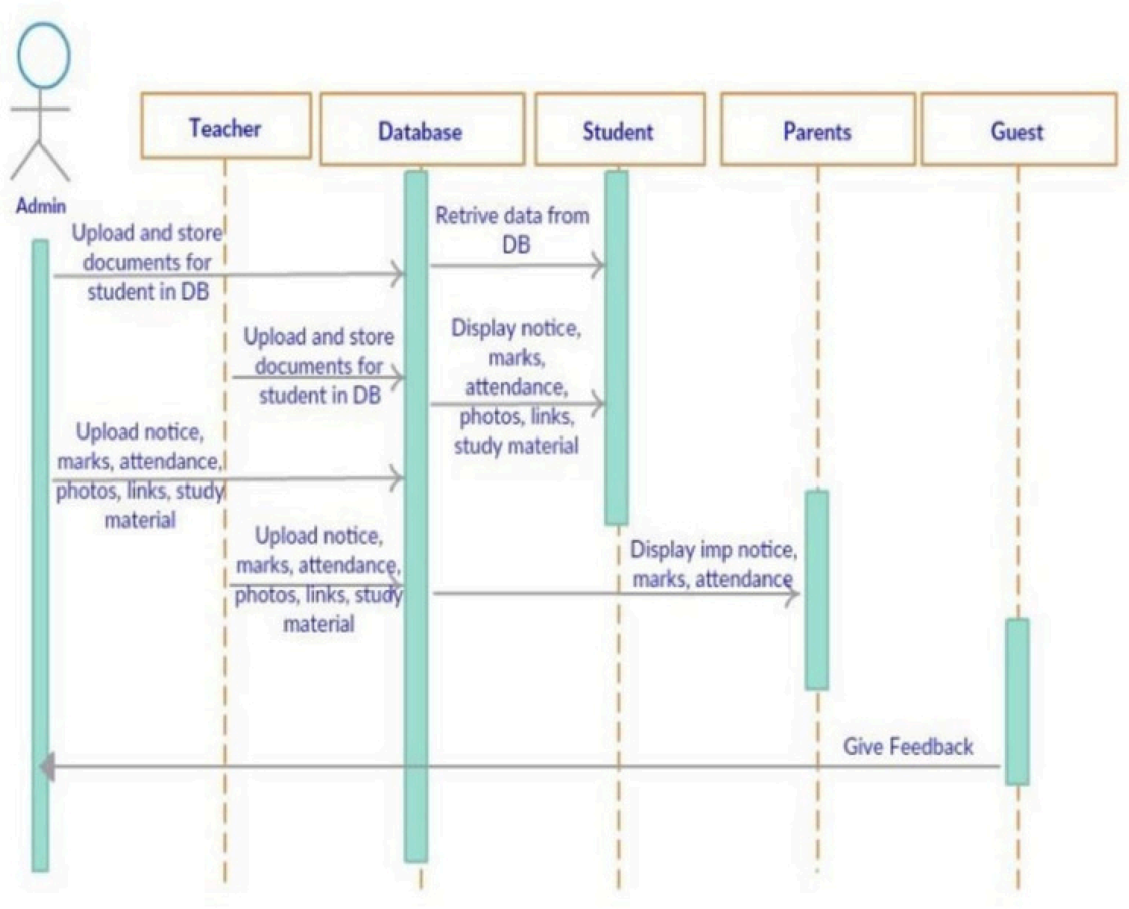
USE CASE DIAGRAM

A Use Case Diagram is a type of UML (Unified Modeling Language) diagram that visually represents the functional requirements of a system. It shows the interactions between users (or other systems) and the system itself in terms of use cases, which describe specific functions or behaviors that the system performs.



SEQUENCE DIAGRAM

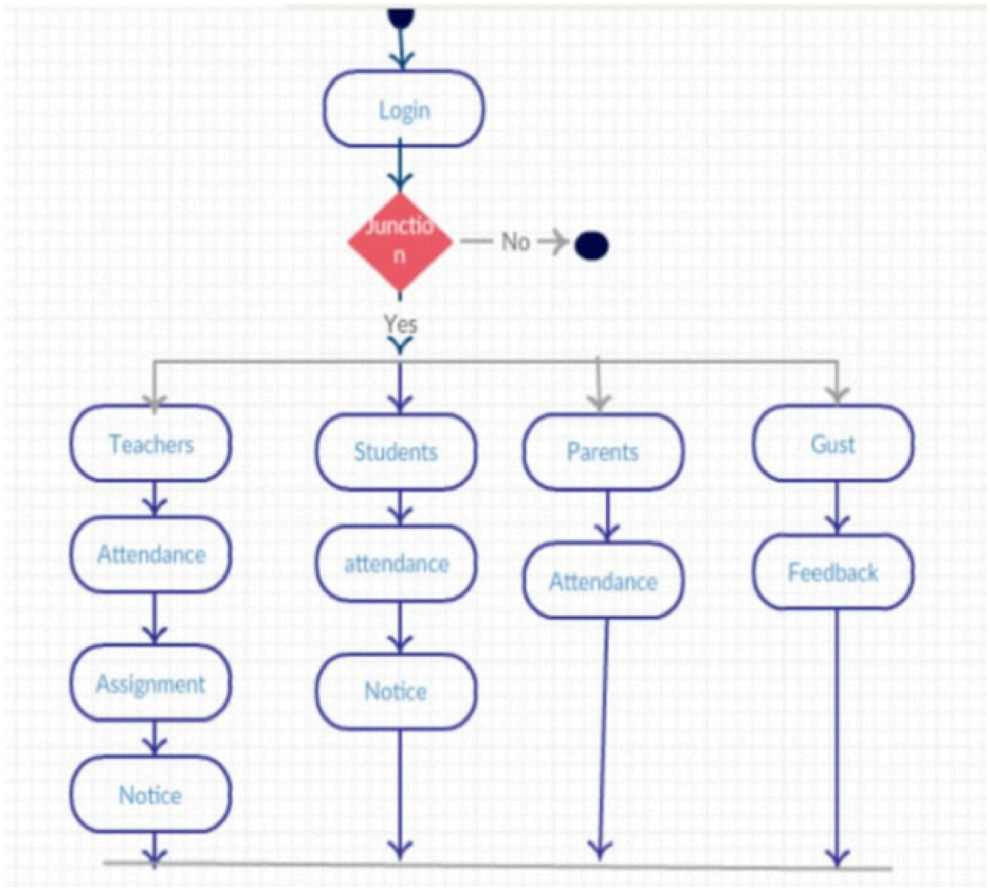
A sequence diagram is a unified Modeling Language (UML) diagram that illustrates the sequences of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the message that they exchange over time during the interaction.



Admin
Teacher
Database
Student
Parents
Guest
Upload and store documents for student in DB
Retrive data from DB
Upload and store documents for student in DB
Display notice, marks, attendance, photos, links, study material
Upload notice, marks, attendance, photos, links, study material
Upload notice, marks, attendance, photos, links, study material
Display imp notice, marks, attendance
Give Feedback

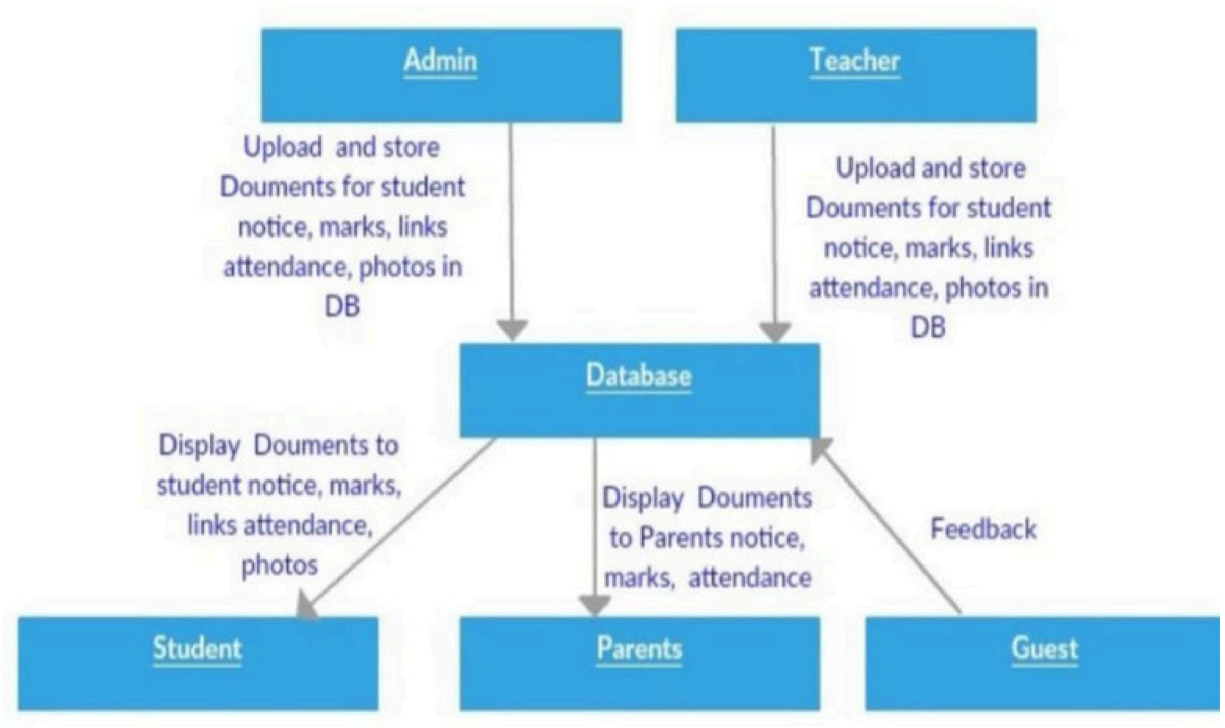
STATE MACHINE DIAGRAM

A State Machine Diagram (also known as a Statechart Diagram) is a type of UML (Unified Modeling Language) diagram that models the dynamic behavior of an individual object in response to various events. The diagram describes the states an object can be in, the transitions between those states, and the events that cause the transitions. State Machine Diagrams are particularly useful for modeling systems where objects go through a series of states based on internal or external events.



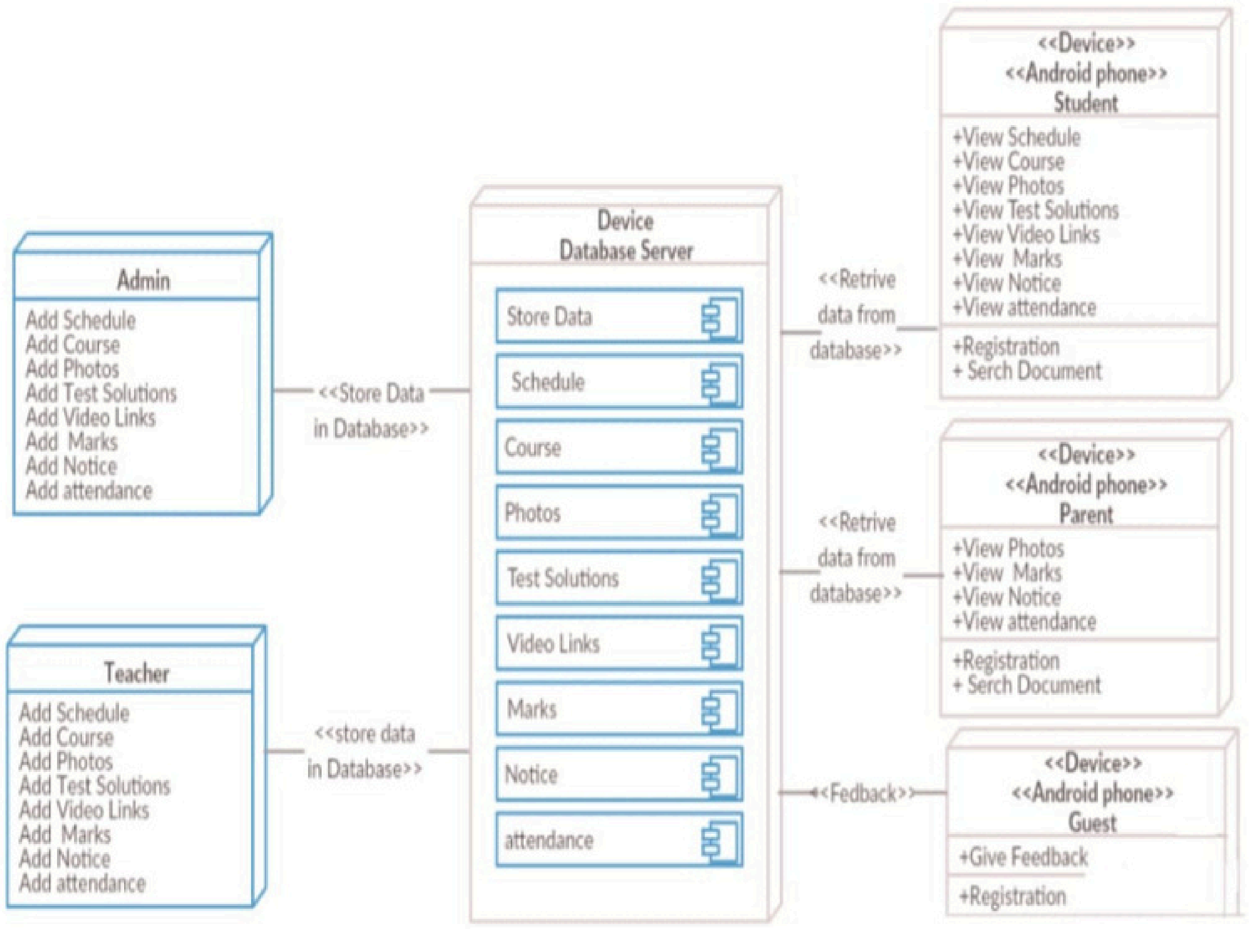
COMMUNICATION DIAGRAM

A Communication Diagram, also known as a Collaboration Diagram, is an interaction diagram in UML that shows how objects interact and communicate with each other through messages. It focuses on the structural organization of objects and their links. The diagram illustrates the sequence and flow of messages exchanged between objects to carry out a function. It's useful for visualizing how different parts of a system collaborate to achieve specific behavior.


Admin
Teacher
Upload and store Douments for student notice, marks, links attendance, photos in DB
Upload and store Douments for student notice, marks, links attendance, photos in DB
Database
Display Douments to student notice, marks, links attendance, photos
Display Douments to Parents notice, marks, attendance
Feedback
Student
Parents
Guest

DEPLOYMENT DIAGRAM

A deployment diagram is a type of UML (Unified Modeling Language) diagram that represents the physical deployment of components and their relationships in a system. In the context of the chatbot system for college enquiry using a knowledgeable database, a deployment diagram can be used to represent the physical deployment of the various components of the system. The deployment diagram for the chatbot system can include nodes such as User Interface Node, Natural Language Processing Engine Node, Knowledgeable Database Node, and Feedback Mechanism Node. Each node can represent a physical machine or a logical grouping of machines that host the corresponding components.



6.IMPLEMENTATION

6.1. Development and Deployment Setup

Certainly! A college enquiry chatbot can be built using a combination of LSTM (Long Short-Term Memory) and CNN (Convolutional Neural Network) models to process natural language inputs and generate appropriate responses.

Here is how it can work:

- Data collection: The first step is to collect a large amount of relevant data, such as frequently asked questions, course information, admission requirements, campus facilities, etc. This data will be used to train the chatbot model. The relevant data is taken from Concordia university for the overview of the project.

- Pre-processing: The first step is to pre-process the text inputs to extractimportant features and remove any noise. This can involve steps such as tokenization, stemming, lemmatization, stop word removal, and spell correction.

Natural Language Processing is a subfield of data science that works with textual data.
When it comes to handling the Human language, textual data is one of the most unstructured types of data available.NLP is a technique that
operates behind the it, allowing for extensive text preparation prior to any output. Before using the data for analysis in any Machine Learning work, it's critical to analyse the data. To deal with NLP-based problems, a variety of libraries and algorithms are employed. For text cleaning, a regular expression(re) is the most often used library. The next libraries are NLTK (Natural language toolkit) and spacy, which are used to execute natural language tasks like eliminating stop words. Pre-processing data is a difficult task. Text pre-processing is done in order to prepare the text data for model creation. It is the initial stage of any NLP project.

The following are some of the pre-processing steps:
• Tokenization
• Removing Stop words
• Lemmatization
• Lower casing

6.1.1. Tokenization

The initial stage in text analysis is tokenization. It enables to determine the text's core components. Tokens are the fundamental units. Tokenization is beneficial since it divides a text into smaller chunks. Internally, spacey determines if a "." is a punctuation and separates it into tokens, or whether it is part of an abbreviation like as "B.A." and does not separate it. Based on the problem, we may utilize sentence tokenization or word tokenization.
a. **Sentence tokenization:** using the sent_tokenize () function, dividing a paragraph into a collection of sentences.
b. **Word tokenization:** using the word_tokenize () technique, dividing a statement into a list of words.

6.1.2. Removing Stop words

To eliminate noise from data, data cleaning is essential in NLP. Stop words are the most frequently repeated words in a text that give no useful information.
The NLTK library includes a list of terms that are considered stop words in English. [I, no, nor, me, mine, myself, some, such we, our, you'd, your, he, ours, ourselves, yours, yourself, yourselves, you, you're, you've, you'll, most, other] are only a few of them.
The NLTK library is a popular library for removing stop words, and it eliminates about 180 stop words. For certain difficulties, we can develop a customized set of stop words. Using the add technique, we can easily add any new word to a collection of terms. Removing stop words refers to the process of considered to be common uninformative.

6.1.3. Lemmatization

The process of reducing inflected forms of a word while verifying that the reduced form matches to the language is known as lemmatization. A lemma is a simplified version or base word. Lemmatization uses a pre-defined dictionary to saves word context and verify the word in the dictionary as it decreases. Organizes, organized, and organizing, for example, are all forms of organize. The lemma in this case is organize. The inflection of a word can be used to communicate grammatical categories such as tense (organized vs organize). Lemmatization is required since it aids in the reduction of a word's inflected forms into a particular element for analysis. It can also assist in text normalization and the avoidance of duplicate words with similar meanings.

6.1.4. Lower casing

When the text is in the same case, a computer can easily read the words since the machine treats lower and upper case differently. Words like Cat and cat, for example, are processed differently by machines. To prevent such issues, we must make the word in the same case, with lower case being the most preferable instance. In python lower () is a function that is mostly used to handle strings. The lower () function accepts no parameters. It converts each capital letter to lowercase to produce lowercased strings from the provided string. If the supplied string has no capital characters, it returns the exact string.

Intent Recognition: The next step is to identify the intent behind the user's input. For example, if the user asks "What are the admission requirements for Computer Science?", the intent can be recognized as "Admission Requirements". This can be done using techniques such as rule-based systems, machine learning algorithms like Naive Bayes, or neural network models like LSTM.

Entity Recognition: Once the intent is recognized, the chatbot needs to extract the relevant entities from the user's input. In the above example, the entities would be "Computer Science". This can be done using techniques such as Named Entity Recognition (NER) or Part-of-Speech (POS) tagging.

Dialogue Management: The chatbot needs to maintain a conversation flow with the user and respond appropriately to their inputs. This can be achieved using techniques such as rule-based systems, finite-state machines, or reinforcement learning algorithms.

Response Generation: Finally, the chatbot generates a response to the user's input based on the intent and entities identified in the previous steps. The response can be a pre-defined template or a dynamically generated sentence. The response can be generated using techniques such as rule-based systems, templates, or machine learning algorithms like sequence-to-sequence models or Generative Pre-trained Transformer (GPT) models.

6.2. Algorithms

6.2.2. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of deep learning algorithm commonly used in image recognition and computer vision applications. This stands for Convolution Neural Network where Image data is mapped to a target variable. They have proven to be successful in that they are now the techniques of choice for any form of prediction issue utilizing data as an input to the model. CNN is a multi-layered feed-forward neural network that is built by layering several hidden layers on top of one another in a certain sequence. These layers are frequently outlawed by several layers in CNN, while activation layers are usually enhanced by layers in the convolutional network. In the context of a college enquiry chatbot system, CNN can be useful in several ways:

Data Analysis: CNN can be used to analyze textual data related to college enquiries. For example, if a user asks a question about admission requirements for a particular program, the chatbot can use a CNN model to extract the most important keywords and concepts from the text and provide a relevant response based on that information.

Improved Accuracy: Using a CNN model can improve the accuracy of the chatbot's responses, as it can quickly and accurately analyze large amounts of data related to college enquiries and provide the most relevant responses to users.

Chatbot training: CNNs can be used as a part of the training process for chatbots. For example, CNNs can be used to analyze large datasets of user queries and responses to identify patterns and improve the chatbot's ability to understand and respond to user queries.

Text classification: CNNs can be used to classify user input into different categories or intents. This is useful in chatbots as it allows the chatbot to understand the user's query and respond appropriately. CNNs can learn to identify patterns in text data and can be trained on large datasets to improve their accuracy.

Entity extraction: CNNs can be used to extract relevant information from unstructured text data, such as course descriptions or faculty biographies. This can be useful in chatbots for providing detailed information to users.

Contextual understanding: CNNs can be used to improve the chatbot's contextual understanding of user input. Overall, CNN can be a valuable tool in a college enquiry chatbot system, as it can help to enhance the accuracy and effectiveness of the chatbot in responding to user enquiries, especially when it comes to analyzing visual and textual information. Contextual understanding refers to the ability to coin the context in which it is presented.

Your paImage Recognition: CNN can help the chatbot to identify images related to college enquiries. For example, if a user sends an image of a college campus, the chatbot can use a pre-trained CNN model to recognize the image and extract relevant information such as the name of the college, its location, and other details that can assist the user in their enquiry.

6.3. Module Implementation

6.3.1. RDFLIB

RDFLib is a pure Python package for working with RDF. RDFLib contains most things you need to work with RDF, including:

- parsers and serializers for RDF/XML, N3, NTriples, N-Quads, Turtle, TriX,
- Trig and JSON-LD
- a Graph interface which can be backed by any one of a number ofStore implementations
- store implementations for in-memory, persistent on disk (Berkeley DB) and remote SPARQL endpoints
- a SPARQL 1.1 implementation - supporting SPARQL 1.1 Queries and
- Update statements
- SPARQL function extension mechanisms

6.3.2. RE

A RegEx, also known as a Regular Expression, is a string of characters that defines a search patterns. This module's functions allow to see if a given string matches a given regular expression.

6.3.3. RANDOM

The Python Random module is a built-in module for generating random integers in Python. These are sort of fake random numbers which do not possess true randomness. We can therefore use this module to generate random numbers, display a random item for a list or string, and so on.

6.3.4. CSV

The CSV module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats.

6.4. SAMPLE CODE

```
from chatbot import chatbot
from flask import Flask, render_template,
request,session,logging,url_for,redirect,flash
from flask_recaptcha import ReCaptcha
import mysql.connector
import os

app = Flask(__name__)
recaptcha = ReCaptcha(app=app)
app.secret_key=os.urandom(24)
app.static_folder = 'static'


app.config.update(dict(
   RECAPTCHA_ENABLED = True,
   RECAPTCHA_SITE_KEY =
"6LdbAx0aAAAAANl04WHtDbraFMufACHccHbn09L",
   RECAPTCHA_SECRET_KEY =
"6LdbAx0aAAAAMmkgBKJ2Z9xsQjMD5YutoXC6Wee"
))

recaptcha=ReCaptcha()
recaptcha.init_app(app)

app.config['SECRET_KEY'] = 'cairocoders-ednalan'

#database connectivity
conn=mysql.connector.connect(host='localhost',port='3306',user='root',passwo
rd='#####@25',database='register',auth_plugin='mysql_native_password')
cur=conn.cursor()
```

```
# Google recaptcha - site key :
6LdbAx0aAAAAAANl04WHtDbraFMufACHccHbn09L
# Google recaptcha - secret key :
6LdbAx0aAAAAAMmkgBKJ2Z9xsQjMD5YutoXC6Wee

@app.route("/index")
def home():
   if 'id' in session:
      return render_template('index.html')
   else:
      return redirect('/')

@app.route('/')
def login():
   return render_template("login.html")

@app.route('/register')
def about():
   return render_template('register.html')

@app.route('/forgot')
def forgot():
   return render_template('forgot.html')
@app.route('/login_validation',methods=['POST'])
def login_validation():
   email=request.form.get('email')
   password=request.form.get('password')

   cur.execute("""SELECT * FROM `users` WHERE `email` LIKE '{}' AND
`password` LIKE '{}'""".format(email,password))
   users = cur.fetchall()
```

```
if len(users)>0:
        session['id']=users[0][0]
        flash('You were successfully logged in')
        return redirect('/index')
else:
 flash('Invalid credentials !!!')
 return redirect('/')
 # return "The Email is {} and the Password is {}".format(email,password)
 # return render_template('register.html')
@app.route('/add_user',methods=['POST'])
def add_user():
    name=request.form.get('name')
    email=request.form.get('uemail')
    password=request.form.get('upassword')

    #cur.execute("UPDATE users SET password='{}'WHERE name =
'{}'".format(password, name))
    cur.execute("""INSERT INTO  users(name,email,password)
VALUES('{}','{}','{}')""".format(name,email,password))
    conn.commit()
    cur.execute("""SELECT * FROM `users` WHERE `email` LIKE
'{}'""".format(email))
    myuser=cur.fetchall()
    flash('You have successfully registered!')
    session['id']=myuser[0][0]
    return redirect('/index')
@app.route('/suggestion',methods=['POST'])
def suggestion():
    email=request.form.get('uemail')
    suggesMess=request.form.get('message')

    cur.execute("""INSERT INTO  suggestion(email,message)
VALUES('{}','{}')""".format(email,suggesMess))
    conn.commit()
    flash('You suggestion is succesfully sent!')
    return redirect('/index')
```

```
cur.execute("""INSERT INTO suggestion(email,message)
VALUES('{}','{}')""".format(email,suggesMess))
 conn.commit()
 flash('You suggestion is succesfully sent!')
 return redirect('/index')
@app.route('/add_user',methods=['POST'])
def register():
   if recaptcha.verify():
      flash('New User Added Successfully')
      return redirect('/register')
   else:
      flash('Error Recaptcha')
      return redirect('/register')


@app.route('/logout')
def logout():
   session.pop('id')
   return redirect('/')

@app.route("/get")
def get_bot_response():
   userText = request.args.get('msg')
   return str(chatbot.get_response(userText))

if __name__ == "__main__":
   # app.secret_key=""
   app.run()
```

app.py

```
from chatterbot import ChatBot
from chatterbot.trainers import ListTrainer
import spacy
spacy.load('en_core_web_sm')
# from spacy.lang.en import English
from chatterbot.trainers import ChatterBotCorpusTrainer
```

```
# from spacy.lang.en import English
from chatterbot.trainers import ChatterBotCorpusTrainer
# Creating ChatBot Instance
chatbot = ChatBot('<b>CRCE BOT</b>')

# nlp = spacy.load("en_core_web_sm")

chatbot = ChatBot(
    'ChatBot for College Enquiry',
    storage_adapter='chatterbot.storage.SQLStorageAdapter',
    logic_adapters=[
        {
            'import_path': 'chatterbot.logic.BestMatch',
            'default_response': "Hi there, Welcome to Fr. CRCE! 👋 If you need any
assistance, I'm always here.Go ahead and write the number of any query. 😃
✨<b><br><br>  Which of the following user groups do you belong to? <br>
<br>1.&emsp;Student's Section Enquiry.</br>2.&emsp;Faculty Section
Enquiry. </br>3.&emsp;Parent's Section Enquiry.</br>4.&emsp;Visitor's
Section Enquiry.</br><br>",
            'maximum_similarity_threshold': 0.90
        }
    ],
    database_uri='sqlite:///database.sqlite3'
)
trainer = ListTrainer(chatbot)
# python app.py
# Training with Personal Ques & Ans
conversation = [
"Hi",
"Helloo!",
"Hey",
"How are you?",
"I'm good.</br> <br>Go ahead and write the number of any query. 😃✨
<br> 1.&emsp;Student's Section Enquiry.</br>2.&emsp;Faculty Section
Enquiry. </br>3.&emsp;Parent's Section Enquiry.</br>4.&emsp;Visitor's
Section Enquiry.</br>",

"Great",
"Go ahead and write the number of any query. 😃✨ <br> 1.&emsp;Student's
Section Enquiry.</br>2.&emsp;Faculty Section Enquiry.
</br>3.&emsp;Parent's Section Enquiry.</br>4.&emsp;Visitor's Section
Enquiry.</br>",

"good",
"Go ahead and write the number of any query. 😃✨ <br> 2.&emsp;Faculty
Section Enquiry. </br>3.&emsp;Parent's Section Enquiry.
</br>4.&emsp;Visitor's Section Enquiry.</br>",

"fine",
"Go ahead and write the number of any query. 😃✨ <br> 2.&emsp;Faculty
Section Enquiry. </br>3.&emsp;Parent's Section Enquiry.
</br>4.&emsp;Visitor's Section Enquiry.</br>",

"Thank You",
"Your Welcome 😄",

"Thanks",
"Your Welcome 😄",
 ]

trainer.train(conversation)
```

```
import mysql.connector

try:
    conn = mysql.connector.connect(
        host="localhost",
        port="3306",
        user="root",
        password="#####@25",
        database="register",
        auth_plugin="mysql_native_password"
    )
    print("✅ Connected successfully!")
    conn.close()
except mysql.connector.Error as err:
    print(f"❌ Error: {err}")
```

6.5. ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures the system meets the functional requirements.

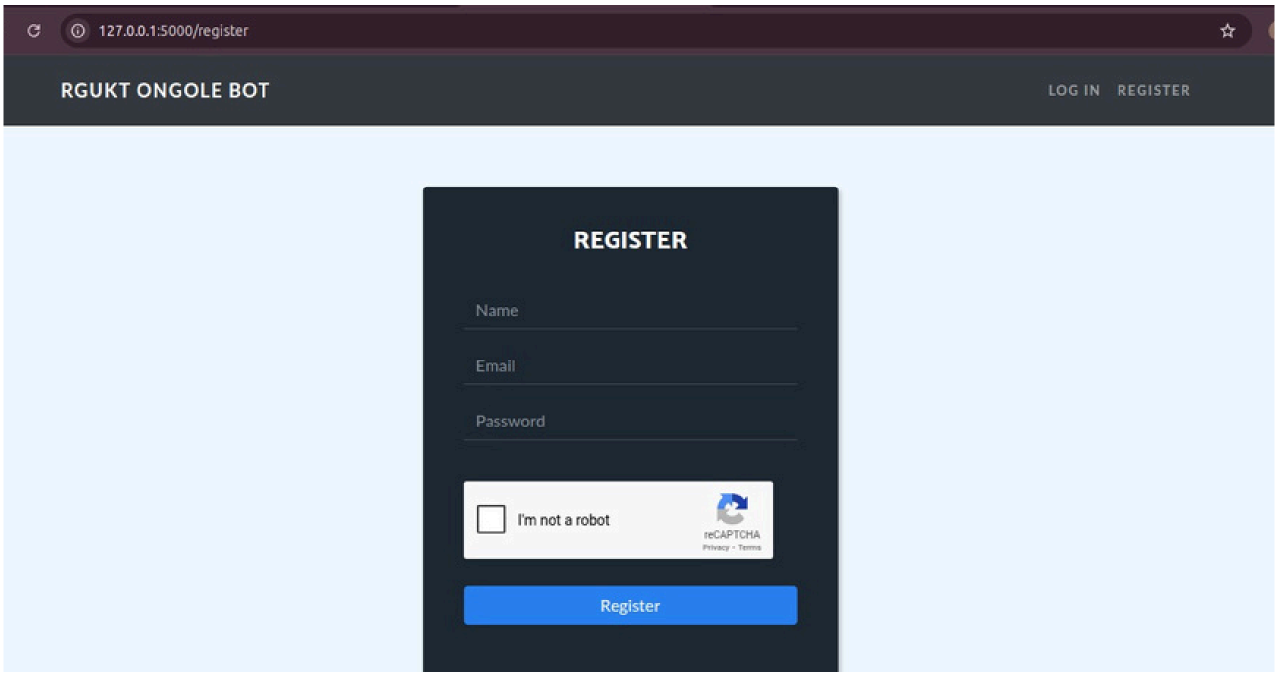
Test Case No. 1
Name of Test: Category Selection and FAQ Display
Test Case Description:
User selects a category, and relevant FAQs should appear.

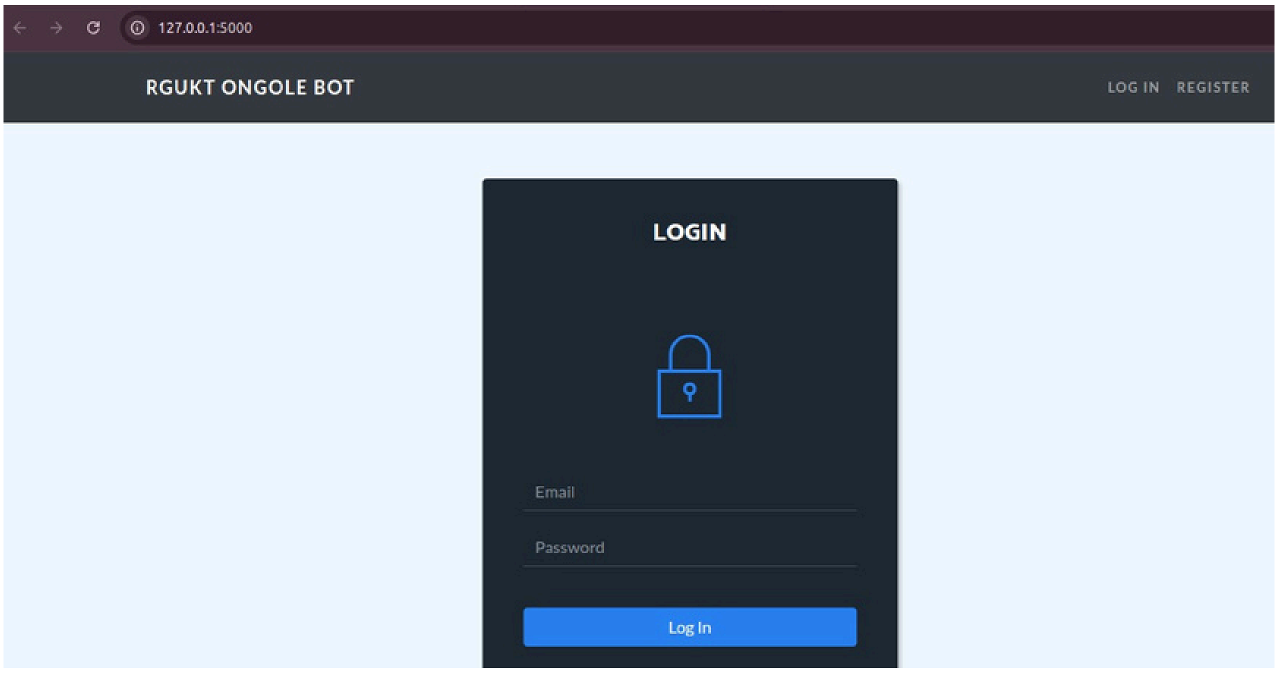
- Sample Input: User clicks on the “Admission” category
- Expected Output: FAQs related to admission are displayed
- Actual Output: Correct admission FAQs shown
- Status: Pass

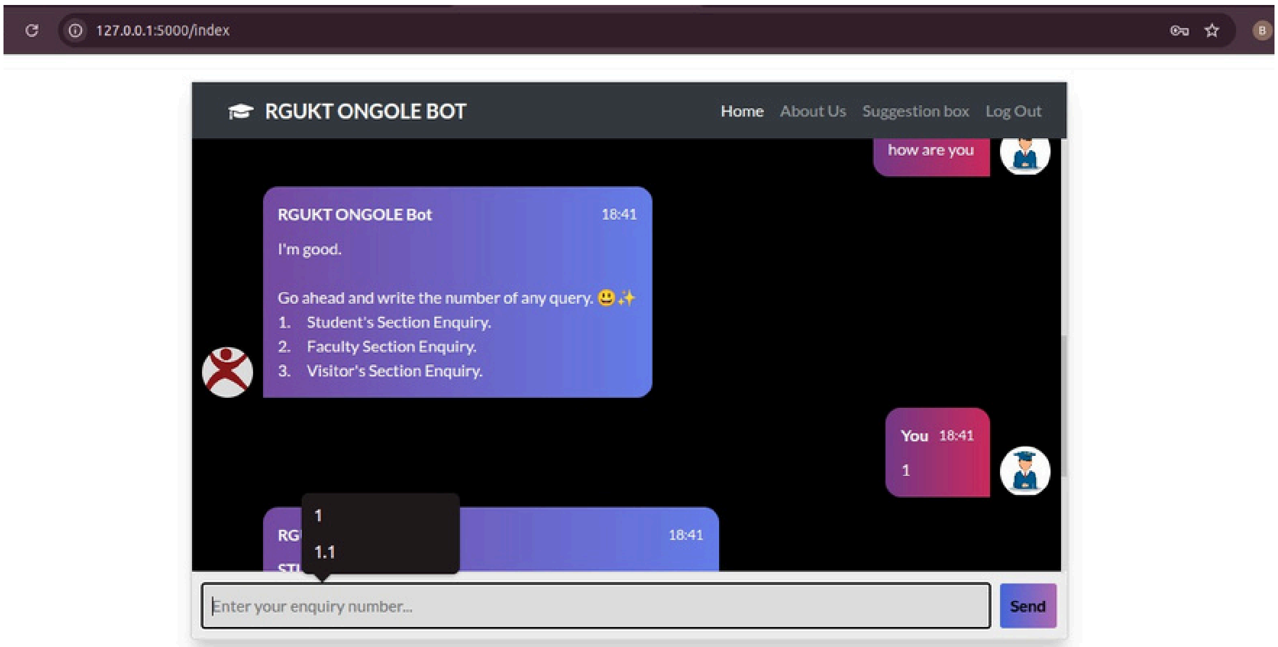
Test Case No. 2
Name of Test: Relevant Answer Display
Test Case Description:
User selects a question and should receive an appropriate answer.

- Sample Input: "What is the admission process?"
- Expected Output: Chatbot replies with the admission process details
- Actual Output: Relevant answer displayed correctly
- Status: Pass

6.6. SCREENSHOTS






127.0.0.1:5000/index

RGUKT ONGOLE BOT

Home About Us Suggestion box Log Out

how are you

RGUKT ONGOLE Bot 18:41

I'm good.

Go ahead and write the number of any query. 😃✨
1. Student's Section Enquiry.
2. Faculty Section Enquiry.
3. Visitor's Section Enquiry.

You 18:41

1

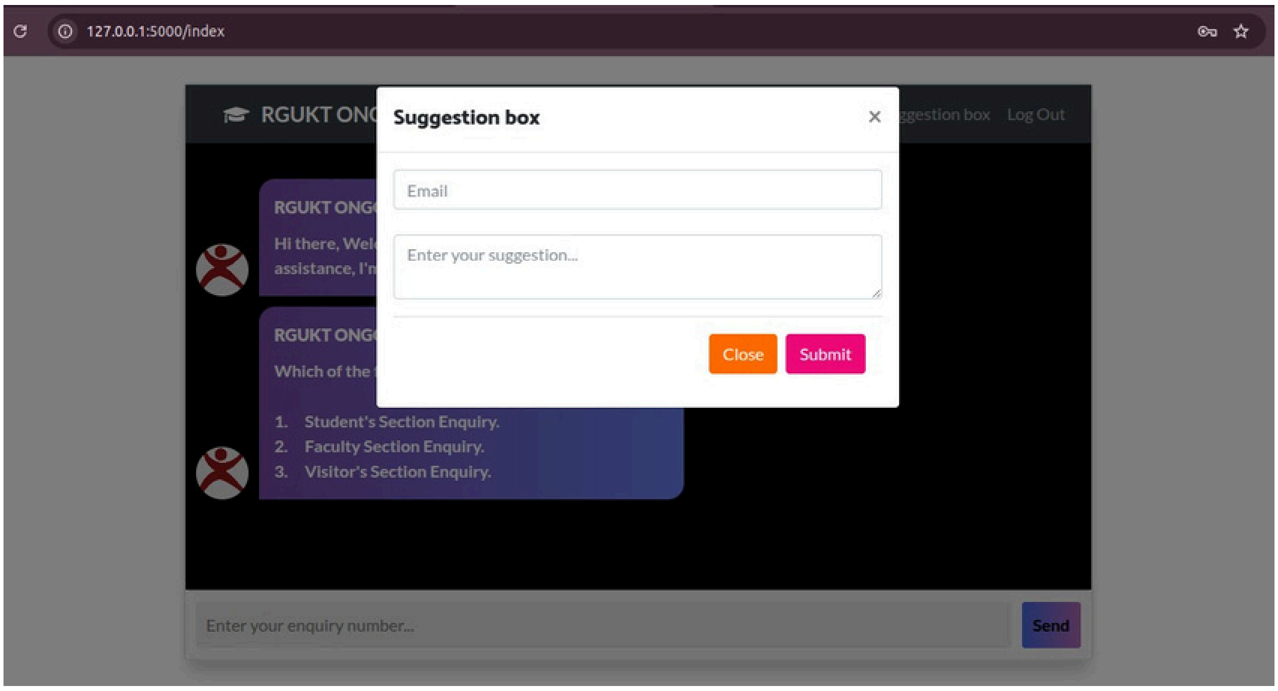
1

1.1

18:41

Enter your enquiry number...

Send


127.0.0.1:5000/index

Suggestion box

Email

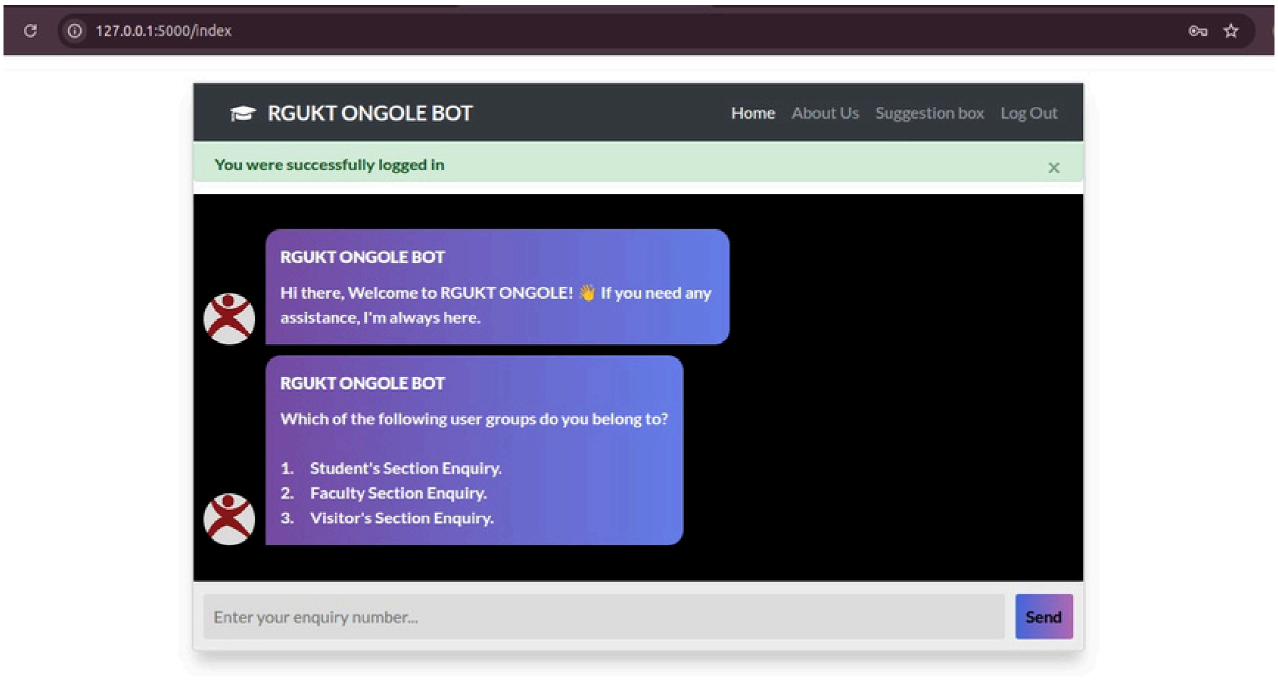
Enter your suggestion...

Close Submit

Student's Section Enquiry.
Faculty Section Enquiry.
Visitor's Section Enquiry.

Enter your enquiry number...

Send



127.0.0.1:5000/index

RGUKT ONGOLE BOT

Home About Us Suggestion box Log Out

You were successfully logged in

RGUKT ONGOLE BOT

Hi there, Welcome to RGUKT ONGOLE! 👋 If you need any assistance, I'm always here.

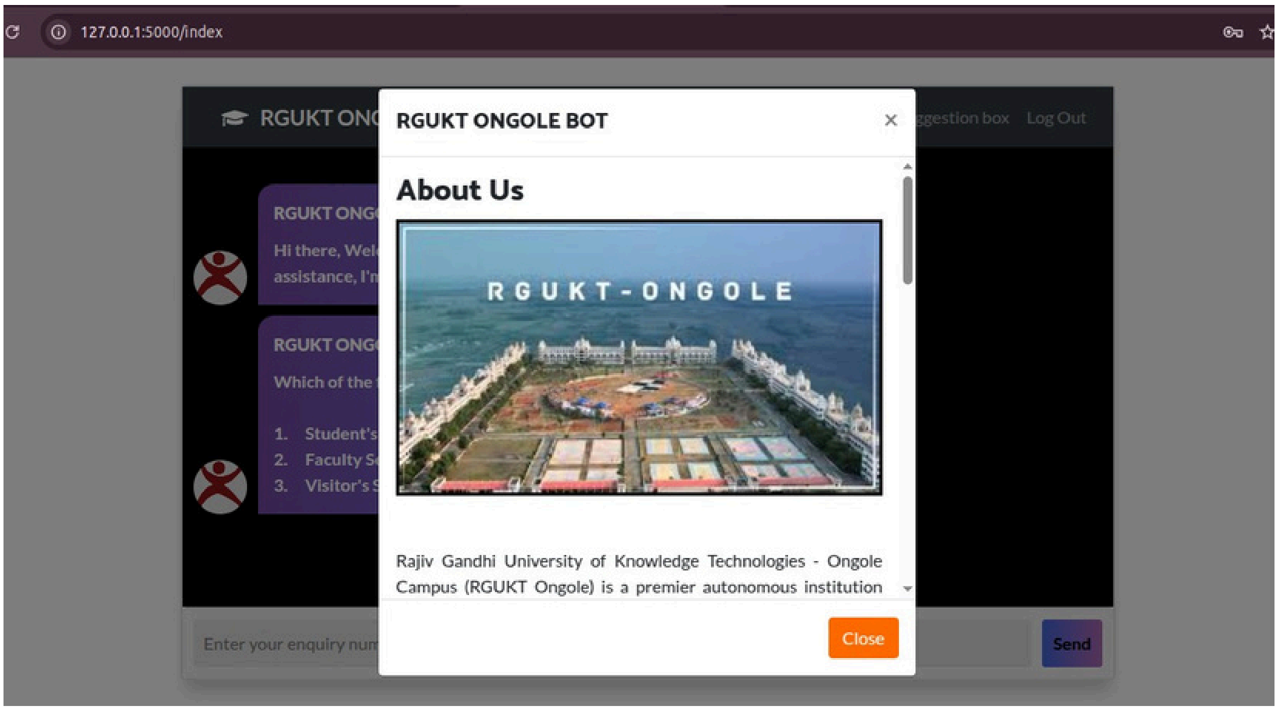
RGUKT ONGOLE BOT

Which of the following user groups do you belong to?

1. Student's Section Enquiry.
2. Faculty Section Enquiry.
3. Visitor's Section Enquiry.

Enter your enquiry number...

Send



127.0.0.1:5000/index

RGUKT ONGOLE BOT

About Us

Rajiv Gandhi University of Knowledge Technologies - Ongole Campus (RGUKT Ongole) is a premier autonomous institution

Close

7.CONCLUSION AND FUTURE WORK

The College Chatbot System has been successfully developed to streamline communication between students and the college administration. By providing quick and accurate responses to common queries, the chatbot reduces the dependency on manual support and enhances the overall user experience. It serves as a reliable and accessible platform for students to obtain essential information anytime, thus improving the efficiency of information dissemination within the institution.

Future enhancement

While the current system effectively handles frequently asked questions and basic interactions, there is significant scope for future improvements. Potential enhancements include:

- Integration with College Databases: Connecting the chatbot with internal systems (e.g., student portal, exam results, timetables) for personalized responses.
- Natural Language Processing (NLP): Improving the chatbot's ability to understand complex queries and respond in a more human-like manner.
- Voice Assistant Support: Adding voice-based interaction for hands-free communication.
- Multilingual Support: Enabling the chatbot to interact in multiple languages to accommodate a diverse student population.

8.REFERENCES

- A. Nuruzzaman, O. K. Hussain, “A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks,” 2018 IEEE 15th International Conference on e-Business Engineering (ICEBE), pp. 54-61.

- R. Sharma, M. Bhardwaj, “AI Based Chatbot for Educational Institution,” International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol. 9, Issue 3, January 2020.
- ISSN: 2278-3075

- S. J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, 3rd Edition, Pearson Education, 2010.
  ISBN: 978-0136042594

- Shawar, B. A., & Atwell, E. (2007). Chatbots: Are they really useful?
  International Journal of Knowledge-Based Systems, 20(7), 489-500.

- Patil, D., & Deshmukh, R. (2018). Chatbot using Artificial Intelligence: A Review.
  International Journal of Scientific Research in Computer Science, Engineering and Information Technology, Vol. 3(3).