CHATBOT SYSTEM FOR COLLEGE ENQUIRY USING KNOWLEDGEABLE DATABASE

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

GOOTY JOSHI NAGA VENKATA AKHILESH YADAV (REG NO: 39110343)
YASWANTH HANUMANTHU (REG NO: 39110365)

DEPARTMENT OF COMPUTER SCIENCE SCHOOL OF ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status
by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119

APRIL-2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **Gooty Joshi Naga Venkata Akhilesh Yadav (39110343) and Yaswanth Hanumanthu (39110365)** who carried out the project Phase-2 entitled "**CHATBOT SYSTEM FOR COLLEGE ENQUIRY USING KNOWLEDGEABLE DATABASE**" under my supervision from Jan 2023 to April 2023.

Internal Guide
Dr. P. Asha, M.E., Ph.D

Head of the Department
Dr L. Lakshmanan, M.E., Ph.D

Submitted for Viva voce Examination held on 20.04.2023

Internal Examiner **External Examiner**

DECLARATION

I, **Gooty Joshi Naga Venkata Akhilesh Yadav (39110343),** hereby declare that the project report entitled "**CHATBOT SYSTEM FOR COLLEGE ENQUIRY USING KNOWLEDGEABLE DATABASE"** done by us under the guidance of **Dr. P. Asha, M.E., Ph.D** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

Akhilesh yadav
Y HANUMANTHU

DATE:20.04.2023
PLACE: CHENNAI **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

We are pleased to acknowledge my sincere thanks to **the Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. T. Sasikala, M.E., PhD**, **Dean**, School of Computing, **Dr. L. Lakshmanan, M.E., PhD, Head of the Department** of **Computer Science and Engineering,** for providing us with necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to my project guide, **Dr. P. Asha, M.E., Ph.D,** for her valuable guidance, suggestions, and constant encouragement, which paved the way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

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.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S.No	ABBREVIATIONS	EXPANSION
1	AI	Artificial Intelligence
2	CNN	Convolutional Neural Network
3	GUI	Graphical User Interface
4	LSTM	Long Short Term Memory
5	NER	Named Entity Recognition
6	NLG	Natural Language Generation
7	NLP	Natural Language Processing
8	NLTK	Natural Language Tool Kit
9	UML	Unified Modeling Language

CHAPTER – 1
INTRODUCTION

1.1 GENERAL INFORMATION:

This Application is for college students, staff, and parents. Easy way to interaction and time consuming. This project is mainly targeted at colleges and the synchronization of all the sparse and diverse information regarding regular college schedule. Generally, students face problems in getting correct notifications at the correct time, sometimes important notices such as campus interview, training and placement events, holidays, and special announcements. Smart Campus tries to bridge this gap between students, teachers, and college administrators. Therefore in the real world scenario, such as college campus, the information in the form of notices, oral communication, can be directly communicated through the android devices and can be made available for the students, teachers directly for their android devices and the maintenance of application will be easier in later future because of the use of architectural MVC which separates the major works in the development of an application such as data management, mobile user interface display and web service which will be the controller to make sure for fast and efficient maintenance of application.

The College bot project is built using artificial algorithms that analyses 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 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.

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 must 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. Chatbot is a computer program that humans will interact with in natural spoken language and including artificial intelligence techniques such as NLP (Natural language processing) that makes the chatbot more interactive and more reliable.

Based on the recent epidemiological situation, the increasing demand and reliance on electronic education has become very difficult to access to the university due to the curfew imposed, and this has led to limited access to information for academics at the university. This project aims to build a chatbot for Admission and Registration to answer every person who asks about the university, colleges, majors, and admission policy. Artificial intelligence (AI) is a branch of computer science that focuses on creating machines that can perform tasks that typically require human intelligence, such as perception, reasoning, learning, and decision-making.

AI uses a combination of techniques, including machine learning, natural language processing, computer vision, and robotics, to enable machines to learn from data and adapt to new situations. In the context of a college enquiry chatbot, AI would allow the chatbot to understand and respond to natural language queries from students, providing them with relevant information and support. Artificial intelligence (AI) plays a crucial role in the development and functionality of chatbots. Chatbots are computer programs that use natural language processing (NLP) to interact with humans and simulate conversation. AI algorithms power the NLP capabilities of chatbots, enabling them to understand and respond to users' requests.

Here are some ways in which AI helps in chatbots:

Natural Language Processing: AI algorithms enable chatbots to understand natural language inputs from users and interpret them accurately. NLP algorithms analyze the text or voice input and break it down into its component parts, including keywords, entities, and intent. This analysis helps the chatbot to understand what

the user is asking and respond appropriately.

Machine Learning: AI algorithms enable chatbots to learn from user interactions and improve their responses over time. Machine learning algorithms analyze the data collected from user interactions and identify patterns and trends. Based on this analysis, the chatbot can be trained to provide more accurate and relevant responses.

Personalization: AI algorithms enable chatbots to personalize their responses based on user preferences and behavior. By analyzing user data, chatbots can tailor their responses to each user's specific needs and preferences.

Natural Language Generation: AI algorithms enable chatbots to generate natural language responses that sound human-like. Natural Language Generation (NLG) algorithms analyze the intent and context of the user's request and generate a response that is both relevant and grammatically correct.
Overall, AI is an essential component of chatbot development, enabling chatbots to deliver personalized, intelligent, and natural language-based conversations with users.

1.2 PROBLEM STATEMENT

At the start of each academic semester, registration opens for those wishing to join the university in various disciplines, and telephone calls for admission and registration abound. This leads to an increase in the loads and work for the employees of the Deanship of Admission and Registration as a result of the constant pressure of those wishing to register and their families by flocking to the Deanship, so the employees are not able to answer the phone calls and social media. This often leads to many students who wish to register to be ignored. The process of providing information and support to prospective and current students in a timely and efficient manner is a challenge for colleges, leading to frustration and dissatisfaction among users.

To achieve this, the chatbot system must be built on a robust and comprehensive

knowledge database that contains all relevant information about the college and its operations. This database should be regularly updated to ensure that the information provided by the chatbot is accurate.

1.3 OBJECTIVES

- Save effort and time for both the admission and registration staff and students who wish to enroll.
- Provide detailed information about colleges and majors.
- Easy access to information.
- To minimize the time required to solve the queries.
- To give response to the user based on queries.
- To simplify communication between user and machine.

1.4 SYSTEM ARCHITECTURE

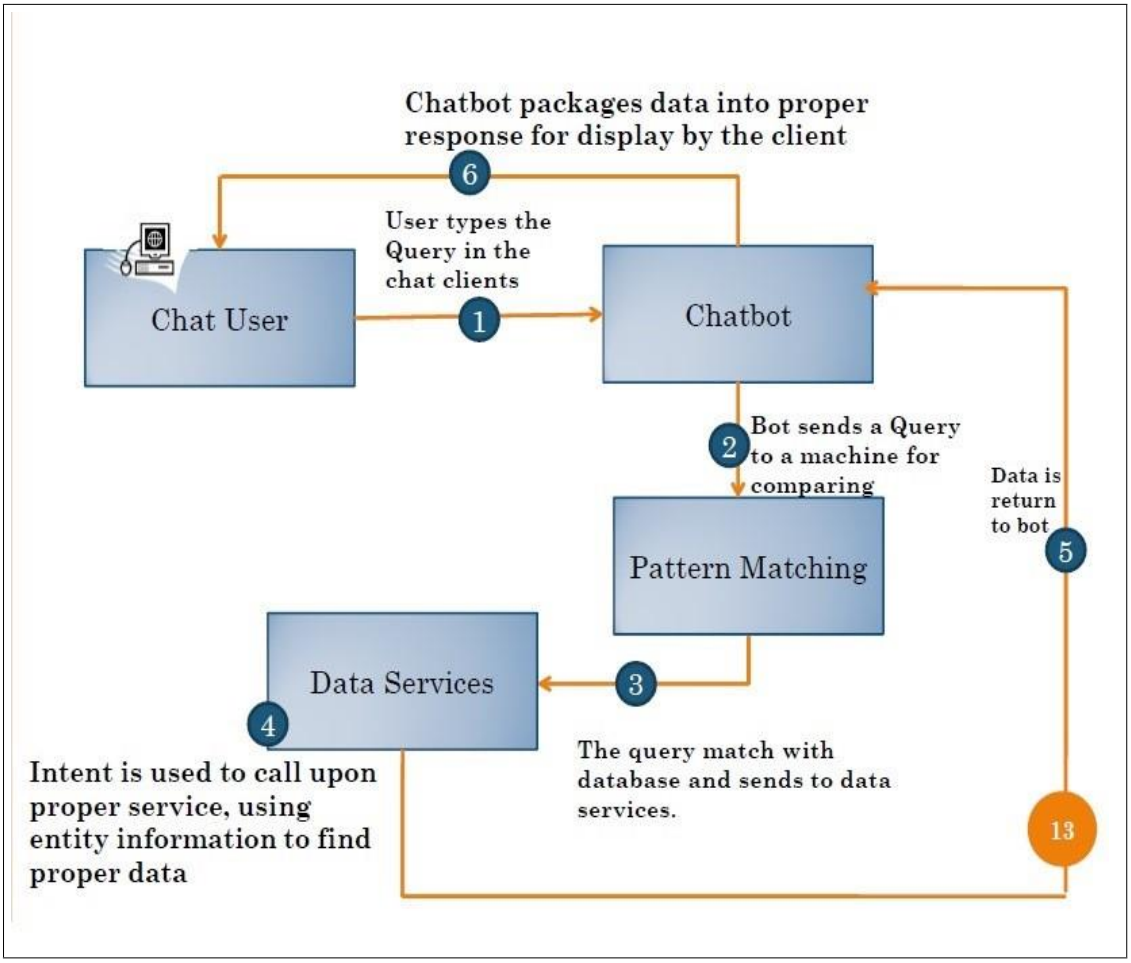

Figure 1.1 System Architecture

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. It may also show how the system operates, what are its inputs and outputs at various stages, and how the information, and/or materials flow through it. The block diagram for" Online chatting system for college enquiry knowledgeable Database" The proposed system has a client server architecture. All the information will be kept in an optimized database on the central server. This information can be accessed by the users through the android application installed on their smartphones (client machines). Each client machine will have an improved user interface.

A chatbot is a technology that allows users to have natural conversations to access content and services. Chatbots typically take the form of a chat client, leveraging

natural language processing to conduct a conversation with the user. Chatbots control conversation flow based on the context of the users requests and respond with natural language phrases to provide direct answers, request additional information or recommend actions that can be taken. The diagram below provides a high-level description of how a chat client could be used to leverage natural language processing to assist with access to content or perform data queries.

Modules Client-Server (chat user): The proposed system has a client server architecture. All the information will be kept in an optimized database on the central server. This information can be accessed by the users through the android application installed on their smartphones (client machines). Each client machine will have an improved user interface.

Chatbot: A chatbot is a technology that allows users to have natural conversations to access content and services. Chatbots typically take the form of a chat client, leveraging natural language processing to conduct a conversation with the user. Chatbots control conversation flow based on the context of the users requests and respond with natural language phrases to provide direct answers, request additional information or recommend actions that can be taken.

Pattern matching: Bot send a query to a machine for comparing. The query match with database sends to data services.

Data Services: Intent is used to call upon proper service.using entity information to find proper data. Hence all the modules are described above are completed in polynomial time sec t, So this problem is P.

1.4 STATEMENT SCOPE

In today's world as there are everything is digital. In education system work is very lengthy and time consuming and required extra manpower. We develop this application for students, teachers, parents, and guest. In this project we implement android application due to this application The Student does not have to go personally to college office for the enquiry. The application enables the students to

be updated with college cultural activities. If application saves time for the student as well as teaching and non-teaching staffs. It is useful for parents also to show his/her child marks and important notices.

1.6 NATURAL LANGUAGE PROCESSING

NLP is an interdisciplinary subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, how to program computers to process and analyze large amounts of natural language data. The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves. NLP techniques allow the chatbot to understand the natural language queries of users and provide accurate and relevant responses. NLP is a critical component of many applications that involve language, such as chatbots, voice assistants, machine translation, sentiment analysis, and more. It involves several techniques and approaches, including statistical modeling, machine learning, deep learning, and rule-based systems, to analyze and process natural language data. NLP is a rapidly evolving field, and recent advances in machine learning and deep learning have led to significant improvements in the accuracy and performance of NLP models.

Challenges in natural language processing frequently involve speech recognition, natural-language understanding, and natural-language generation.

In the early days, many language-processing systems were designed by symbolic methods, i.e., the hand-coding of a set of rules, coupled with a dictionary lookup such as by writing grammars or devising heuristic rules for stemming.

More recent systems based on machine-learning algorithms have many advantages over hand-produced rules:

- The learning procedures used during machine learning automatically focus on the most common cases, whereas when writing rules by hand it is often

not at all obvious where the effort should be directed.

- Automatic learning procedures can make use of statistical inference algorithms to produce models that are robust to unfamiliar input (e.g. containing words or structures that have not been seen before) and to erroneous input (e.g. with misspelled words or words accidentally omitted). Generally, handling such input gracefully with handwritten rules, or, more generally, creating systems of handwritten rules that make soft decisions, is extremely difficult, error-prone and time-consuming.

- Systems based on automatically learning the rules can be made more accurate simply by supplying more input data. However, systems based on handwritten rules can only be made more accurate by increasing the complexity of the rules, which is a much more difficult task. In particular, there is a limit to the complexity of systems based on handwritten rules, beyond which the systems become more and more unmanageable. However, creating more data to input to machine-learning systems simply requires a corresponding increase in the number of man-hours worked, generally without significant increases in the complexity of the annotation process.

Despite the popularity of machine learning in NLP research, symbolic methods are still (2020) commonly used:

- when the amount of training data is insufficient to successfully apply machine learning methods, e.g., for the machine translation of low-resource languages such as provided by the Apterium system,

- for pre-processing in NLP pipelines, e.g., tokenization, or

- for postprocessing and transforming the output of NLP pipelines, e.g., for knowledge extraction from syntactic parses.

Sentiment analysis: Sentiment analysis is the process of identifying the emotion or sentiment behind a user's query. This can be useful in chatbots designed to provide emotional support or mental health services. Sentiment analysis is typically done using machine learning algorithms that are trained on large datasets of text

labeled with sentiment.

Intent recognition: NLP techniques can be used to recognize the intent of user queries, allowing the chatbot to provide appropriate responses.

Named entity recognition (NER): NER is a subtask of entity recognition that focuses specifically on identifying named entities such as people, organizations, and locations. NER is commonly used in chatbots designed for customer service or support, where identifying customer names or order numbers is important.

Language translation: Chatbots can be designed to provide multilingual support by using language translation techniques. Machine learning algorithms are trained on large datasets of text in multiple languages to provide accurate translations.

Text generation: Text generation techniques can be used to generate natural language responses to user queries. These techniques use machine learning algorithms to analyze the context of the user's query and generate a relevant response. Text generation can be particularly useful in chatbots designed to handle complex or multi-turn conversations.

Natural Language Processing (NLP) techniques can play a crucial role in developing a college enquiry chatbot.

Overall, NLP techniques are critical to the success of chatbots, allowing them to accurately understand user queries and provide relevant and personalized responses.

CHAPTER – 2
LITERATURE SURVEY

Professor Girish Wadhwa suggested that the institution build an inquiry chatbot using artificial intelligence in March-April 2017. Algorithms that might analyze consumer inquiries and recognize consumer messages. This machine might be a chatbot with the intention to provide solutions to students' questions. Students actually need to pick out a category for department requests and then request a bot to be used for chat. The project's main goal is to develop an algorithm that may be used to correct the answers to queries that customers ask. It is essential to create a database where all related statistics can be kept as well as to expand the online interface. A database can develop to be able to compile information on queries, responses, key words, logs, and messages. 2016 saw Bayu Setiaji publish "Chatbot the usage of database knowledge." A chatbot is made to communicate with technology.

Machine learning is built to recognize sentences and concluded, such as the answer to a question. Personalized message, i.e. A request is saved in accordance with the response. The more similarly the statements are stated, the more it will be marked as similarity of the sentences. It is then answered in light of the answers from the first sentence. The sentence similarity calculator breaks the input sentence down into its component letters. A database stores the knowledge of chatbots. A chatbot has interfaces, and the database control system's access point through this interface is at its core. The Chatbot application was created using a variety of programming languages with the addition of a user interface that allows users to give input and get a response. Starting with the symbol of entity date, which produced 11 entities and their cardinalities, the structure and building of tables was done as an indication of the knowledge contained inside the database. SQL was used in a way that was tailored to the model that was kept inside the programme.

Elisa is regarded as the first chatbot to operate in a single machine model. Joseph Weizenbaum was the one who created it in 1964. ALICE is a rule-based chatbot that uses Artificial Intelligence Markup Language (AIML). It includes approximately 40,000 categories with an average of an example and a response for each category. A summary of chatbot programmes that have evolved through the usage of AIML

scripts was presented by Md. Shahriare Satu and Shamim-Ai- Mamun. They asserted that entirely AIML-based chatbots are easy to set up, lightweight, and eco-friendly to use. post provides information on the various ways that chatbots are used. An AIML and LSA based chatbot was created by Thomas N. T. and Amrita Vishwa to provide customer support on e-commerce platforms.

We can implement chatbots in the Android-powered device utilising a variety of techniques. In their post on Android Chatbot, Rushab Jain and Burhanuddin Lokhandwala demonstrate one method. Creating a Chatbot that Imitates a Historical Person by Emanuela Haller and Trajan Rebedea, IEEE Conference Publications, July 2013. A person with expertise in creating databases constructed the database. Yet, very few academics have looked into the idea of building a chatbot with an artificial personality and character by starting with pages or simple text about a particular person. In order to create a debate agent that can be used in CSCL high school settings, the paper discusses a method for highlighting the key information in texts that chronicle the life of a (private) historical figure.

An Introduction to Teaching AI in a Simple Agent Environment by Maya Pantik, Reinir Zwitserloot, and Robbert Jan Grootjans, IEEE Transactions on Education, Vol. 38, number three, August 2005 in this article, a flexible approach to basic the use of a novel, totally Java-based, simple agent framework developed specifically for this course to teach artificial intelligence (AI) is described. Despite the fact that many agent frameworks have been presented in a variety of literature, none of them has been widely adopted to be simple enough for first-year laptop technology college students. Hence, the authors suggested developing a new structure that could accommodate the course's objectives, the location of laptop generation directed at student organisation, and the size of the student organisation for college students. "An Intelligent Chatbot System for College Admission Process" by S. Sheikh et al. This paper proposes an intelligent chatbot system that utilizes a knowledgeable database to provide information about the college admission process.

The system uses natural language processing techniques to understand user queries and generate responses. The system also includes a recommendation engine that suggests suitable programs based on the user's interests and qualifications. The inclusion of recommendation engines further enhances the usefulness of these systems by suggesting suitable programs based on the user's

interests and qualifications.

2.1 OPEN PROBLEMS IN EXISTING SYSTEM

There are several open problems that need to be addressed in college enquiry chatbots to improve their performance and provide better user experience. Here are some of the key open problems in college enquiry chatbots:

Intent Identification: One of the primary challenges in developing a college enquiry chatbot is accurately identifying the user's intent. College enquiries can cover a wide range of topics, and the chatbot needs to correctly identify the user's intention to provide an appropriate response.

Accuracy: While chatbots can provide quick and convenient access to information, they are not always accurate in their responses. This is because the chatbot's database may not always be up-to-date, or the natural language processing algorithms may not be able to correctly interpret the user's queries.

Knowledge Base Management: A college enquiry chatbot needs to have access to a large amount of information about the college, including admission criteria, course offerings, faculty, and campus facilities. Managing this knowledge base is a significant challenge, as the information is often dispersed across multiple sources and needs to be kept up-to-date.

Natural Language Processing: Chatbots need to be able to understand and process natural language inputs accurately. However, natural language processing (NLP) technology is still in its early stages, and there are many challenges in accurately interpreting the meaning of user queries.

Multilingual Support: Colleges often have students from different parts of the world, speaking different languages. Providing multilingual support in college enquiry chatbots is a challenge that requires advanced NLP capabilities and a well-designed language model.

Personalization: To provide a better user experience, college enquiry chatbots need to personalize their responses based on the user's profile, preferences, and history. This requires advanced machine learning algorithms that can analyze user data and provide tailored responses.

Context Management: College enquiries often involve complex and multi-turn conversations. Chatbots need to be able to maintain context across these conversations to provide accurate and relevant responses.

User Engagement: Finally, chatbots need to be engaging and interactive to keep users interested and motivated to continue using them. This requires designing chatbots that can simulate human-like conversations and provide relevant and interesting information to users.

2.2 INFERENCES FROM LITERATURE SURVEY

Based on a literature survey of college enquiry chatbots, several key inferences can be drawn:

College enquiry chatbots are becoming increasingly popular: There is a growing trend of colleges and universities adopting chatbots to handle student enquiries. Several studies have shown that chatbots can significantly reduce the workload on college administrators and provide faster, more efficient, and personalized services to students.

Natural language processing (NLP) is a critical component of college enquiry chatbots: NLP technology is used to understand and interpret user queries, and to generate natural language responses. Several studies have focused on improving the accuracy and effectiveness of NLP in college enquiry chatbots.

Machine learning (ML) algorithms are being used to improve the performance of college enquiry chatbots: ML algorithms are used to train chatbots on large datasets of student queries and responses. This helps chatbots to learn from past interactions and provide more accurate and relevant responses.

Chatbots are being used to support a wide range of college enquiries: College enquiry chatbots can handle a wide range of enquiries, including admission inquiries, course registration, financial aid, campus facilities, and career services. The success of college enquiry chatbots depends on effective design and development, including careful consideration of user needs, the use of appropriate testing and optimization.

Multilingual support is a growing area of research in college enquiry chatbots: Many colleges and universities have a diverse student population, and providing multilingual support is essential to ensure that all students can access the information and services they need.

Context management is a critical challenge in college enquiry chatbots: College enquiries often involve complex and multi-turn conversations. Chatbots need to be able to maintain context across these conversations to provide accurate and relevant responses.

User experience is a critical factor in the success of college enquiry chatbots: Chatbots need to be engaging, interactive, and easy to use to keep students interested and motivated to use them. Several studies have focused on designing chatbots that can simulate human-like conversations and provide a personalized experience to users. College enquiry chatbots can help colleges to provide more efficient and effective customer service to prospective and current students, and can improve the overall user experience.

Chatbots can be integrated with a variety of channels, including websites, social media platforms, and messaging apps, to provide convenient access to information. Natural language processing (NLP) and machine learning (ML) techniques are commonly used in college enquiry chatbots to understand user queries and provide relevant responses. The use of chatbots in college enquiry systems can help to reduce workload for administrative staff and free up time for more complex tasks.

Personalization is an important feature of successful college enquiry chatbots, and can be achieved through the use of user data and machine learning techniques. Chatbots can help colleges to collect valuable data on user behaviour and preferences, which can be used to improve the quality of the chatbot's responses.

CHAPTER - 3

REQUIREMENT ANALYSIS

3.1 SOFTWARE AND HARDWARE REQUIREMENTS SPECIFICATION DOCUMENT

SOFTWARE AND HARDWARE REQUIREMENTS:

Hardware:

Operating system	: Windows 7 or 7+
RAM	: 2 GB MEMORY
Hard disc or SSD	: More than 500 GB
Processor	: Processor Dual Core

Software:

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

3.2 SYSTEM USE CASE

A college enquiry chatbot can have several use cases, including:

Admission Enquiry: The chatbot can provide information about the admission process, eligibility criteria, important dates, and documents required for admission.

Course Information: The chatbot can provide detailed information about the courses offered by the college, including the duration of the course, syllabus, fees, and career opportunities.

Campus Facilities: The chatbot can provide information about the various facilities available on the college campus, such as libraries, laboratories, sports facilities, and accommodation options.

Fees and Scholarships: The chatbot can provide information about the fees structure for different courses and scholarships available for students based on their academic performance.

Important Dates: The chatbot can remind students about important dates such as admission deadlines, fee payment dates, and exam schedules.

FAQs: The chatbot can answer frequently asked questions by students, such as how to apply for admission, how to check the admission status.

Student life: The chatbot can provide information about student life at the college, including clubs and societies, extracurricular activities, and student resources.

Counseling: The chatbot can provide counseling to students regarding their career options, course selection, and academic performance.

Academic support: The chatbot can assist students with academic enquiries, including course registration, exam schedules, and study resources.

Admission and enrolment enquiries: The chatbot can assist prospective students with admission and enrolment enquiries, including deadlines, application requirements, and documentation. Overall, a college enquiry chatbot can provide a seamless and hassle-free experience for students who are looking for information about the college and its courses.

CHAPTER – 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 STUDY OF THE PROJECT

This project is mainly targeted at colleges and the synchronization of all the sparse and diverse information regarding regular college schedule. Generally, students face problems in getting correct notifications at the correct time, sometimes important notices such as campus interview, training and placement events, holidays and special announcements. Smart Campus tries to bridge this gap between students, teachers, and college administrators. Therefore in the real world scenario, such as college campus, the information in the form of notices, oral communication, can be directly communicated through the android devices and can be made available for the students, teachers directly for their android devices and the maintenance of application will be easier in later future because of the use of architectural MVC which separates the major works in the development of an application such as data man agreement, mobile user interface display and web service which will be the controller to make sure for fast and efficient maintenance of application.

A study is carried out to select the best system that meets the performance requirements. Feasibility is the determination of whether a project is worth doing or not. The process followed in making this determination is called a feasibility study. This type of study determines if a project can and should be taken. Since the feasibility study may lead to the commitment of large resources, it becomes necessary that it should be conducted competently and that no fundamental errors of judgment are made. Depending on the results of the initial investigation, the survey is expanded to a more detailed feasibility study. Feasibility study is a test of system proposal according to its work-ability, impact on the organization, ability to meet user needs, and effective use of resources. The objective of the feasibility study is not to solve the problem but to acquire a sense of its scope. During the study, the problem definition is crystallized and aspects of the problem to be included in the system are determined.

Save timing of students and teachers and also save extra manpower. Student can see all document related college like, notice, study material, question papers etc. on

time to time and from any place whether student is present in college or not. And also reduce the work of staff. It is proper communication in between staff and students.

Natural language processing algorithms: To interpret user queries and generate accurate responses.

Knowledgeable database: To store information about college programs, courses, and admission requirements.

Recommendation engine: To suggest suitable programs based on the user's interests and qualifications.

User interface: To provide a user-friendly and intuitive interface for users to interact with the chatbot.

Data collection and processing: To gather and organize information about college programs, courses, and admission requirements.

Algorithm development: To develop natural language processing algorithms that can interpret user queries and generate accurate responses.

Database design and implementation: To design and implement a knowledgeable database that can store and retrieve information about college programs, courses, and admission requirements.

User interface design and implementation: To design and implement a user interface that is intuitive and user-friendly.

Testing and evaluation: To test the chatbot system for accuracy, usability, and performance.

4.2 EXISTING METHODOLOGY

Knowledge graph creation: The first step is to create a knowledge graph that contains all the relevant information about college programs, courses, and admission requirements. This can be done using existing ontologies or by manually curating the knowledge graph.

4.2.1. To develop the problem under consideration and justify feasibility using concept of knowledge canvas and IDEA matrix.

I	D	E	A
Increase	Drive	Educate	Accelerate
Improve	Deliver	Evaluate	Associate
Ignore	Decrease	Eliminate	Avoid

TABLE 4.1 – IDEA

Learning objective: 1. Project feasibility

- Project feasibility
- Find Knowledge gap
- Learn IDEA matrix
- Knowledge canvas

IDEA Matrix:

IDEA matrix is nothing but a matrix representation of characteristic requirement of the project.

The IDEA matrix of our project can be thus represented as:

I	D	E	A
Increase efficiency of Search Engine.	Drive a search Engine which is smart enough to be search relevant search.	Educate the human to how to search appropriate result	Accelerate speed of Searching result.
Improve relevant search result.	Deliver the exact result of search with help of Smart crawler.	Evaluate technical advancements of society for its betterment.	Associate database with Inventory system.
Ignore irrelevant result.	Decrease visiting to unwanted link of our search result.	Eliminate large amount of processing efforts.	Avoid processing in maintaining daily records of the database

TABLE 4.2 – IDEA MATRIX

Brief explanation about each characteristic:

Increase: In our project we are thus increase the use and operating efficiency of

current search engine. We are increasing searching capacity of the relevant result.
Improve: Improve the traditional search engine by making it smarter using technologies such as Smart Crawler.
Ignore: We are ignoring the irrelevant result of given searches. Our traditional search engine gives both results relevant and irrelevant searches among from them we take relevant search using smart technologies like smart crawler.
Drive: Hereby we are driving a smart search engine against a traditional search engine which helps us reducing extra search efforts.
Deliver: We are delivering a quick and easy solution for the maintenance of database that needs to be updated on regular interval.
Decrease: The extra visit to unwanted result will be decreased by using Smart Crawler and profession login option also provided on the smart crawler.
Educate: We are trying to make the management authority and efficiency of search engine aware of technical advancements around.
Evaluate: By considering the searching on internet reviews and requirements which needs to be satisfied given by the users we are evaluating the technology to be used along with algorithms needs to reduce efforts.
Eliminate: By implementation of smart crawler need for massive number of system processing is eliminated which leads to efficiency.
Accelerate: Searching is done at much higher speed as there would be we are using smart technologies and algorithms so that it removes unwanted results.
Associate: Here we are associating or linking database with the inventory so that if the sites go below threshold level inventory must make required arrangements so that the sides should not be unavailable.
Avoid: If any irrelevant search result in updating database goes may lead to wrong search result in the system. This needs to be avoided. Hence an updating mechanism is added with help of smart crawler.

KNOWLEDGE CANVAS:

Knowledge canvas is a graphical representation of knowledge gap between any two components of the project considered.

- Knowledge canvas Diagram

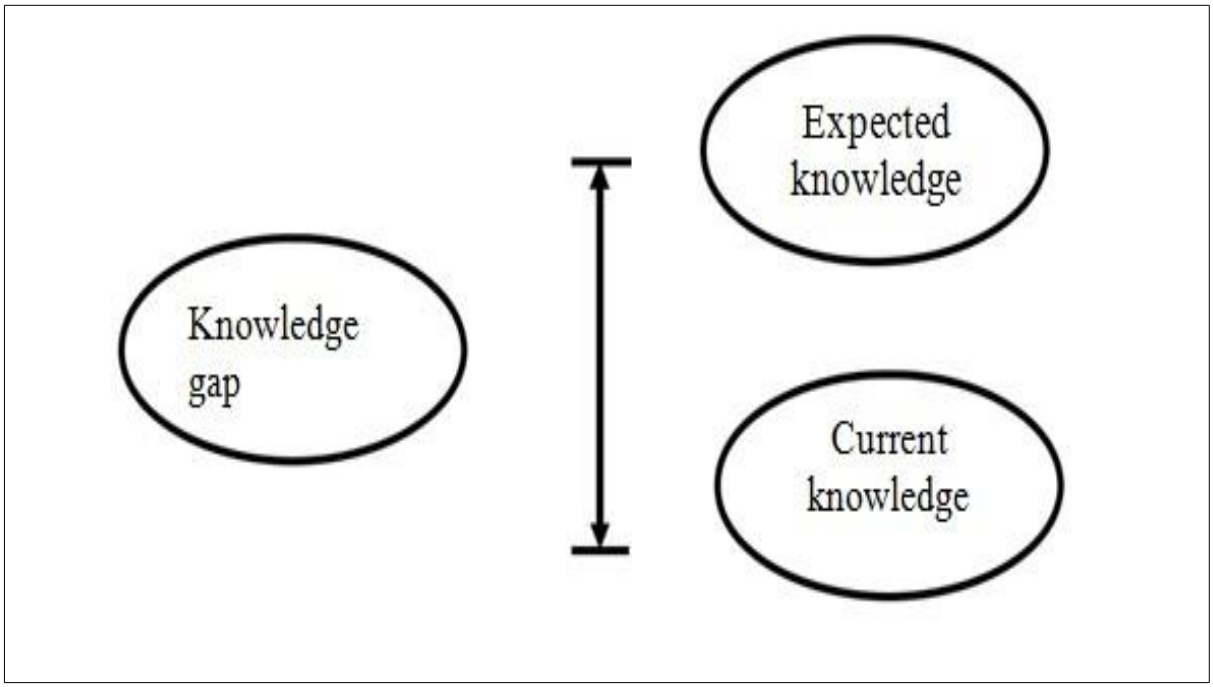

Fig 4.1 Knowledge Canvas Diagram

4.2.2. Project problem statement feasibility assessment using NP-Hard, NP-Complete.

P

Polynomial time solving. Problems which can be solved in polynomial time, which take time like O(n), O(n2), O(n3). E.g.: finding maximum element in an array or to check whether a string is palindrome or not.

So, there are many problems which can be solved in polynomial time.

NP

Non deterministic Polynomial time solving. Problem which can't be solved in polynomial time like TSP(travelling salesman problem) or An easy example of this is subset sum: given a set of numbers, does there exist a subset whose sum is zero?. But NP problems are checkable in polynomial time means that given a solution of a problem , we can check that whether the solution is correct or not in polynomial time.

NP-hard

If a problem is NP-hard, this means I can reduce any problem in NP to that problem. This means if I can solve that problem, I can easily solve any problem in NP. If we could solve an NP-hard problem in polynomial time, this would prove P = NP.

NP-complete

A problem is NP-complete if the problem is both
NP-hard, and
In NP.
Algorithms & Techniques:
Algorithm 1: Exact Pattern Matching Algorithm 2: OCR-Optical Character Recognition Time Complexity:
It takes time to fetch URL from web-server, also to extract query entered by user. It takes data from database as well as from log file so
Time Complexity=o(n)
OCR-Optical Character Recognition
Complexity Analysis
Algorithm 1: Exact Pattern Matching Algorithm
O(N + K).
Algorithm 2: OCR-Optical Character Recognition
O (N 2 log (N)).
Overall time required: O(N+K) +O (N 2 log(N)) Space Complexity:
More the storage of data more is the space complexity. Each time we store resultant data in log file also in database. We store URL (bookmarked) in database. So, more time complexity.

4.2.3. Project problem statement satiability issues using modern algebra and/or relevant mathematical models. Mathematical Model

System S is defined as S = LP, i ,U , A, I, O,T1,Su,F
Input:
Login Process LP = lp1, lp2, lpn
Where, LP is the set of login users and lp1, lp2, lp3, ...,lpn are the number of users.
Query i = i1, i2, , in Where, I is the set of queries and i1, i2, i3, ...,in are the number individuals query.
A=Admin.
U=Set of users
U=St,P,T,G,
St=set of Students= St1, St2...
P=set of Parents = p1, p2...

T=set of teachers = T1, T2...

G=Guest

I=Set of Inputs.

I=I1, I2...

Where,

I1=text, I2=Audio, T1=Task Processing.

Process:

Search

Match String As follows with database:

L(i-1) =Previous [i].

L(i) L(i+1) =next[i]

Output:

Su=Data Found.

F=Data Not Found/Server Down.

Success Conditions: As per user input desired output is generated

Failure Conditions: Desired output is not obtained

4.3 PROPOSED METHODOLOGY

Admin:

Add Student: The Admin adds the student and the password is generated by the system and sent to the students Mail Id.

Add Course: The Admin is allowed to add the Course and its Subjects semester wise.

Add Timetable: The Admin is allowed to add the timetable for the course semester wise in the form of an .jpg

Add Schedule: The Admin is allowed to add the Schedule for the course semester wise in the form of an .jpg

Add Booklet: The Admin adds the booklet limited to a pdf file only.

Add Test Solutions: The Admin adds the test solutions limited to a pdf file only.

Add Vide Links: The Admin adds the video links which is a URL.

Add Weekly Marks: The Admin adds weekly marks; weekly marks are not subjecting wise and out of 25.

Add PT1/PT2: The Admin is responsible to add the marks for PT1 and PT2 which

are subject wise out of 25.
Add College related information e.g., Events, workshop doc, photos, branch info with photos. Which is useful for represent college.

Student:

Student Login: The Student is allowed to login into the App with password sent to his/her email Id and is remembered once logged In.
View Timetable: The student can check timetable limited to only his/her course and semester, it's an Image and can be pinch zoomed.
View Schedule: The student can check Schedule limited to only his/her course and semester, it's an Image and can be pinch zoomed.
View Booklet: The Student can see a list of the booklets limited to his/her course and semester which are viewed by default by Google docs.
View Test Solutions: The Student can see a list of the test solutions limited to his/her course and semester which are viewed by default by Google docs.
View Video Links: The Student can checkout video links which are directed to the dedicated web link.
View Weekly Marks: The Student can see his weekly marks and the marks are displayed as a Bar Report.
View PT1/PT2: The Student can see his marks in the form of 2 reports namely Line Chart and Pie Chart.
Line Chart is divided into 3 fragments (Highest, Average and Students
Marks) to help the student with his progress and rank Pie Chart shows only the students marks.
University Link: The link is redirected to the Web.
Text to Speech: The bot also speaks out the answer. (If student have any query student write query in text view and android app answer it in voice and also text format.)
View College related information e.g., Events, workshop doc, photos, branch info with photos. Which is useful for represent college.

Parent:

Parent Login: The Parent is allowed to login into the App with password sent to his/her email Id and is remembered once logged In.

View College related information e.g. Events, workshop doc, photos, branch info with photos. Which is useful for represent college.

View Marks: The Parents can see his/her child marks and the marks are displayed as a Bar Report.

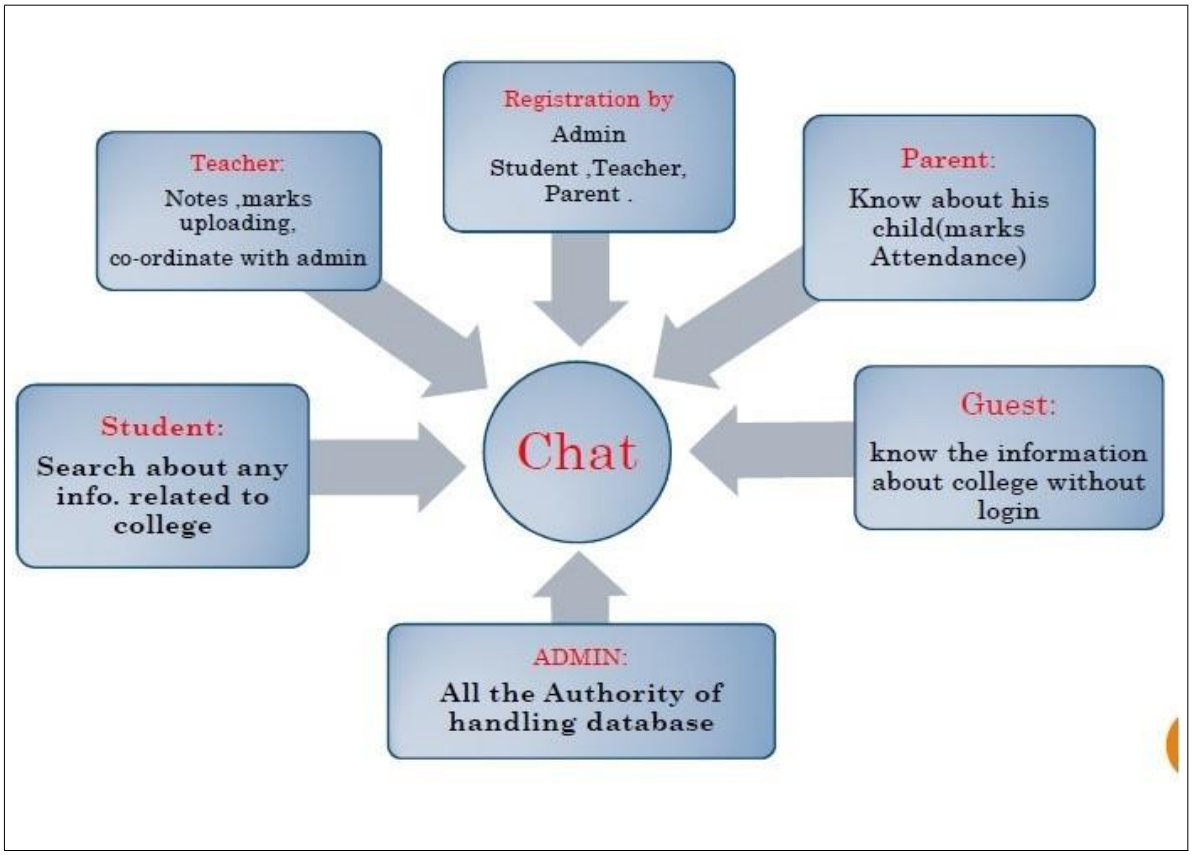

Fig 4.2 Proposed System Architecture

4.4 PROJECT TASK SET/PROJECT MANAGEMENT PLAN:

- Task 1-Requirement Gathering, Review of papers
- Task 2-Defining problem statement
- Task 3-Identifying scope and requirements of project
- Task 4-Mathematical analysis
- Task 5-System design analysis
- Task 6-UML diagrams
- Task 7-System Implementation
- Task 8-System Testing
- Task 9-Result Analysis
- Task 10-Documentation

CHAPTER – 5

IMPLEMENTATION DETAILS

5.1 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.

- **Preprocessing:** The first step is to preprocess the text inputs to extract important 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:

• Removing Stop words

• Lower casing

• Tokenization

• Lemmatization

5.1.1. 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.

5.1.2. 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.

5.1.3. 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.

5.1.4. 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.

5.2 ALGORITHMS

5.2.1. Long Short-Term Memory (LSTM)

LSTM is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. LSTM was designed by Hochreiter & Schmid Huber. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long period of time. It is used for processing, predicting, and classifying based on time-series data. Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is specifically designed to handle sequential data, such as time series, speech, and text. LSTM networks can learn long-term dependencies in sequential data, which makes them well suited for tasks such as language translation, speech recognition, and time series forecasting. A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs address this problem by introducing a memory cell, which is a container that can hold information for an

extended period. The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell. The input gate controls what information is added to the memory cell. The forget gate controls what information is removed from the memory cell. And the output gate controls what information is output from the memory cell. This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies.

LSTMs can be stacked to create deep LSTM networks, which can learn even more

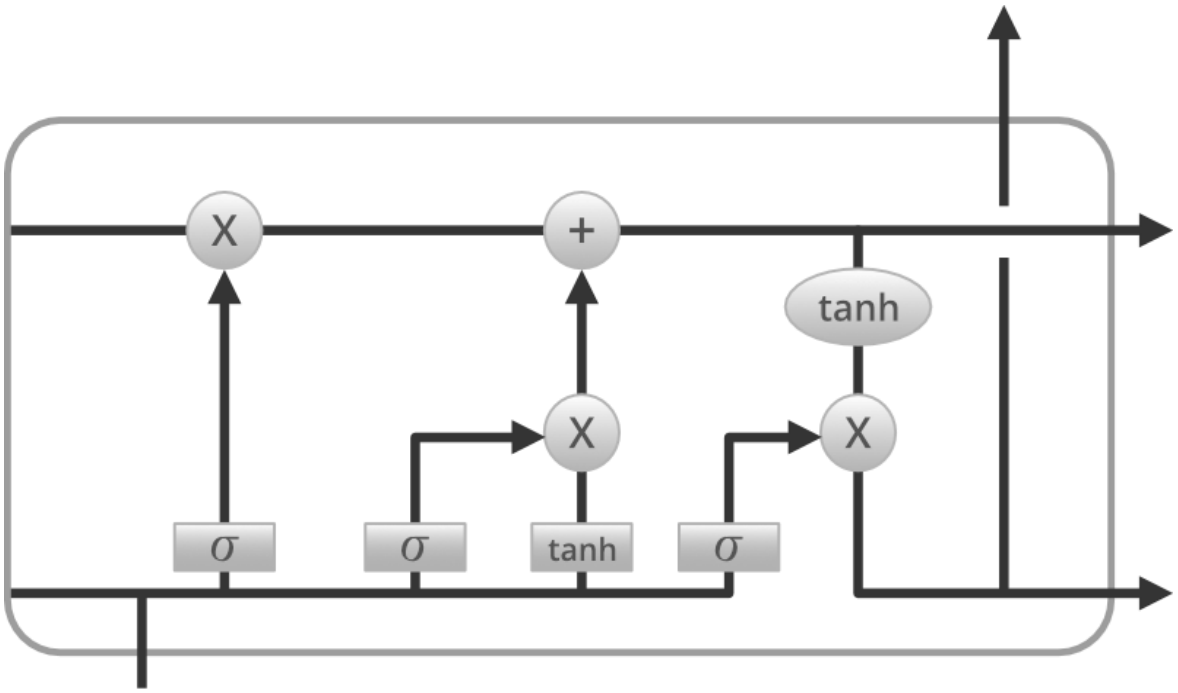

Fig 5.1 Structure of LSTM

complex patterns in sequential data. LSTMs can also be used in combination with other neural network architectures, such as Convolutional Neural Networks (CNNs) for image and video analysis.LSTM has a chain structure that contains four neural networks and different memory blocks called cells. Information is retained by the cells and the memory manipulations are done by the gates.

There are three gates –

1. Forget Gate: The information that is no longer useful in the cell state is removed with the forget gate. Two inputs x_t (input at the particular time) and h_t-1 (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use.

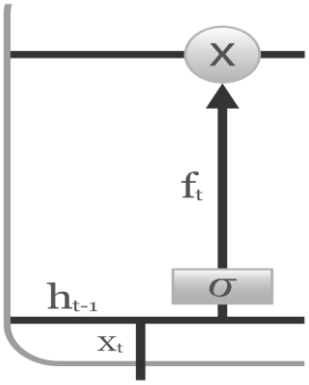

Fig 5.2 Forget Gate

2. Input gate: The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the v

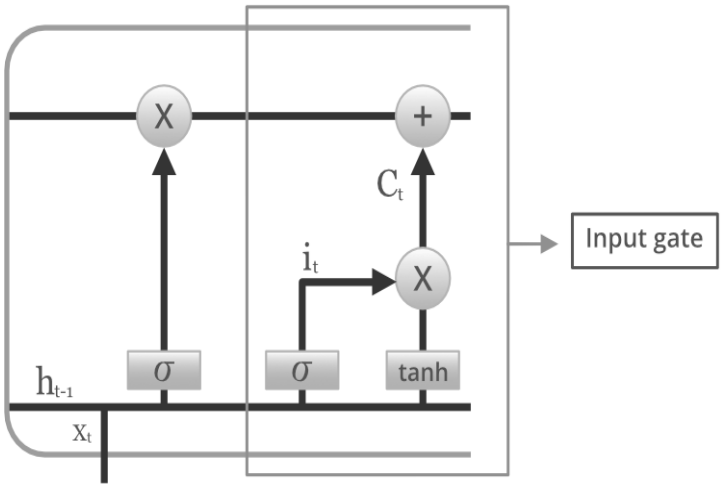

Fig 5.3 Input Gate

alues to be remembered like the forget gate using inputs h_t-1 and x_t. Then, a vector is created using tanh function

that gives an output from -1 to +1, which contains all the possible values from h_t-1 and x_t. At last, the values of the vector and the regulated values are multiplied to obtain the useful information

3.Output gate: The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying tanh function on the cell. Then, the information is regulated using the sigmoid function and filter by the values to be remembered using inputs h_t-1 and x_t. At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell.

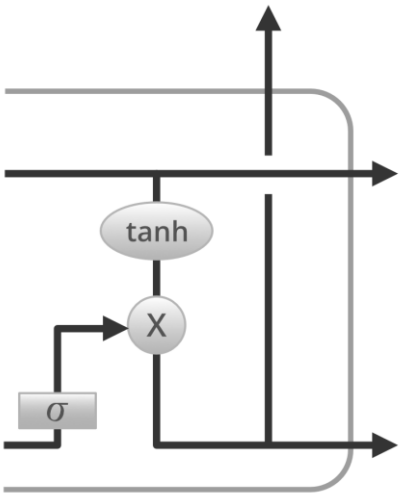

Fig 5.4 Output Gate

5.2.2 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:

Image 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.

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.

5.3 MODULE IMPLEMENTATION

5.3.1. 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 of Store 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

5.3.2. 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.

5.3.3. 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.

5.3.4. 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.

5.3.5. SPOTLIGHT

Data validation for Python, inspired by the Laravel framework. The main focus of the Spotlight library is on deep learning techniques such as matrix factorization, neural networks, and sequence modeling. It provides a flexible framework for building different types of recommendation systems, including collaborative filtering, content-based filtering, and hybrid models.

5.4 DATA FLOW DIAGRAMS

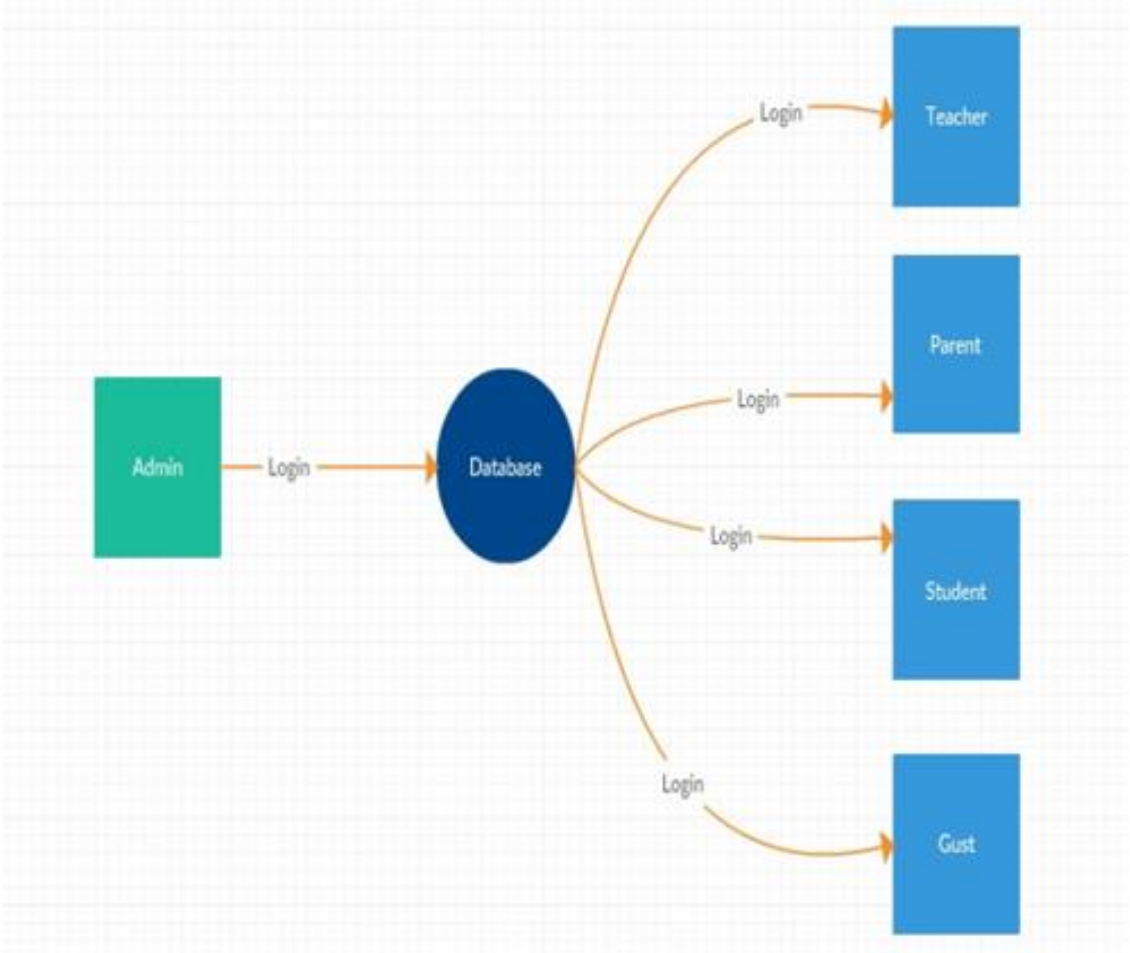

Fig 5.5.1 Level 0 Data Flow Diagram

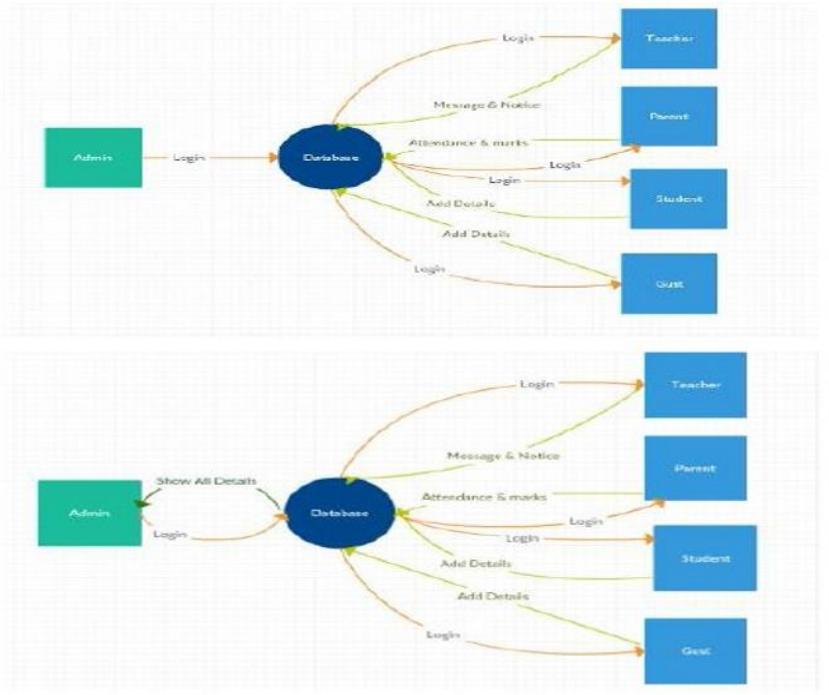

Fig 5.5.2 Level 1 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the flow of data in a system. In the context of the chatbot system for college enquiry using a knowledgeable database, a DFD can be used to illustrate the flow of data between the various components of the system. The DFD can help in understanding the data inputs, processing, and outputs of the system. The DFD for the chatbot system can be divided into four main components: the user interface, the natural language processing engine, the knowledgeable database, and the response generation component. The user interface component receives the input queries from the user in natural language. The input query is then passed on to the natural language processing engine, which processes the query and extracts relevant information such as intent, entities, and sentiment.

5.5 USE CASE DIAGRAM

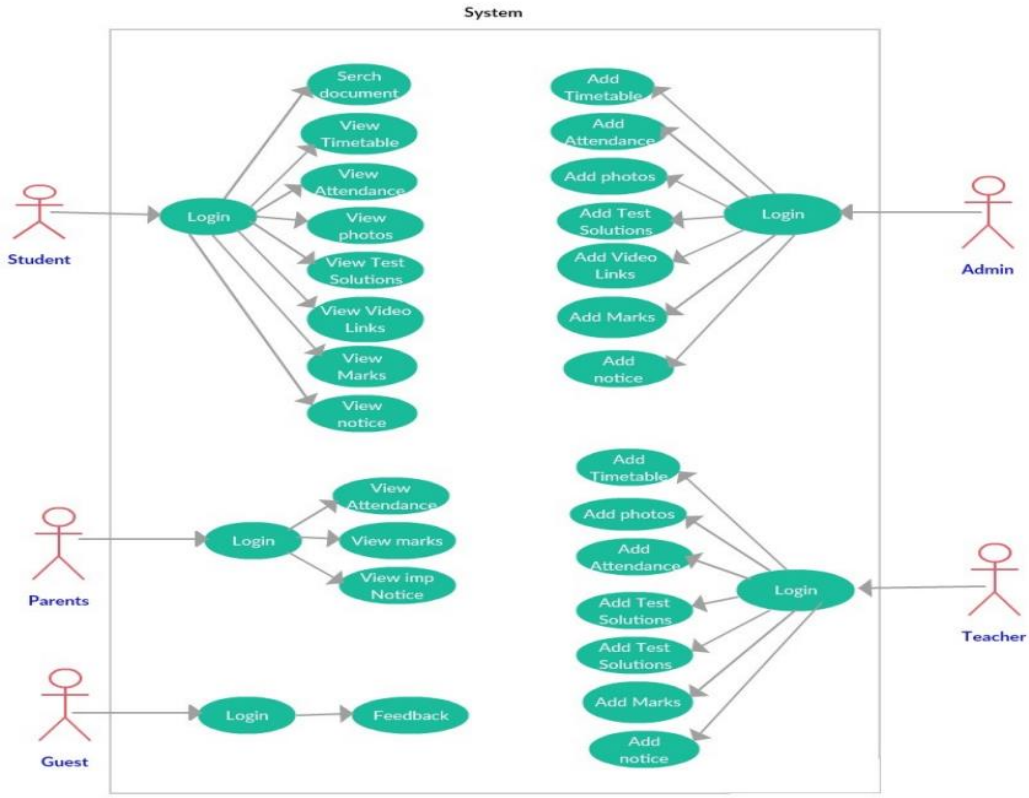

Fig 5.6 Use Case Diagram

A use case diagram is a graphical representation of the interactions between actors (users) and the system. In the context of the chatbot system for college enquiry using a knowledgeable database, a use case diagram can be used to identify the various use cases or scenarios in which the system is used.The use case diagram for the chatbot system can include the actors (users) such as prospective students, parents, and other stakeholders who are interested in obtaining information about the college. The various use cases can include querying information about courses, admission requirements, campus facilities, and other related information.

5.6 CLASS DIAGRAM

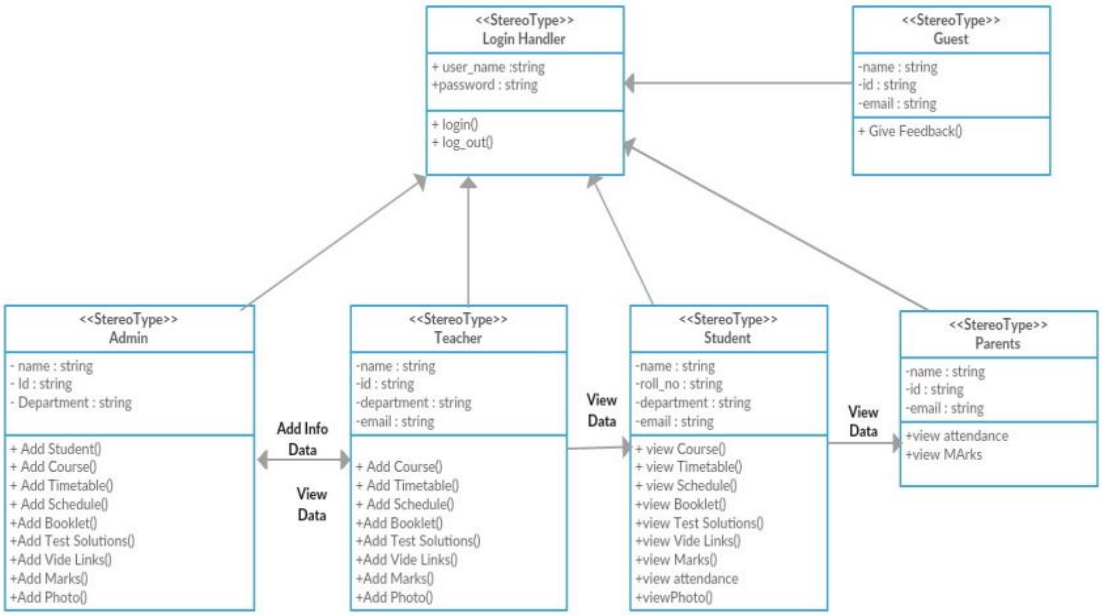

Fig 5.7 Class Diagram

A class diagram is a type of UML (Unified Modeling Language) diagram that represents the classes and their relationships in a system. In the context of the chatbot system for college enquiry using a knowledgeable database, a class diagram can be used to represent the various classes in the system and their relationships.The class diagram for the chatbot system can include classes such as User, Query, Response, Natural Language Processing Engine, Knowledgeable Database, Retrieval-based Algorithm, Rule-based Algorithm, Machine Learning Algorithm, Hybrid Approaches, and Feedback Mechanism. Each class can have attributes and methods that define its behavior and properties.

5.7 SEQUENCE DIAGRAM

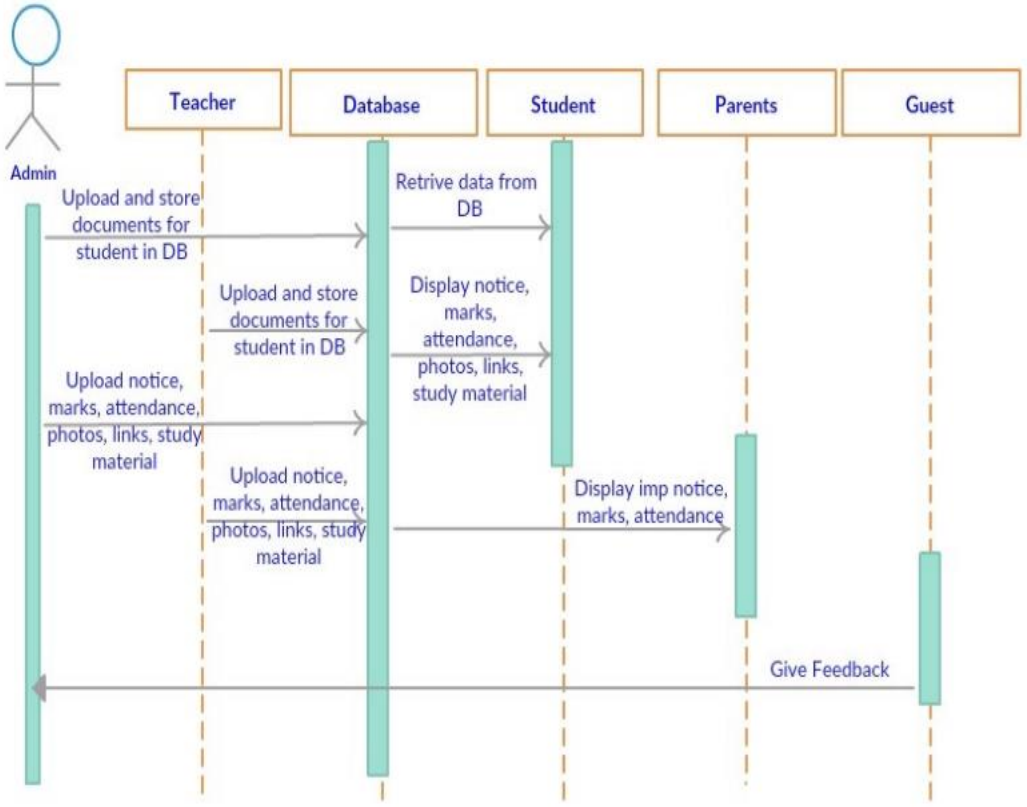

Fig 5.8 Sequence Diagram

A sequence diagram is a type of UML (Unified Modeling Language) diagram that represents the interactions between objects in a system over time. In the context of the chatbot system for college enquiry using a knowledgeable database, a sequence diagram can be used to represent the sequence of interactions between the user and the system when making a query. The sequence diagram for the chatbot system can include the user object, query object, response object, and the various algorithms and components of the system such as the Natural Language Processing Engine, Knowledgeable Database, Retrieval-based Algorithm, Rule-based Algorithm, Machine Learning Algorithm, Hybrid Approaches, and Feedback Mechanism.

5.8 COMPONENT DIAGRAM

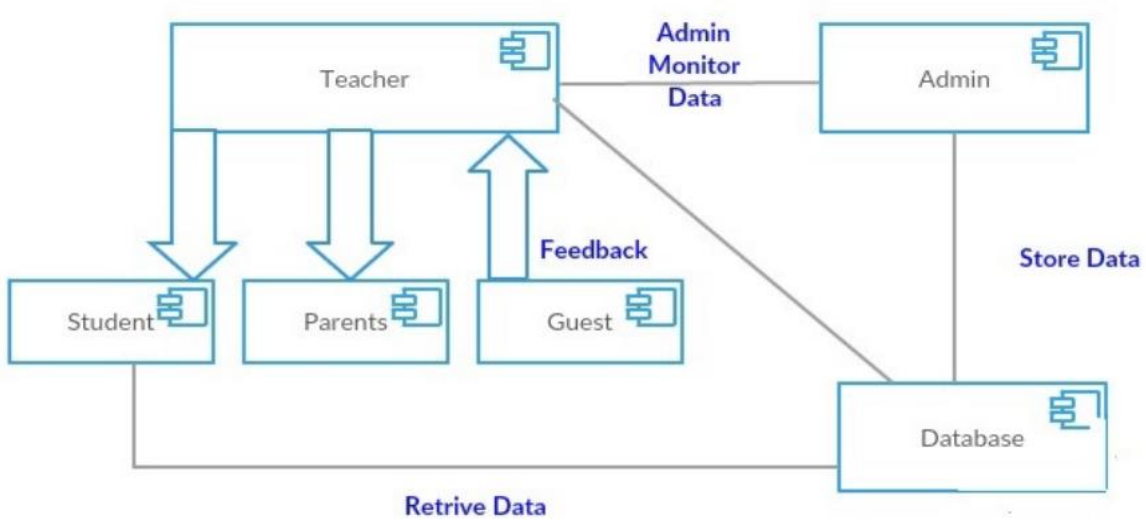

Fig 5.9 Component Diagram

A component diagram is a type of UML (Unified Modeling Language) diagram that represents the physical and logical components of a system and their relationships. In the context of the chatbot system for college enquiry using a knowledgeable database, a component diagram can be used to represent the various components of the system and their relationships. The component diagram for the chatbot system can include components such as User Interface, Natural Language Processing Engine, Knowledgeable Database, Retrieval-based Algorithm, Rule-based Algorithm, Machine Learning Algorithm, Hybrid Approaches, and Feedback Mechanism. Each component can have its own set of interfaces, ports, and dependencies that define its behavior and interactions with other components.

5.9 DEPLOYMENT DIAGRAM

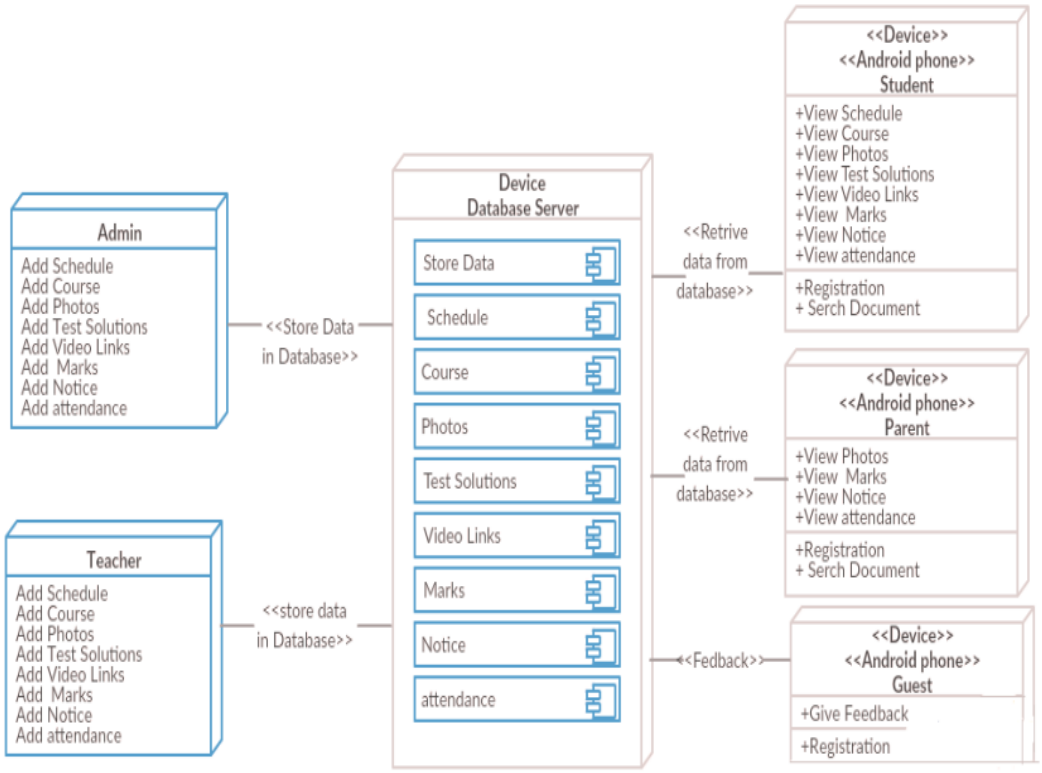

Fig 5.10 Deployment Diagram

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.

5.10 COLLABORATION DIAGRAM

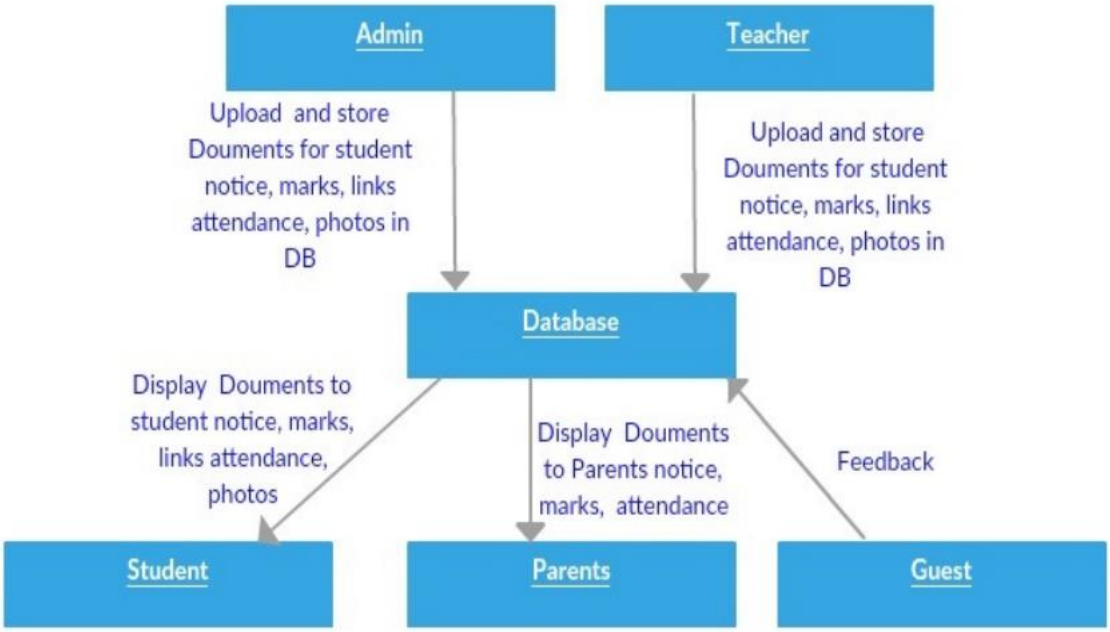

Fig 5.11 Collaboration Diagram

5.11 STATE CHART DIAGRAM

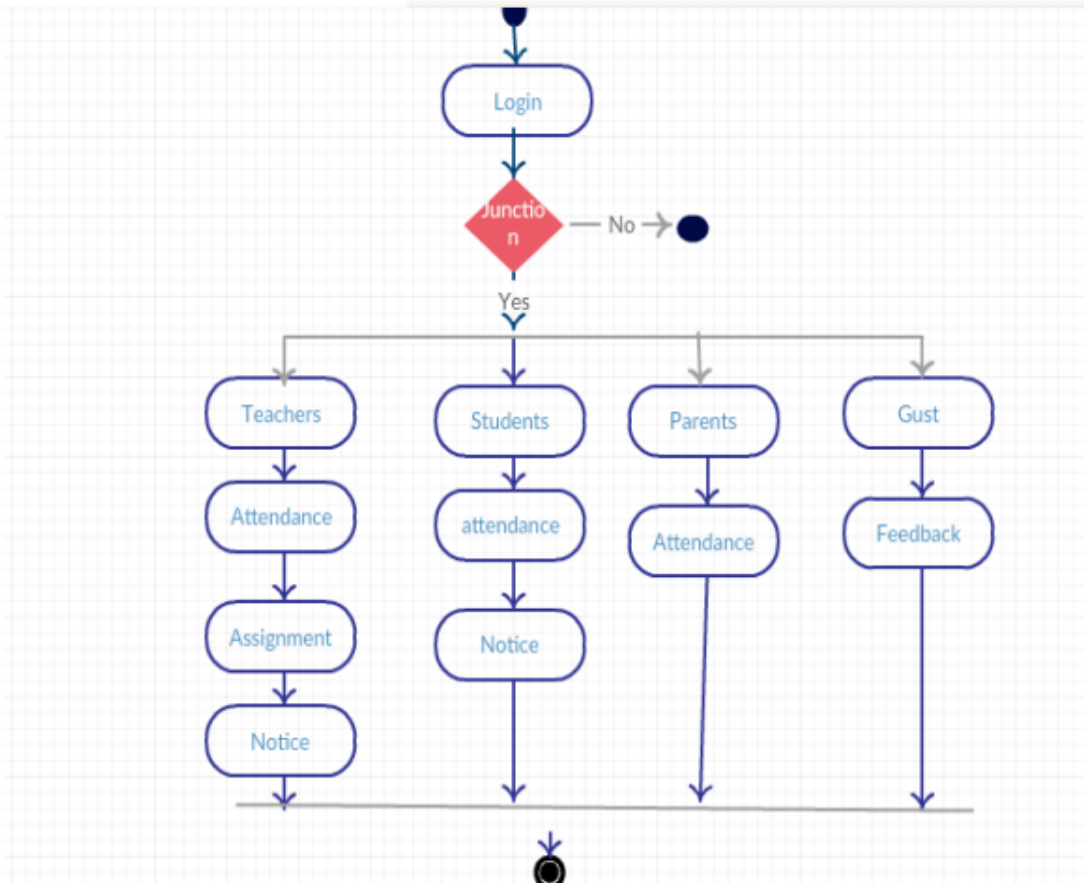

Fig 5.12 State Chart Diagram

CHAPTER-6

RESULTS AND DISCUSSION

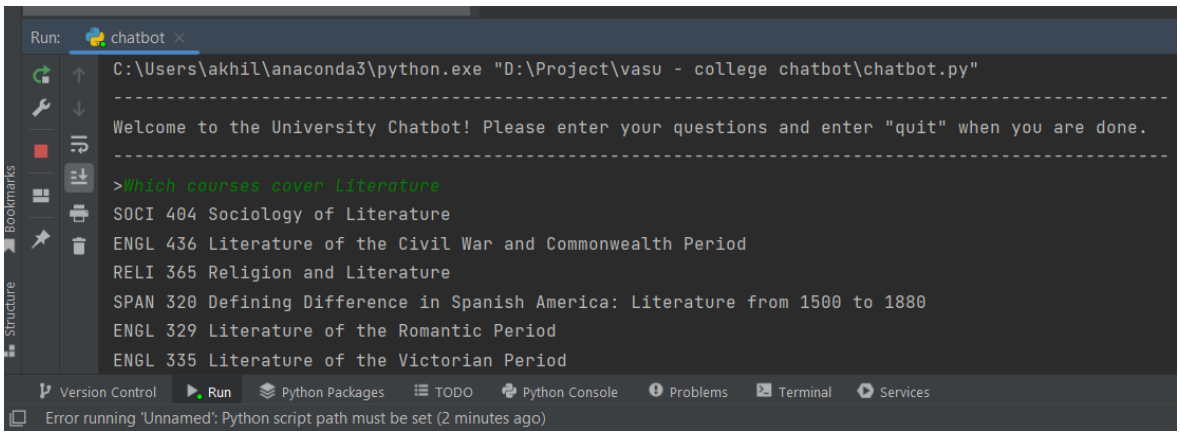

Fig.6.1. Execution (Output)

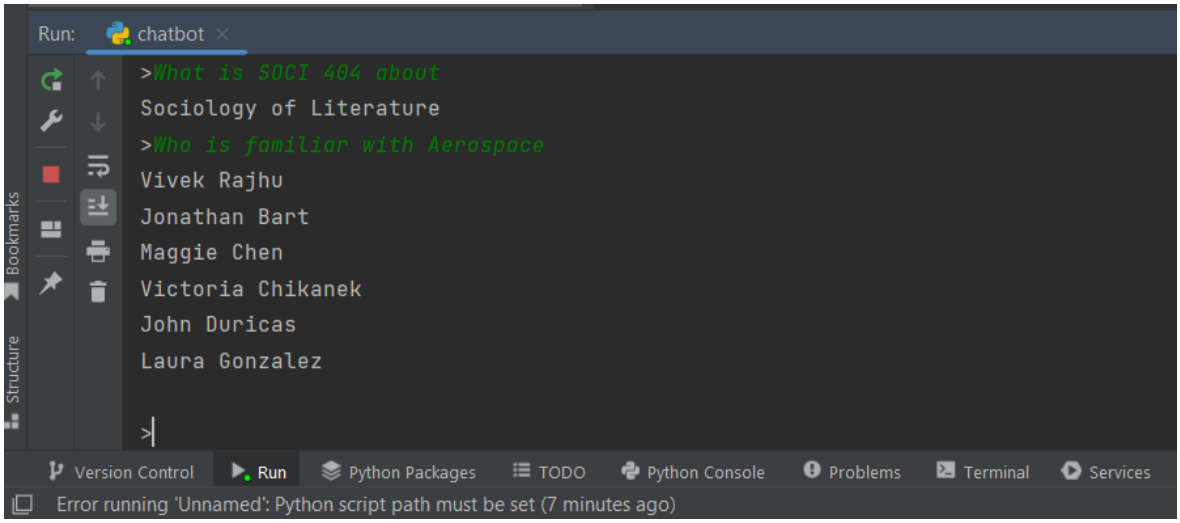

Fig.6.2. Execution (Output)

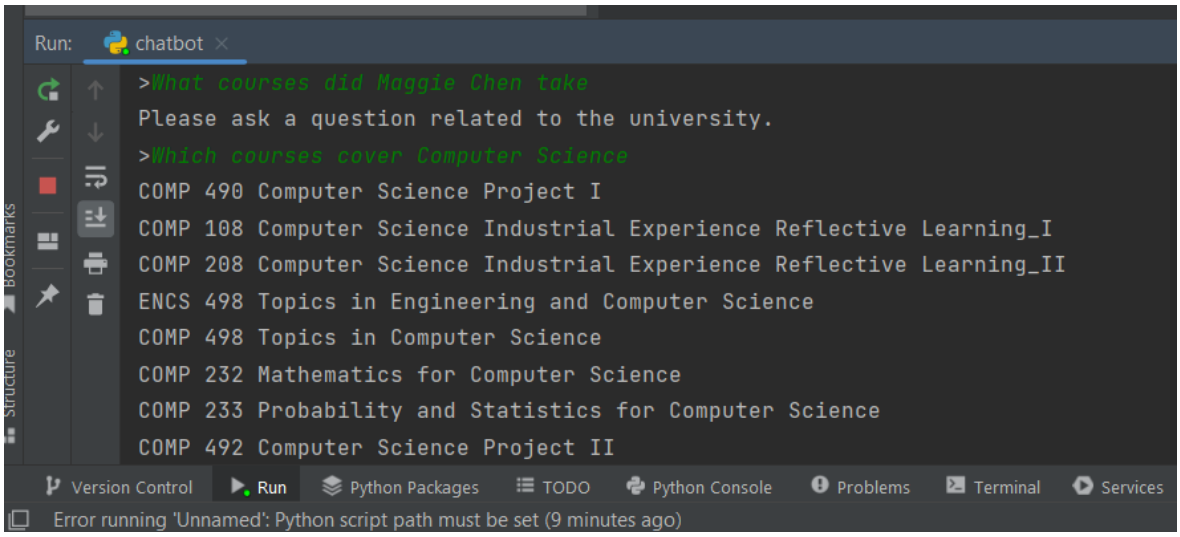

Fig.6.3. Execution (Output)

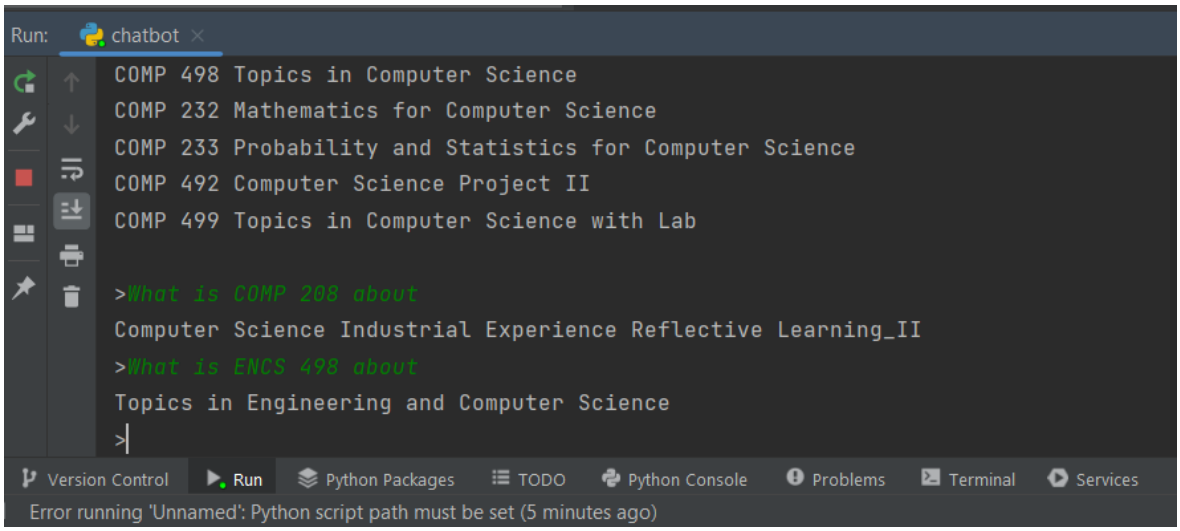

Fig.6.4. Execution (Output)

Our Chatbot provides information regarding to the college. It is the website. It is communicate to the client like guardians, understudy. By utilizing NLP human language changed into an information language. By utilizing AI to client give college data. This could be type-based (composed) discussion, even a non-verbal discussion. At the point when ChatBot innovation is incorporated with well known

web administrations it very well may be used safely by a significantly bigger crowd. Chabot framework is carried out to meet scholarly necessities of the clients. Generating reaction from a Chabot is information based one. WordNet is answerable for recovering the reactions and for this situation; it contains all rationales that is set off at whatever point the client setting is coordinated. At the point when a client starts asking questions in the Chabot Graphical User Interface (GUI). The question is looked in the information base. On the off chance that the reaction is found in the information base it is shown to the client else the framework tells the administrator about the missing reaction in the data set and gives a predefined reaction to the client. Several studies have been conducted on college enquiry chatbots, and the results suggest that chatbots can significantly improve the efficiency and effectiveness of college enquiries.
Here are some brief results and discussions from these studies:

Improved efficiency: Chatbots can significantly reduce the workload on college administrators and provide faster, more efficient, and personalized services to students. For example, a study by Turel et al. (2021) found that a chatbot developed for student admissions reduced the average response time from 3 days to less than 1 minute.

Higher user satisfaction: Several studies have found that students are generally satisfied with the performance of college enquiry chatbots. For example, a study by Stieger et al. (2020) found that students rated the chatbot developed for their university highly on ease of use, usefulness, and overall satisfaction.

Accuracy and effectiveness: Chatbots have been shown to be effective in handling a wide range of college enquiries, including admission inquiries, course registration, financial aid, campus facilities, and career services. However, accuracy and effectiveness can vary depending on the quality of the chatbot's NLP and ML algorithms.

Challenges in chatbot development: Developing an effective college enquiry chatbot is not without challenges. Challenges include accurately identifying user intent, managing a large knowledge base, providing multilingual support,

maintaining context across conversations, and ensuring a positive user experience.

Future research directions: Several research directions have been proposed for college enquiry chatbots, including improving NLP and ML algorithms, designing chatbots that can handle complex and multi-turn conversations, providing personalized recommendations and support, and developing chatbots that can handle emotional and mental health inquiries.

Overall, the results and discussions from the literature suggest that college enquiry chatbots , providing faster and personalized services to students. However, there is still much work to be done to improve the accuracy and effectiveness of chatbots and to address the challenges in chatbot development. Chatbots can gather data on user queries, preferences, and behavior, which can be used to improve the chatbot's performance and inform college decision-making. Chatbots can provide a more conversational and This can lead to increased user engagement and satisfaction. Chatbots can handle routine and repetitive enquiries, freeing up staff time to focus on more complex queries and tasks.

Chatbots can be accessed anytime and anywhere through a range of devices, making it easier for students to get the information they need. Chatbots can be designed to provide personalized responses based on the user's profile, interests, and previous interactions with the chatbot. Chatbots can handle multiple enquiries simultaneously, providing quick and efficient responses to users. However, there are also some challenges and limitations to the implementation of college enquiry chatbots. These include the need for ongoing maintenance and updates, the potential for errors in natural language processing, and the need to ensure user privacy and data protection. Additionally, chatbots may not be able to handle complex, and some users may still prefer to interact.

CHAPTER – 7

CONCLUSION

7.1 CONCLUSION

Fastest-growing technology in history is artificial intelligence. utilizing a database that is both artificially intelligent and knowledgeable. We are able to transform virtual aid and pattern matching. This method is creating a chatbot based on the Android operating system with the help of a virtual assistant and an artificially intelligent database. A chatbot that can distinguish between human and machine speech and answers to user enquiries is something we can make. Researchers must cooperate and decide on a common strategy in order to build a chatbot. In this study, we investigated the development of chatbots and their applications across several industries. Also, there are parallels with other chatbots. The knowledge base of the chatbot should generally be brief, approachable, and simple to understand. Even if some of the commercial solutions have just become accessible, there is still work to be done in order to discover a standard method for building chatbots. In conclusion, a chatbot system for college enquiry using a knowledgeable database can provide a convenient and efficient way for students, faculty, and other stakeholders to access information about college programs, courses, and admission requirements. By utilizing natural language processing algorithms, a knowledgeable database, and a recommendation engine, the chatbot system can generate accurate and relevant responses to user queries in a timely manner. Save timing of students and teachers and also save extra manpower. Student can see all document related college like, notice, study material, question papers etc. on time to time and from any place. It is proper communication in between staff and students.

7.2 FUTURE WORK

As stated in the paper, the project has a broad reach in the current context. The proposal's majority of proposed features have been implemented. So, if I continue working on this project, I intend to create a database for the system where the admin

may keep the extracted data. Further, future study will include a more in- depth examination of certain techniques, further research on other libraries, and new approaches to explore different methods.

REFERENCES

[1] CHATBOT BASED ON AI Information technology professors Nikita Hatwarl, Ashwini Patil 2, and Diksha Gondane 3123 are from Nagpur/RTMNU in India. Volume 3, Issue 2 (March-April 2016), International Journal of Emerging Trends in Engineering and Basic Sciences (UEEBS), ISSN(Online)2349-6967)

[2] Chatbot Utilizing a Knowledge in Database at the 2016 7th International Conference on Intelligent Systems, Modelling and Simulation by Bayu Setiaji and Ferry Wahyu Wibowo.

[3] Chatbot, https://en.wikipedia.org/wiki/Chatbot.Schantz, Herbert F., The History of OCR, Recognition Technologies Users Association, Manchester Center, Vermont, 1982.

[4] P. Chenna, T. Rao, et al (2018). HFSS software is used to design a twin inverted L microstrip antenna for sustainable systems. International Journal of Computer-Aided Engineering Technology (IJCAET).

[5] J. Quintero, an IEEE student member, and R. Asprilla, an IEEE member Proceedings of the 2020 IEEE Thirty-Fifth Central American and Panama Convention: Towards an Efficient Voice-Based Chatbot (concapan xxxv).

[6] Rao, P.T., and Kumar, P.P. (2015,January). Twin microstrip patch antenna in the form of a staircase. 2015 saw the ICPC (International Conference on Pervasive Computing) (pp. 1-5). The Technical Writer's Handbook by IEEE M. Young. University Science, Mill Valley, California, 1989.

[7] Rao, P. T., and Kumar, P. P. (2021). dual dual folded inverted-L antenna with a frequency-tunable circular polarisation and varactor-loaded split-ring resonator constructions. Global Journal of Communication Systems, e4715.

[8] Ly Pichponreay and Chi-Hwan Choi created the Chungbuk National

University Smart Responding Chatbot based on OCR and Overgenerating Transformations and Rating.

[9] Trinatha Rao, P., and Praveen Kumar, P. C. (2020). Using HFSS, a double folded inverted-L antenna with a rectangular ground plane is designed to be reconfigurable. IETNetworks,9(5),229-234.

[10] Automated Question Answering System Utilizing Ontology and Semantic Role International Conference on Innovative Mechanisms for Industrial Applications (ICIMIA2017) by S. Jayalakshmi and Dr. Ananthi Sheshasaayee.

[11] Yuriy Dyachenko and Nayden Nenkov Articial Intelligence Technologies For Personnel Learning Management Systems 2019 IEEE 8th Internatignal Conference on Intelligent Systems Volodymyr DahlEast Ukrainian National University Severodonetsk, Ukrainev.

[12] "Development of Chatbot System for College Enquiry" by J. Rajitha and S. Sathish Kumar. International Journal of Engineering Research & Technology, vol. 6, no. 3, March 2017.

[13] "Implementation of a Chatbot System for College Enquiry" by P. R. Ramya and M. Mythili. International Journal of Innovative Research in Computer and Communication Engineering, vol. 5, no. 5, May 2017.

[14] "Design and Development of a Chatbot for College Enquiry System" by P. R. Praveen, P. S. Sridhar, and K. Divya. International Journal of Advanced Research in Computer Science and Software Engineering, vol. 7, no. 7, July 2017.

[15] "Chatbot Based Enquiry System for College" by S. Sowmiya and S. Sathish Kumar. International Journal of Innovative Research in Science, Engineering and Technology, vol. 6, no. 8, August 2017.

[16] "Development of Chatbot for College Enquiry System using AIML" by R. Arun and A. Shanthi. International Journal of Computer Science and Mobile Computing, vol. 6, no. 9, September 2017.

[17] "Chatbot Based Enquiry System for College Admissions" by S. S. Gupta and R. K. Singh. International Journal of Computer Sciences and Engineering, vol. 6, no. 2, February 2018.

[18] "An AI-Based Chatbot System for College Enquiry" by N. Nidheesh et al. (2020). This paper describes the development of an AI-based chatbot system for college enquiry that utilizes a knowledgeable database and machine learning techniques to provide accurate responses to user queries.

APPENDIX

A. SOURCECODE

```
from rdflib import Graph, Literal

import re
import random

# generate graph of knowledge base
g = Graph()
g.parse("knowledge_base.nt", format="nt")

class eliza:
  def __init__(self):
    self.keys = list(map(lambda x:re.compile(x[0], re.IGNORECASE),responses))
    self.values = list(map(lambda x:x[1],responses))

  def translate(self,str,dict):
    words = str.lower().split()
    keys = dict.keys();
    for i in range(0,len(words)):
      if words[i] in keys:
        words[i] = dict[words[i]]
    return ' '.join(words)

  def respond(self,str):
    for i in range(0, len(self.keys)):
      match = self.keys[i].match(str)

      if match:
        resp = random.choice(self.values[i])
        pos = resp.find('%')
```

```
while pos > -1:
  num = int(resp[pos+1:pos+2])
  result = ''
  if (re.search("[wW]hat is \w+\s\d+ about?", str)):
    subject = match.group(num).split()[0]
    number = match.group(num).split()[1]
    res = g.query("""
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX ex: <http://example.org/>
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
    SELECT ?name
            WHERE {
               ?course ex:hasSubject ?subject .
               ?course ex:hasNumber ?number .
               ?course foaf:name ?name

            }
    """, initBindings={'subject': Literal(subject), 'number': Literal(number)})
    for row in res:
      result = row[0]

  if (re.search("[wW]hich courses did \w+\s\w+ take?", str)):
    student = match.group(num)
    res = g.query("""
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX ex: <http://example.org/>
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
    SELECT DISTINCT ?subject ?number ?name
       WHERE {
          ?student foaf:name ?studentName .
          ?student ex:hasCompleted ?course .
          ?course ex:hasSubject ?subject .
          ?course ex:hasNumber ?number .
```

```
            ?course foaf:name ?name .
            ex:hasCompleted ex:hasGrade ?grade
        }
    """, initBindings={'studentName': Literal(student)})
    if not res:
      result = student + ' did not take any courses!'
    else:
      for row in res:
        result += row[0] + ' ' + row[1] + ' ' + row[2] + '\n'

if (re.search("[wW]hich courses cover \w+|w+\s\w+", str)):
    topic = match.group(num)
    res = g.query("""
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
    PREFIX ex: <http://example.org/>
    SELECT ?subject ?number ?name
        WHERE {
            ?course ex:hasSubject ?subject .
            ?course ex:hasNumber ?number .
            ?course foaf:name ?name .
            ?course ex:hasTopic ?topic .
            ?topic foaf:name ?topicName
             }
    """, initBindings={'topicName': Literal(topic)})
    if not res:
      result = 'There are no courses that cover ' + topic + '!'
    else:
      for row in res:
        result += row[0] + ' ' + row[1] + ' ' + row[2] + '\n'

if (re.search("[wW]ho is familiar with \w+|w+\s\w+", str)):
    topic = match.group(num)
```

```
            res = g.query("""
            PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
            PREFIX ex: <http://example.org/>
            PREFIX foaf: <http://xmlns.com/foaf/0.1/>
            SELECT DISTINCT ?name
                    WHERE {
                       ?student ex:hasCompleted ?course .
                       ?student foaf:name ?name .
                       ?topic foaf:name ?topicName .
                       ?course ex:hasTopic ?topic
                    }
            """, initBindings={'topicName': Literal(topic)})
            if not res:
              result = 'There are no students that are familiar with ' + topic + '!'
            else:
              for row in res:
                result += row[0] + '\n'

          resp = resp[:pos] + \
            result + \
            resp[pos+2:]

          pos = resp.find('%')

        if resp[-2:] == '?.': resp = resp[:-2] + '.'
        if resp[-2:] == '??': resp = resp[:-2] + '?'
        return resp

reflections = {
  "am"      : "are",
  "was"     : "were",
  "i"       : "you",
  "i'd"     : "you would",
  "i've"    : "you have",
```

```
  "i'll"    : "you will",
  "my"      : "your",
  "are"     : "am",
  "you've"  : "I have",
  "you'll"  : "I will",
  "your"    : "my",
  "yours"   : "mine",
  "you"     : "me",
  "me"      : "you"
}

responses = [
  [r'What is (.*) about?',
  [  "%1"]],

  [r'Which courses did (.*) take?',
  [  "%1"]],

  [r'Which courses cover (.*)?',
  [  "%1"]],

  [r'Who is familiar with (.*)?',
  [  "%1"]],

  [r'quit',
  [  "Thank you for your questions.",
    "Goodbye!",
    "Thank you, that will be $100.  Have a good day!"]],

  [r'(.*)',
  [  "Please ask a question related to the university.",
    "Can you elaborate on that?",
    "I see. Do you have a question?",
    "Please ask questions about courses, students and topics."]]
```

```
]

def command_interface():
  print('-' * 100)
  print('Welcome to the University Chatbot! Please enter your questions and enter "quit" when you are done.')
  print('-'*100)

  s = ''
  chatbot = eliza();
  while s != 'quit':
    try:
      s = input('>')
    except EOFError:
      s = 'quit'
    while s[-1] in '!.':
      s = s[:-1]
    print(chatbot.respond(s))


if __name__ == "__main__":
  command_interface()
from rdflib import Graph, Literal, RDF, URIRef, Namespace
from rdflib.namespace import FOAF, RDFS, XSD

import csv

import spotlight

# define namespaces
ex = Namespace("http://example.org/")
exdata = Namespace("http://example.org/data#")

g = Graph()
```

```
# create knowledge base
g.add( (ex.University, RDF.type, RDFS.Class) )
g.add( (ex.University, RDFS.subClassOf, FOAF.organization) )
g.add( (ex.University, RDFS.label, Literal("University", lang="en")) )
g.add( (ex.University, RDFS.comment, Literal("Organization at which the students
go to")) )

g.add( (ex.Course, RDF.type, RDFS.Class) )
g.add( (ex.Course, RDFS.label, Literal("Course", lang="en")) )
g.add( (ex.Course, RDFS.comment, Literal("Course is offered at a university_data
towards the granting of an approved degree")) )
g.add( (ex.Course, FOAF.name, XSD.string) )
g.add( (ex.Course, ex.hasSubject, XSD.string) )
g.add( (ex.Course, ex.hasNumber, XSD.integer) )
g.add( (ex.Course, ex.hasDescription, XSD.string) )
g.add( (ex.Course, RDFS.seeAlso, XSD.anyURI) )
g.add( (ex.Course, ex.hasTopic, ex.Topic) )

g.add( (ex.Topic, RDF.type, RDFS.Class) )
g.add( (ex.Topic, RDFS.label, Literal("Topic", lang="en")) )
g.add( (ex.Topic, RDFS.comment, Literal("Topic is part of a course material")) )
g.add( (ex.Topic, FOAF.name, XSD.string) )
g.add( (ex.Topic, RDFS.seeAlso, XSD.anyURI) )

g.add( (ex.Student, RDF.type, RDFS.Class) )
g.add( (ex.Student, RDFS.subClassOf, FOAF.person) )
g.add( (ex.Student, RDFS.label, Literal("Student", lang="en")) )
g.add( (ex.Student, RDFS.comment, Literal("Person who studies at a
university_data")) )
g.add( (ex.Student, FOAF.name, XSD.string) )
g.add( (ex.Student, ex.hasID, XSD.integer) )
g.add( (ex.Student, FOAF.mbox, XSD.string) )
g.add( (ex.Student, ex.hasCompleted, ex.Course) )
```

```
g.add( (ex.hasSubject, RDF.type, RDF.Property) )
g.add( (ex.hasSubject, RDFS.label, Literal("hasSubject", lang="en")) )
g.add( (ex.hasSubject, RDFS.comment, Literal("Course has a subject")) )
g.add( (ex.hasSubject, RDFS.domain, ex.Course) )
g.add( (ex.hasSubject, RDFS.range, XSD.string) )

g.add( (ex.hasNumber, RDF.type, RDF.Property) )
g.add( (ex.hasNumber, RDFS.label, Literal("hasNumber", lang="en")) )
g.add( (ex.hasNumber, RDFS.comment, Literal("Course has a number")) )
g.add( (ex.hasNumber, RDFS.domain, ex.Course) )
g.add( (ex.hasNumber, RDFS.range, XSD.integer) )

g.add( (ex.hasDescription, RDF.type, RDF.Property) )
g.add( (ex.hasDescription, RDFS.label, Literal("hasDescription", lang="en")) )
g.add( (ex.hasDescription, RDFS.comment, Literal("Course has a description")) )
g.add( (ex.hasDescription, RDFS.domain, ex.Course) )
g.add( (ex.hasDescription, RDFS.range, XSD.string) )

g.add( (ex.hasID, RDF.type, RDF.Property) )
g.add( (ex.hasID, RDFS.label, Literal("hasID", lang="en")) )
g.add( (ex.hasID, RDFS.comment, Literal("Student has an ID number")) )
g.add( (ex.hasID, RDFS.domain, ex.Student) )
g.add( (ex.hasID, RDFS.range, XSD.integer) )

g.add( (ex.hasTopic, RDF.type, RDF.Property) )
g.add( (ex.hasTopic, RDFS.label, Literal("hasTopic", lang="en")) )
g.add( (ex.hasTopic, RDFS.comment, Literal("Course has a topic")) )
g.add( (ex.hasTopic, RDFS.domain, ex.Course) )
g.add( (ex.hasTopic, RDFS.range, ex.Topic) )

g.add( (ex.hasCompleted, RDF.type, RDF.Property) )
g.add( (ex.hasCompleted, RDFS.label, Literal("hasCompleted", lang="en")) )
g.add( (ex.hasCompleted, RDFS.comment, Literal("Student has completed a
```

```
course")) )
g.add( (ex.hasCompleted, RDFS.domain, ex.Student) )
g.add( (ex.hasCompleted, RDFS.range, ex.Course) )

g.add( (ex.hasGrade, RDF.type, RDF.Property) )
g.add( (ex.hasGrade, RDFS.subPropertyOf, ex.hasCompleted) )
g.add( (ex.hasGrade, RDFS.label, Literal("hasGrade", lang="en")) )
g.add( (ex.hasGrade, RDFS.comment, Literal("Student has a grade for a
completed course")) )
g.add( (ex.hasGrade, RDFS.domain, ex.hasCompleted) )
g.add( (ex.hasGrade, RDFS.range, XSD.string ) )

# processing university_data data into RDF triples
with open("dataset/university_data") as data:
    file = csv.reader(data, delimiter=',')
    for row in file:
        university = URIRef(exdata + row[0].replace(" ", "_")) # define university URI
using first column
        link = URIRef(row[1]) # define link URI to university's entry in dbpedia using
second column

        g.add( (university, RDF.type, ex.University) )
        g.add( (university, FOAF.name, Literal(row[0])) )
        g.add( (university, RDFS.seeAlso, link) )

# processing course data into RDF triples
with open("dataset/course_data") as data:
    file = csv.reader(data, delimiter=',')
    for row in file:
        course = URIRef(exdata + row[0].replace(" ", "_")) # define course URI using
first column
        link = URIRef(row[3]) # define link URI to online source of course using fourth
column
```

```
        g.add( (course, RDF.type, ex.Course) )
        g.add( (course, FOAF.name, Literal(row[0])) )
        g.add( (course, ex.hasSubject, Literal(row[1])) )
        g.add( (course, ex.hasNumber, Literal(row[2])) )
        g.add( (course, RDFS.seeAlso, link) )

        try:
            # use dbpedia spotlight to find topics
            topics = spotlight.annotate('http://model.dbpedia-spotlight.org/en/annotate',
                                row[0],
                                confidence=0.2, support=20)

            # process topics of course into RDF triples
            for topicRow in topics:
                print(topicRow)
                topic = URIRef(exdata + topicRow['surfaceForm'].replace(" ", "_")) #
define topic URI using topic's surfaceForm from result
                topicLink = URIRef(topicRow['URI']) # define link URI to dbpedia source
of the topic

                # only add topic to graph if not already in graph
                for s, p, o in g:
                    if not (topic, RDF.type, ex.Topic) in g:
                        g.add( (topic, RDF.type, ex.Topic) )
                        g.add( (topic, FOAF.name, Literal(topicRow['surfaceForm'])) )
                        g.add( (topic, RDFS.seeAlso, topicLink) )

                # add topic to this course
                g.add( (course, ex.hasTopic, topic))
        except:
            print()


# processing student data into RDF triples
```

```
with open("dataset/student_data") as data:
    file = csv.reader(data, delimiter=',')
    for row in file:
        student = URIRef(exdata + row[0].replace(" ", "_")) # define student URI using
first column
        course = URIRef(exdata + row[3].replace(" ", "_")) # define course URI using
fourth column

        # only add student to graph if not already in graph
        for s, p, o in g:
            if not (student, RDF.type, ex.Student) in g:
                g.add( (student, RDF.type, ex.Student) )
                g.add( (student, FOAF.name, Literal(row[0])) )
                g.add( (student, ex.hasID, Literal(row[1])) )
                g.add( (student, FOAF.mbox, Literal(row[2])) )

                if not (row[3] == ''):
                    g.add( (student, ex.hasCompleted, course) )
                if not (row[4] == ''):
                    g.add( (ex.hasCompleted, ex.hasGrade, Literal(row[4])) )
            else:
                if not (row[3] == ''):
                    g.add( (student, ex.hasCompleted, course) )
                if not (row[4] == ''):
                    g.add( (ex.hasCompleted, ex.hasGrade, Literal(row[4])) )

# print graph in N-Triples format to knowledge_base.nt file
# run this only once to populate the .nt file
print(g.serialize("knowledge_base.nt", format="nt"))
from rdflib import Graph, Literal

g = Graph()
g.parse("knowledge_base.nt", format="nt")
```

```
# returns total number of triples in the knowledge base
res = g.query("""
SELECT (COUNT(*) as ?triples)
        WHERE {
            ?s ?p ?o
        }
""")
for row in res:
    print("Total number of triples in the knowledge base: " + row[0])

# # returns total number of students
res = g.query("""
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ex: <http://example.org/>
SELECT (COUNT(?student) as ?count)
    WHERE {
        ?student rdf:type ex:Student
    }
""")

for row in res:
    print("Total number of students: " + row[0])

# returns total number of courses
res = g.query("""
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ex: <http://example.org/>
SELECT (COUNT(?course) as ?count)
        WHERE {
            ?course rdf:type ex:Course
        }
""")

for row in res:
```

```
    print("Total number of courses: " + row[0])

#returns total number of topics
res = g.query("""
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ex: <http://example.org/>
SELECT (COUNT(?topic) as ?count)
          WHERE {
              ?topic rdf:type ex:Topic
          }
""")
for row in res:
    print("Total number of topics: " + row[0])

# returns topics for a given course and their link to dbpedia
res = g.query("""
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ex: <http://example.org/>
SELECT ?name ?link
    WHERE {
        ?course foaf:name "Income Taxation in Canada" .
        ?course ex:hasTopic ?topic .
        ?topic foaf:name ?name .
        ?topic rdfs:seeAlso ?link
          }
""")
for row in res:
    print(row[0] + "(" + row[1] + ")")

# # returns all courses completed for a given student
res = g.query("""
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
PREFIX ex: <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?subject ?number ?name
          WHERE {
              ?student foaf:name "Dania Kalomiris" .
        ?student ex:hasCompleted ?course .
        ?course ex:hasSubject ?subject .
        ?course ex:hasNumber ?number .
        ?course foaf:name ?name .
        ex:hasCompleted ex:hasGrade ?grade
          }
""")
for row in res:
    print(row[0] + ' ' + row[1] + ' ' + row[2])

# returns list of all students familiar with a given topic
res = g.query("""
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ex: <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
          WHERE {
              ?student ex:hasCompleted ?course .
              ?student foaf:name ?name .
              ?topic foaf:name "Aerospace" .
              ?course ex:hasTopic ?topic
          }
""")
for row in res:
    print(row[0])

# # returns list of all topics a given student is familiar with
res = g.query("""
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

```
PREFIX ex: <http://example.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?name
        WHERE {
            ?student ex:hasCompleted ?course .
            ?student foaf:name "Victoria Chikanek" .
            ?course ex:hasTopic ?topic .
            ?topic foaf:name ?name
        }
""")
for row in res:
    print(row[0])
```

B. SCREENSHOTS

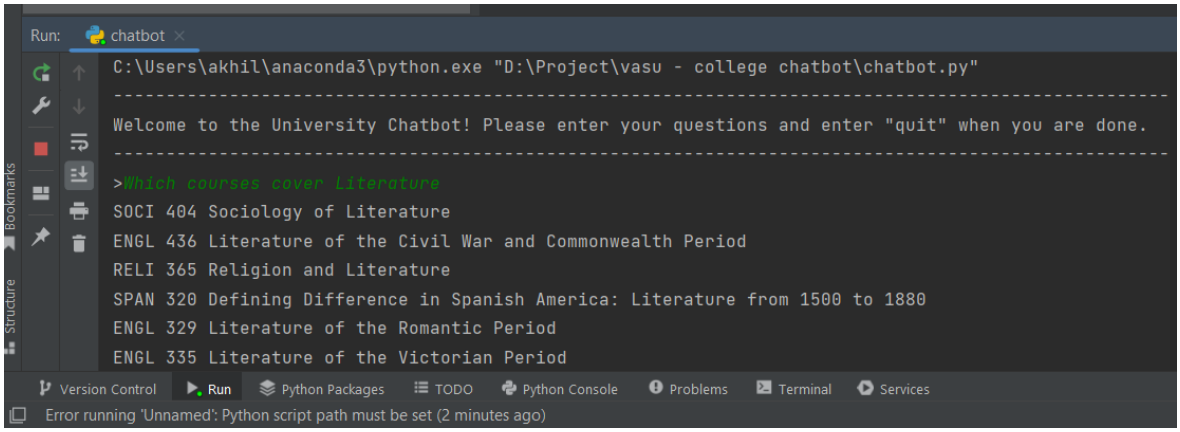

Fig.B.1. Execution (Output)

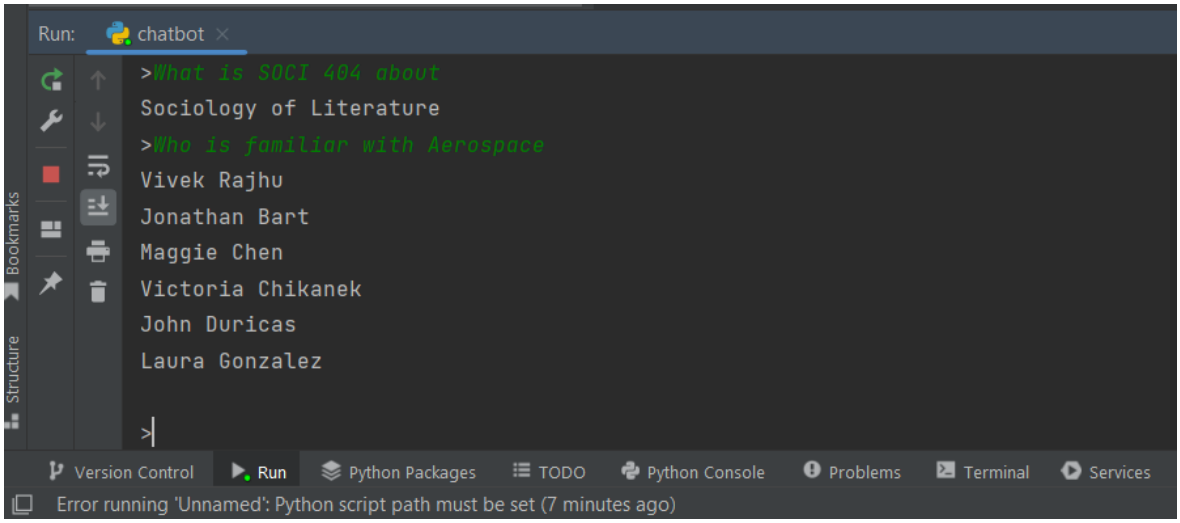

Fig.B.2. Execution (Output)

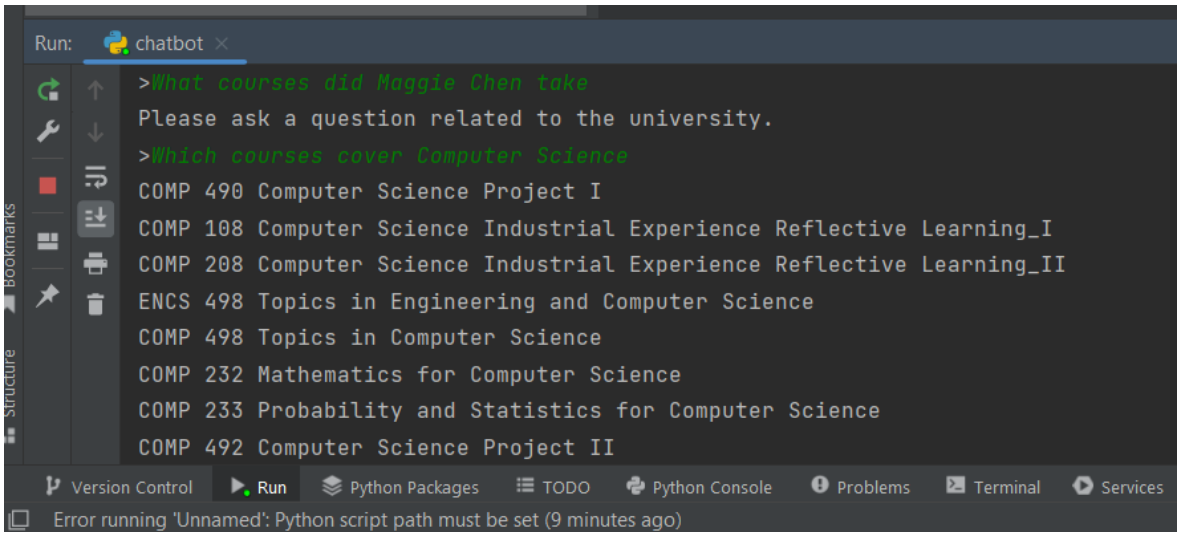

Fig.B.3. Execution (Output)

C.RESEARCH PAPER

Chatbot system for college enquiry using knowledgeable database

Gooty Joshi Naga Venkata Akhilesh Yadav
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai,India
akhiljoshi1424@gmail.com

Yaswanth Hanumanthu
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai,India
yaswanthyasw143@gmail.com

Dr. P . Asha, Ph.D.,
Department of CSE
Sathyabama Institute Of Science And Technology
Chennai,India
asha.cse@sathyabama.ac.in

***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**.

Keywords—Artificial Intelligence, Database, Intelligence Machine.

I.INTRODUCTION

Parents, employees, and college students are the intended users of this application. Time-consuming and simple method of contact. The primary audience for this project is colleges, who will benefit from the synchronisation of all the fragmentary and inconsistent data surrounding the typical college calendar. In general, students have trouble receiving accurate messages at the appropriate times, including occasionally crucial information like holidays and special announcements. Smart Campus seeks to bridge the communication gap between college management, faculty, and students. The use of architectural MVC, which divides the primary works in the construction of an application such as data management, mobile computing, and oral communication, will make the maintenance of the application simpler. In a real-world setting, such as a college campus, information in the form of notices and oral communication can be directly transmitted through android devices and made available for the students and professors on their devices. A robust graphical user interface used by the system to react gives the user the sense that they are speaking with a real person. The user only needs to logon to the system and register themselves. The user can access the numerous help pages after logging in. Several assistance pages have a chatbot that users can utilise to ask questions about college activities. The system responds to the user more quickly thanks to an effective graphical user interface. The user can request information on college-related activities through the application. The user can look up college-related events such as the day and hour of sports days, yearly days, and other social gatherings. The student can stay informed about college activities thanks to this approach. A web-based notice board will also be part of the suggested system. This notice board may be used to display any text notices or PDF files. This will help the user by informing them of important notices. The search for important notices won't take the user too long. The answer to the inquiry will take into account the user's queries as well as the knowledge base. The knowledge base will be searched for the solutions to those keywords after the important keywords have been extracted from the keywords. If a match is made, the user will be given the appropriate response or, if none is found, the notice "Response to this question is not available at the moment, please revert back after some time" will be displayed. "Keyword Matching" is used. A computer software known as a chat bot communicates with users via text or audio. It is sometimes referred to as an artificial conversational entity, a chatterbox, a talk bot, or a bot. These algorithms typically pass the Turing test

because they faithfully mimic how a human would act as a conversation partner. Chatbots are frequently employed in dialogue systems for a variety of advantageous activities, such as data collection or customer assistance. Chat bots are frequently a part of the dialogue systems of automated online assistants, enabling them to engage in quick or casual conversations that fall beyond the purview of their main expert systems. Using artificial intelligence techniques, the College Inquiry Chat Bot project will be able to comprehend user messages and analyse user enquiries. This system, which will be an Android application, will respond to the kids' questions. The department query category must first be chosen by the student before they may ask the bot a question.

II. LITERATURE REVIEW

The "College Inquiry Chat Bot" project was proposed by Prof. Girish Wadhwa in (March–April 2017) and will utilise artificial intelligence algorithms to evaluate user queries and comprehend user messages. The children's inquiries will be answered by this technology, which will be a chatbot. Before asking the chatbot a question, students will simply need to select the department category for it. The main goal of the project is to create an algorithm that will be used to find responses to questions submitted by users. Both a web interface and a database must be created, where all pertinent data will be stored. Data about requests, responses, keywords, logs, and feedback messages will be stored in a database. A database will be created to house data on queries, responses, keywords, logs, and feedback messages. Bayu Setiaji published "Chatbot Using Knowledge in Database" (in 2016). A chatbot tries to communicate with both humans and machines. The machine can recognise texts and come to a decision in response to a question thanks to built-in knowledge. The user message or query is saved while adhering to the response-theory standard. Following a match with the input sentence's replies, the similarity of the reference phrases will be assessed; the greater the score, the more similar the reference phrases are. The sentence similarity computation breaks the provided sentence down into two letters. The database houses the chatbot's knowledge. The core data in relational database management systems is accessed via the interfaces that make up the chatbot. The creation of a user interface for sending input and receiving responses was part of the process of developing Chatbot applications in a variety of programming languages. Tables representing the knowledge in the database were designed and created as a starting point for entity-relationship diagrams, which yielded 11 entities and their cardinalities. For pattern matching, the stored application had used SQL, or structured query language.

The first chatbot to employ a pattern matching algorithm is thought to be Eliza. In (1964), Joseph Weizenbaum invented it. The rule-based chatbot ALICE is built on Artificial Intelligence Markup Language (AIML). With a mix of pattern and reaction for each category, there are more than 40,000 different categories. Md. Shahriare Satu and Shamim-AI-Mamun exhibited a chatbot app evaluation using AIML scripts. They argued that chatbots powered by AIML are portable, simple to set up, and efficient. In their research, they go into great detail regarding the many chatbot applications. An AIML and LSA based chatbot was created by Thomas N. T. Amrita Vishwa to provide customer support over e-commerce platforms. Their strategy demonstrates how adding more models could enhance chatbot performance.

There are various ways to incorporate the chatbot into the Android operating system. In their study on Android-based chat-bots, (Rushabh Jain and Burhanuddin Lokhandwala) offer one way. the creation of a chatbot that embodies a historical figure Traian Rebedea and Emanuela Haller, IEEE Conference Publications, (July 2013). Conversational bots frequently store their data in databases that have been built by human professionals, despite the fact that there are numerous programmes that imitate human speech and look human.

Yet, studies have only superficially considered the possibility of building a chat-bot with a phoney personality and character based on web pages or plain text about a certain person. To build a conversational agent that may be employed in middle-school CSCL situations, the most crucial details in texts that explain the life—including the personality—of a historical figure must be determined. In this essay, a method for doing it is discussed. "Teaching Introductory Artificial Intelligence Using A Basic Agent Framework," IEEE Transactions on Education, (Vol. 48, No. 3, August 2005), by Maja Pantic, Robbert Jan Grootjans, and Reinier Zwitserloot. This article introduces a revolutionary simple agent framework implemented in Java that was created specifically for this course as the basis for an adaptable approach to teaching basic artificial intelligence (AI). Although various additional agent frameworks have been put forth in the substantial amount of research, none of them have proven to be user-friendly enough for students in their first year of computer science. The authors set out to create a novel framework in order to produce one that would be suitable for the course's objectives, the predicted group of students' level of computing proficiency, and the quantity of this group of students.

III.PROPOSED WORK

Administrator:

Include scholar: A student is given by the administrator, and the computer generates and sends a password to the student's ID number.

Course Addition: The instructor may highlight the path and its subjects during the semester.

Schedule: Administrators may display a semester course schedule as a JPEG file. Schedule addition is permitted by administrators to include a semester schedule of classes in a JPEG image.

Add a document: The administrator finds that a report looks best in a pdf format. Test solutions can best be added by the administrator in PDF format.

Add Video Links: The administrator includes URLs for videos. Weekly steps aren't an issue for the smart people, therefore the admin has added 25 more.

Including PTI/PT2:

Student Login: Using a password that is sent to his or her email address and saved in memory after login, a student is allowed access to the programme.

Calendar View: A student can view the agenda that is best suited to their direction and semester by enlarging the image.

Timetable View: This is a picture that may be enlarged, allowing a student to quickly verify their direction and semester. A student can view a list of notebooks that are specific to their course and semester in Google Docs' default notebook view.

See Test Solutions: By utilising Google Documents' default settings, A student can look through a list of exam answers according to his or her age and semester.

Seen Weekly Grades: Students have access to their weekly grades, which are shown in a closed graph. The student can view his grades in two reports—a line chart and a chart—using the PTI/PT2 interface.

Three components make up the line chart (Highest, Average and Student Grades) don't help college students advance. The basic information shown by the pie chart is the pupils' grades.

Hyperlink to the university and the Internet used to convert text-to-speech technology, the bot also speaks the solution. (If a student has a request, he or she types it into the Android app, which then replies to him or her in voice and text form). Access information linked to the university, such as events, seminar files, photos, and branch records, using Snap

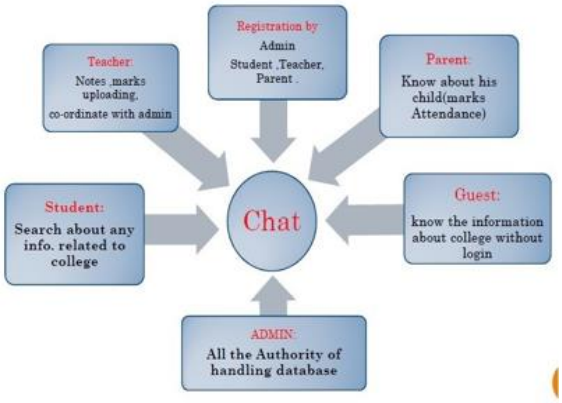

Fig.3.1. System Architecture

The block diagram for "Internet Chat System for College Enquiry Knowledgeable Database" Client-server technology underpins the suggested solution. An efficient database will house all the data. located on the main server. Via the Android app they have installed on their cell phones, consumers can view this information (client machines). There will be upgraded user interfaces on each client PC. To access material and services, consumers can use a chatbot, a technology that mimics conversational communication.

A chat client that interacts with the user using natural language processing is the most common type of chatbot. Chatbots manage the conversation's tempo and respond with natural language expressions to give quick responses, ask for further details, or suggest possible next steps based on the context of the user's needs. The image below gives a general idea of how a chat client might use natural language processing to quicken content access.

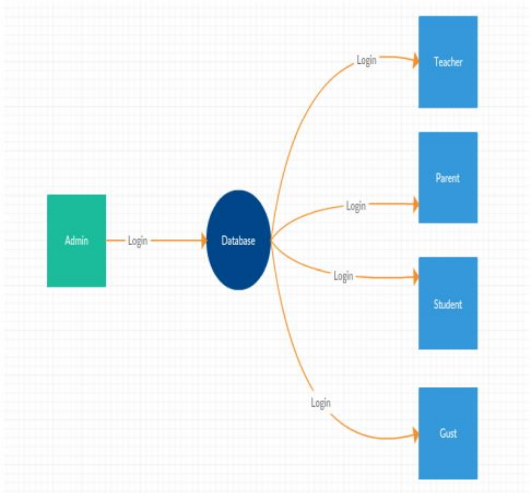

Fig.3.2. Level 0 Data Flow Diagram

In today's environment, everything is

digital. Particularly time-consuming, labor-intensive, and requiring more employees is work in the educational system. The people who will use this application the most are the kids, teachers, parents, and guests. We included an Android application in this project as a result of this application. The student does not have to come into the college office in person to make the inquiry. Students can access information about campus cultural activities through the application. Whether the application helps students, teachers, and other staff members save time. Additionally, parents should show their kids any marks and significant notes. Save time for both teachers and pupils, as well as unnecessary labour. Students have quick access to all college-related documents, such as notices, study materials, and test questions.

The first step involves obtaining a lot of important data, such as frequently asked questions, information on the courses, admissions standards, campus facilities, etc. These data will be used to train the chatbot model.

Following preprocessing, the data must be prepared for model training. This procedure may involve tokenization, stemming, and the eradication of stop words.

The Long Short-Term Memory (LSTM) model stands out among the various types of recurrent neural networks because it is proficient in processing sequential input, including text. The preprocessed data can be used to train the LSTM model to find patterns and correlations between the questions and answers.

A.Image Identification:

Image identification tasks frequently employ the CNN model, a class of neural network. By considering the text as a series of one-dimensional signals, it can also be used for natural language processing. To extract the key elements of the text, the CNN model can be trained on the preprocessed data.

B.Recognition of Intent:

The next stage is to determine the user's intention when providing input. For instance, if a user queries, "What are the admission requirements for Computer Science?" the purpose "Admission Requirements" can be deduced. Techniques like rule-based systems, machine learning algorithms like Naive Bayes, or neural network models like LSTM can be used for this.

C.Entity Recognition:

After the chatbot has identified the user's purpose, it must extract the pertinent entities from their input. The entities in the case would be "Computer Science". To accomplish this, methods like it is possible to tag words using named entity recognition (NER) and part of speech (POS).

D.Conversation Management:

The chatbot must keep the user's inputs flowing in a discussion and respond to them appropriately. Techniques like rule-based systems, finite-state machines, or reinforcement learning algorithms can be used to do this.

E.Response Generation:

After the user's input, the chatbot creates a response based on the user's purpose and the entities found in the earlier steps. The response can be a statement that is generated dynamically or follow a predefined structure. Techniques like rule-based systems, templates, or machine learning algorithms like sequence-to-sequence models or Generative Pre-trained Transformer (GPT) models can be used to create the response.

1. Problem statement: A clearly defined problem statement that served as the basis for analysis.
2. Conclusions and advice compiled: a summary of the study's main conclusions and suggestions. It is ideal for individuals who require immediate access to the research results related to the system under examination. Many recommendations are given, each with a justification, after the conclusion is stated.
3. Details regarding the outcomes: A discussion of the objectives and procedures employed by the potential system is covered after a summary of the methods and procedures used by the current system. Discussions of the prospective system's costs and benefits are also covered, as well as discussions of output reports, file structures, and other topics.
4. Suggestions and conclusions: Particular suggestions for the potential system included.

IV.EXISTING WORK

To solve satiability issues in the project problem statement, use modern algebra or relevant mathematical models. NP The project's problem statement is tested for viability using Hard and NP-Complete.

Problem polynomial problem solving. solutions to time-consuming problems like O, O(n), and O(n2)(n3) in polynomial time. For instance, determining the greatest member in an array or determining whether a string is a palindrome. As a result, many issues can be resolved in polynomial

time.

NP not a polynomial time solution that is deterministic. a conundrum that can't be solved in polynomial time, like the travelling salesman problem (TSP), Subset sum offers a simple illustration of this. Exists a subset of the given collection of numbers?

Whose total is zero? However, NP problems can be analysed in polynomial time, demonstrating that the answer is correct.

Create and put into operation an online discussion system using a database with knowledge and a translator that will be utilised for pattern matching.

In addition to the Microsoft Azure bot service, the Microsoft Cognitive Services Text Analytics, LUIS, and QnA Creator are used to create the chatbot.The majority of currently in use chatbots lack empathy and are unable to manage situations that deviate from the script. The College Inquiry Chatbot expands the use of the current chatbots by addressing these problems with sentiment analysis and active learning.

V .RESULTS AND DISCUSSION

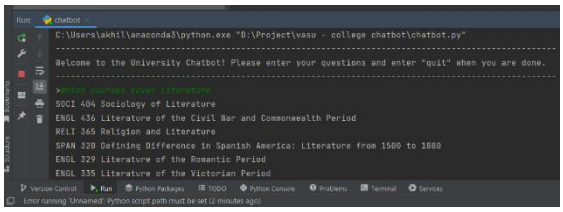

Fig.5.1. Execution (Output)

The output for a chatbot system for college enquiry using a knowledgeable database will depend on the specific design and programming of the chatbot. Generally, the chatbot would be able to answer a wide range of questions related to the college, such as admission requirements, program details, campus facilities, tuition fees, and student life.

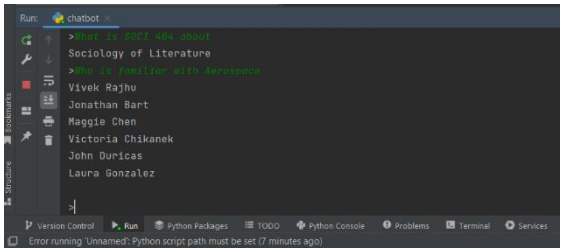

Fig.5.2. Execution (Output)

The knowledgeable database would contain information about the college, which the chatbot can access and use to respond to user queries. The chatbot may use natural language processing (NLP) techniques to understand the user's question and provide a relevant response.

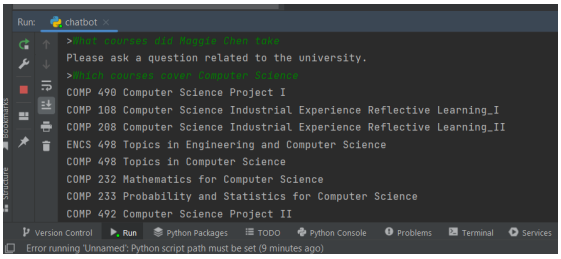

Fig.5.3. Execution (Output)

For example, if a user asks, "What are the admission requirements for the Computer Science program?", the chatbot may provide a response such as, "To apply for the Computer Science program, you must have a high school diploma, minimum GPA of 3.0, and submit your SAT scores. You may also be required to submit letters of recommendation and a personal statement."

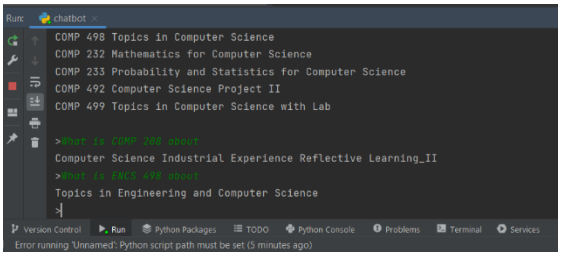

Fig.5.4. Execution (Output)

Overall, the output for a chatbot system for college enquiry using a knowledgeable database would be informative and helpful, providing users with the information they need to make informed decisions about their education.

VI. CONCLUSION

The fastest-growing technology in history is artificial intelligence. utilizing a database that is both artificially intelligent and knowledgeable. We can transform virtual aid and pattern matching. With the use of a virtual assistant and an artificially intelligent database, this technique builds an Android-based chatbot. We can develop a chatbot that answers to user enquiries and can distinguish between human and computer speech.

REFERENCES

[1] CHATBOT BASED ON AI Information technology professors Nikita Hatwarl, Ashwini Patil 2, and Diksha Gondane 3123 are from Nagpur/RTMNU in India. Volume 3, Issue 2 (March-April 2016), International Journal of Emerging Trends in Engineering and Basic Sciences (UEEBS), ISSN(Online)2349-6967)

[2] Chatbot Utilizing a Knowledge in Database at the 2016 7th International Conference on Intelligent Systems, Modelling and Simulation by Bayu Setiaji and Ferry Wahyu Wibowo.

[3] Chatbot, https://en.wikipedia.org/wiki/Chatbot.Schantz, Herbert F., The History of OCR, Recognition Technologies Users Association, Manchester Center, Vermont, 1982.

[4] P. Chenna, T. Rao, et al (2018). HFSS software is used to design a twin inverted L microstrip antenna for sustainable systems. International Journal of Computer-Aided Engineering Technology (IJCAET).

[5] J. Quintero, an IEEE student member, and R. Asprilla, an IEEE member Proceedings of the 2020 IEEE Thirty-Fifth Central American and Panama Convention: Towards an Efficient Voice-Based Chatbot (concapan xxxv).

[6] Rao, P.T., and Kumar, P.P. (2015,January). Twin microstrip patch antenna in the form of a staircase. 2015 saw the ICPC (International Conference on Pervasive Computing) (pp. 1-5). The Technical Writer's Handbook by IEEE M. Young. University Science, Mill Valley, California, 1989.

[7] Rao, P. T., and Kumar, P. P. (2021). dual dual folded inverted-L antenna with a frequency-tunable circular polarisation and varactor-loaded split-ring resonator constructions. Global Journal of Communication Systems, e4715.

[8] Ly Pichponreay and Chi-Hwan Choi created the Chungbuk National University Smart Responding Chatbot based on OCR and Overgenerating Transformations and Rating.

[9] Trinatha Rao, P., and Praveen Kumar, P. C. (2020). Using HFSS, a double folded inverted-L antenna with a rectangular ground plane is designed to be reconfigurable. IETNetworks,9(5),229-234.

[10] Automated Question Answering System Utilizing Ontology and Semantic Role International Conference on Innovative Mechanisms for Industrial Applications (ICIMIA2017) by S. Jayalakshmi and Dr. Ananthi Sheshasaayee.

[11] Yuriy Dyachenko and Nayden Nenkov Articial Intelligence Technologies For Personnel Learning Management Systems 2019 IEEE 8th Internatignal Conference on Intelligent Systems Volodymyr DahlEast Ukrainian National University Severodonetsk, Ukrainev.