INFOBOT – WEB BASED CHATBOT

A PROJECT REPORT

Submitted by

RATNAKUMAR A	**813819205050**
ABISHEK A S	**813819205004**
HARSHAVARTHAN S R	**813819205020**
SEENI ARIVAZHAKAN	**813819205053**

in partial fulfilment for the award of the degree

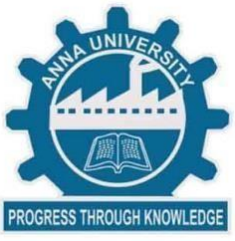
of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

SARANATHAN COLLEGE OF ENGINEERING,TIRUCHIRAPALLI



ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**INFOBOT – WEB BASED CHATBOT"** is the bonafide work of

RATNAKUMAR A	**813819205050**
ABISHEK A S	**813819205004**
HARSHAVARTHAN S R	**813819205020**
SEENI ARIVAZHAKAN	**813819205053**

Who carried out the project work under my supervision

SIGNATURE

Dr.R.Thillaikarasi, M.Tech.,Ph.D.,

HEAD OF THE DEPARTMENT

Professor

Information Technology

Saranathan College of Engineering,

Panjappur,

Trichy – 620012.

SIGNATURE

Mrs.J.Sangeethapriya, M.Tech.,

SUPERVISOR

Assistant Professor

Information Technology

Saranathan College of Engineering,

Panjappur,

Trichy – 620012.

VIVA-VOCE EXAMINATION

INFOBOT – WEB BASED CHATBOT

Submitted by

RATNAKUMAR A	**813819205050**
ABISHEK A S	**813819205004**
HARSHAVARTHAN S R	**813819205020**
SEENI ARIVAZHAKAN	**813819205053**

The viva-voce Examination of this project work done as a part of B. Tech Information Technology was held on _____ .

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We sincerely thank **Shri. S. Ravindran., Secretary, Saranathan College of Engineering** for giving us a platform to realize our project.

We express our sincere thanks to **Dr. D. Valavan Ph.D., Principal, Saranathan College of Engineering** for giving us an opportunity and immense support for the successful completion of the project.

We are obliged to **Dr. R. Thillaikarasi, M.Tech., Ph.D., Professor & Head of Department, Information Technology, Saranathan College of Engineering** for her valuable suggestion and encouragement to our project. We express our heartfelt thanks to our project coordinator **Mr. P. Anand, M.E., Assistant Professor, Department of Information Technology, Saranathan College of Engineering,** and guide **Mrs. J. Sangeethapriya, M.Tech., Assistant Professor, Department of Information Technology, Saranathan College of Engineering,** for us with her valuable ideas.

We would like to thank all our department faculty members and technical assistant for their support and help rendered by them in completion of this project. We are also thankful to the department for providing Wi-Fi connection throughout our project period.

We would like to thank our parents for their constant encouragement and support without which this project would not be possible. Last but not the least we would like to thank our friends who have been instrumental in providing idea and material for the construction of our project.

Above all, we thank the God almighty for his bountiful blessings.

ABSTRACT

A web-based chatbot is an artificial intelligence program that understands and conversely responds to user inquiries using natural language processing and machine learning methods. Several industries, including e-commerce, customer service, healthcare, and education, have widely embraced this technology. Businesses may instantly assist customers, automate tedious operations, and increase productivity by utilizing chatbots. This abstract tries to give a general overview of web-based chatbots, including their features, benefits, and drawbacks. It also emphasizes how chatbots might affect how customer service is provided in the future and how important they are to achieving digital transformation. Web-based chatbots are often linked into websites or messaging services and are available around-the-clock, giving users a quick and easy method to communicate with companies or services. These chatbots are made to seem like actual people, giving consumers a tailored experience and responding to their demands instantly.

Web-based chatbots have the benefit of being able to handle several enquiries at once, which makes them the perfect choice for companies that deal with a lot of client contacts. Additionally, they can lighten the load on customer support teams, freeing up resources to concentrate on more complicated problems. However, creating and implementing web-based chatbots also has its share of difficulties. The response time of our chatbot to the user queries is 0.2 milli seconds. And the loss time for our chatbot to the user queries is 0.03 milli seconds.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

HTML	-	Hyper Text Markup Language
CSS	-	Cascading Style Sheets
JSON	-	JavaScript Object Notation
VPS	-	Virtual Private Sector
GRU	-	Gated Recurrent Units
NLU	-	Natural Language Understanding
ASR	-	Automatic Speech Recognition
NLP	-	Natural Language Processing
API	-	Application Programming Interface
NLTK	-	Natural Language Tool Kit
CRM	-	Customer Relationship Management

CHAPTER 1
INTRODUCTION

1.1 Introduction

A chatbot or chatterbot is a software application used to conduct an online chat conversation via text or text-to-speech, in lieu of providing direct contact with a live human agent. A chatbot is a type of software that can help customers by automating conversations and interact with them through messaging platforms. Chatbots or Chat Robots are the technology of the decade in which a machine having human conversation capabilities interacts with people[2].

A chatbot is a piece of software that interacts with a human through written language. It is often embedded in web pages or other digital applications to answer customer inquiries without the need for human agents, thus providing affordable effortless customer service.

A chatbot is a computer program that simulates human communication. Chatbot are frequently used in a wide variety of online situations, from customer service to sales. Chatbots are not just elements of virtual assistants, but are used by organizations and governments on websites, in applications, and instant messaging platforms to promote products, ideas or services[3]. Chatbot applications are being increasingly adopted as a direct communication channel between companies and end-users[5].

Recently, the demand for chatting with dialogue systems has been increasing. However, one of the significant problems of chat dialog systems is that they cannot respond appropriately to users' spontaneous speech since systems often respond inconsistently to past topics due to their weak dialog history management. To avoid this problem, the system tends to generate conservative, simple and short speech for consistent conversation, which leads to a boring dialogue[10].

1.2 Technologies Used

1.2.1 HTML

HTML stands for Hyper Text Markup Language. It is used to design web pages using the markup language.HTML is the combination of Hyper Text and Markup Language. Hypertext defines the link between the web pages and markup languages defines the text document within the tag that define the structure of webpages. It can be assisted by technologies such as CSS and JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages.

1.2.2 CSS

CSS (Cascading Style Sheets) is a style sheet language for describing the appearance of a document written in a markup language such as HTML. Along with HTML and JavaScript, CSS is a key component of the World Wide Web. CSS is a style sheet that allows you to separate presentation from content, including layout, colors, and fonts. This separation can increase content accessibility, provide more flexibility and control in the specification of presentation features, allow many web pages to share formatting by declaring appropriate CSS in a separate CSS file, and decrease structural content complexity and duplication.

1.2.3 Java Script

JavaScript, abbreviated as JS, is an interpreted high-level programming language. It's a dynamic, weakly typed, prototype-based, and multi-paradigm language, among other things. JavaScript is one of the three essential technologies of the World Wide Web, alongside HTML and CSS. JavaScript allows users to interact with web pages and is thus an important component of online applications. It's used by the vast majority of websites, and it's supported by a dedicated JavaScript engine in all major web browsers.

1.2.4 JSON

JSON (JavaScript Object Notation) is a simple data-transfer format. Humans can read and write with ease. Machines can easily parse and generate it. It is based on a portion of ECMA-262, Third Edition, December 1999's JavaScript Programming Language Standard. Nonetheless, JSON uses patterns that are well-known to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. JSON is a text format that is fully independent of programming language. These qualities make JSON an ideal data-interchange language.

1.2.5 Python

Python is a high-level, all-purpose programming language. Code readability is prioritized in its design philosophy, which makes heavy use of indentation. Python uses garbage collection and has dynamic typing. It supports a variety of programming paradigms, including procedural, object-oriented, and functional programming as well as structured programming (especially this). Due to its extensive standard library, it is frequently referred to as a "batteries included" language. Python was created by Guido van Rossum in the late 1980s to replace the ABC programming language, and it was originally made available as Python 0.9.0 in 1991. In 2000, Python 2.0 was made available. The 2008 release of Python 3.0 was a significant update that was only partially backwards compatible with previous iterations. The final Python 2 release was Python 2.7.18, which was made available in 2020.

1.2.6 Natural Language Processing

Natural Language Processing(NLP) technology is used to enable chatbots to understand and interpret human language. It helps chatbots to identify the intent behind a user's message and respond appropriately. Some popular NLP tools for chatbot development include Dialogflow, Wit.ai, and Microsoft LUIS.

1.2.7 Machine Learning

Machine learning is used to improve the accuracy of chatbot responses by analyzing data and learning from it. It allows chatbots to learn from user interactions and improve their responses over time. Some popular machine learning frameworks for chatbot development include TensorFlow, Keras, and PyTorch.

1.3 Benefits of chatbot

- **Can hold multiple conversations at once:** Chatbot can communicate with thousands of buyers at the same time. This boosts business efficiency and cuts down wait times.
- **Cost-effective:** A chatbot is a faster and cheaper one-time investment than creating a dedicated, cross-platform app or hiring additional employees.
- **Improves customer engagement:** Most companies already engage their customers through social media. Chatbot can make this engagement more interactive. Buyers rarely talk to the people within businesses, so chatbot open a communication channel where customers can engage without the stress of interacting with another person. Chatbots are very efficient in providing solution to customers queries and are available 24 hours to give solution to customer's complaints[8].

1.4 Challenges

- **Security**: Users must have enough trust in the chatbot to disclose personal information with it. As a result, businesses must verify that their chatbot only ask for data that is relevant to them and that the data is safely transmitted over the internet. Chatbot should be designed to be secure and able to prevent hackers from gaining access to chat interfaces.
- **Varieties in how people type their messages:** This can lead to misunderstood intentions. Chatbot must handle both long and short sentences, as well as chat bubbles with lengthy content versus multiple short submissions.
- **User Satisfaction:** Users always want the best experiences but are rarely satisfied. They always want the chatbot to be better than its current version. This means organizations

employing chatbot must consistently update and improve them to ensure users feel like they're talking to a reliable, smart source.

1.5 Roles

- **Chatbot as digital assistant:** With the role of the digital assistant, the chatbot helps the user with doing certain tasks. Mostly, these are repeatable tasks, which takes the user a lot of time to do, but the chatbot does this automatically.
- **Chatbot as information provider:** With the role of information provider, a user asks question to a chatbot and it providers answers to these questions. It is mainly used for obtaining help and information. This kind of chatbot are really interesting for businesses with customer service. Customers now automatically can get answers to their question whenever they want, without any waiting times.
- **General chatbot:** With the role of a general chatbot, a user uses the chatbot to conduct a conversation for entertainment or emotional support. Users think of chatbot as fun and entertaining, and they usually use chatbot whenever they are bored.

1.6 Applications of chatbot

- **Retail and E-Commerce:** Chatbot can field questions about the price and availability of your products. With rich media interactions such as pictures and carousels, your chatbot can also give your customers a visual idea of what your products look like, along with suitable options or alternatives.
- **Travel and Hospitality:** Be it hotel reservations or restaurant reservations, chatbot can take care of the entire booking process for your customers and send confirmations within seconds.
- **Healthcare:** Chatbot have proven to be very helpful in the healthcare industry, especially in recent times when hospitals and other institutions are required to minimize queues and gatherings.

CHAPTER 2

LITERATURE SURVEY

2.1 Title: Implementation chatbot Whatsapp using Python Programming for Broadcast and Reply Message Automatically.

Author: Achmad Ramaditiya, Suci Rahmatia, Aris Munawar,

Octarina Nur Samijayani

Year: 2021

Abstract

The use of VPS (Virtual Private Server) in Indonesia is still very expensive[1]. The Chatbot application system is very important in the marketing field, especially for disseminating information directly and acceptable to many users at a time. This paper focused on using the WhatsApp application for the Chatbot system. This Chatbot system uses the Python programming language. The message Chatbot flow system will be sent first to the user. Then the Python program will read the incoming message to enter Chatbot. If the incoming message matches the existing conditions, the chatbot will send the information according to the condition. But if it doesn't match, Chatbot will continue to repeat the process of reading incoming messages. The Chatbot system is designed to run successfully on 15 contacts at a time. Chatbot server connection speed affects the speed of sending messages and checking every incoming message. Chatbot simulation program cannot read messages that enter the server if the message contains stickers, emojis and gifs. This is because Python program cannot read the message. This research can still be developed by adding a random message feature.

Advantages

This Chatbot program is designed to be able to read messages that are not sensitive, so that if have a similar message, the message will still be read and entered into the Chatbot system.

Limitations

This research can still be developed by adding a random message feature. So, the server doesn't need to save the contact number first to spread the messages. This research can also use a system that can read all messages and send them back directly without having to enter the contact's name.

2.2 Title: A Deep Learning based Chatbot for Cultural Heritage.

Author: Giancarlo Sperli

Year: 2020

Abstract

In this paper we propose an entertainment Chatbot based on the sequence to sequence model according to the Encoder - Decoder framework based on GRU cells for supporting user's cultural heritage path[6].

A preliminary evaluation about the efficiency of the proposed approach has been performed asking to 10 users to interact with the Chatbot on cultural items of Campania Region.

Advantages

In this paper the proposed Chatbot is based on deep learning techniques for supporting user's cultural heritage path. Moreover, the proposed Chatbot is able to respond in both Italian and English languages.

Limitations

Future works will be devoted to extend the evaluation using a larger number of users and improving the Deep Learning approach using an online learning strategy.

2.3 Title: An AI-Based Medical Chatbot Model for Infectious Disease Prediction

Author: Sanjay Chakraborty, Hrithik Paul, Sayani Ghatak, Saroj Kumar Pandey, Ankit Kumar, Kamred Udham Singh, Mohd Asif Shah

Year: 2022

Abstract

The purpose of this paper is to show concisely how we can promote chat bot in the medical sector and cure infectious diseases[4]. We can create awareness through the users and the users can get proper medical solutions to prevent disease. We created a preliminary training model and a study report to improve human interaction in databases in 2021. Through natural language processing, we describe the human behaviours and characteristics of the chat bot. In this paper, we propose an AI Chat bot interaction and prediction model using a deep feed forward multi layer perceptron. Our analysis discovered a gap in knowledge about theoretical guidelines and practical recommendations for creating AI chat bot for lifestyle improvement programs. A brief comparison of our proposed model concerning the time complexity and accuracy of testing is also discussed in this paper. In our work, the loss is a minimum of 0.1232 and the highest accuracy is 94.32%. We believe that our findings will help researchers get a better understanding of the layout and applications of these revolutionary technologies, which will be required for continuous improvement in medical chat bot functionality and will be useful in avoiding COVID-19.

Advantages

This bot offers medical-related information like doctor's contact details, address of nearby hospitals, contact details for getting an oxygen cylinder, about the disease its symptoms, its prevalence, diagnosis, and its treatment procedures.

Limitations

This medical chatbot has wide future opportunities. People in remote areas can also receive benefits from this. Here we use 'TensorFlow', which helps to build the NLP for chatbot and utilizes deep neural network architecture.

2.4 Title: E-Commerce Assistance with a Smart Chatbot using Artificial Intelligence

Author: Manik Rakhra, Gurram Gopinadh, Narasimha Sai Addepalli, Gurasis Singh, Shaik Aliraja, Vennapusa Siva Ganeshwar Reddy,

Muthumula Navaneeswar Reddy

Year: 2021

Abstract

In this present research we use design science analysis to express chatbot design awareness in higher education[7]. For previous research into the design process, we carried out a literature review. Furthermore, we studied the content of student emails and forum articles from four instances of a fundamental Java programming course. We introduce a conceptual architecture for chatbot in higher education from literature and evidence and show how this is applied. We conclude with a debate with tentative design recommendations and a plan for further research. Website and social networking outlets have become more and more popular places for people to voice their opinions on different issues, in particular their frustrations with brands and corporations.

Advantages

This paper is intended to introduce a chatbot based on the Ecommerce engine which seeks to improve the user's engagement with E-Commerce engine.

Limitations

Grammar-based data parsing is needed for efficient Chatbot applications in order for the user to comprehend the intended sentence by defining phrases that are suited to the complexities of the grammar used.

2.5 Title: Development of An e-commerce Sales Chatbot

Author: Mohammad Monirujjaman Khan

Year: 2020

Abstract:

This paper presents the development of an ecommerce sales chatbot in order to provide customer support and increase sales. The system uses machine learning for natural language understanding[9]. It is developed on an modular chatbot framework. Second, a microservice to classify input text and extract entities. Finally, a framework which routes user request to specific controller for processing and serves the response.

Advantages:

The system has a visual platform to train the NLU-Natural Language Understanding Engine. It is a single page application which serves all the frontend login and the views as a single file whenever user browses the site, it reduces server side call, improves performance and provides a smooth experience.

Limitations:

Artificial Neural Network can be used to improve the accuracy of the NLU Engine. Also, a semi supervised learning system can be implemented in order to increase the dataset

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATIONS

3.1 Introduction

3.1.1 Purpose

The main purpose of this INFOBOT is to provide solutions to all the queries of the customer at any time. This Chatbot system is available 24 x 7. In this Chatbot the users can able to change their language according to their needs. This Chatbot is used to find information about the Institution, consult their courses availability and make reservations or bookings of admission. The Institution can get real insight into what they're students prefer, and therefore provide improved answers to their users.

3.1.2 Scope

The main scope of the project is that the Chatbot as a key tool to strengthen the relationships with the clients both internally and externally. If a particular Institution receives many requests, it is not necessary to increase the template or team capacity for receiving queries. A well-built chatbot allows the brand to face all the questions in a simultaneous manner.

3.2 Overall Description

3.2.1 System Environment

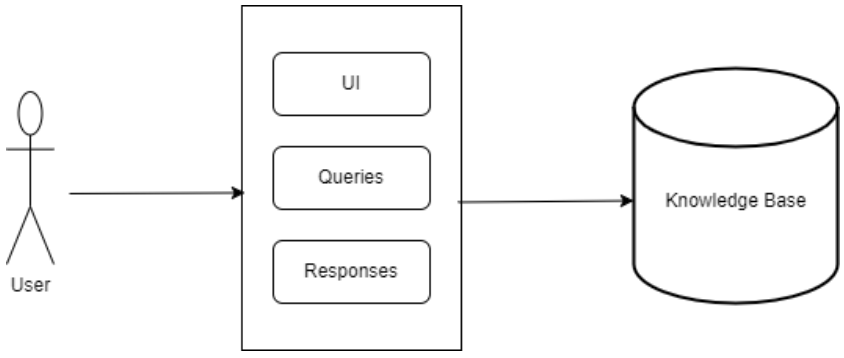


Fig.3.1 System Environment

Description

This diagram illustrates the system environment of the Infobot. Here the basic flow is the user access the chatbot which resides in the institutional website and they can post their queries in the chatbot's user interface and they could get their appropriate answer.

3.2.2 User Module

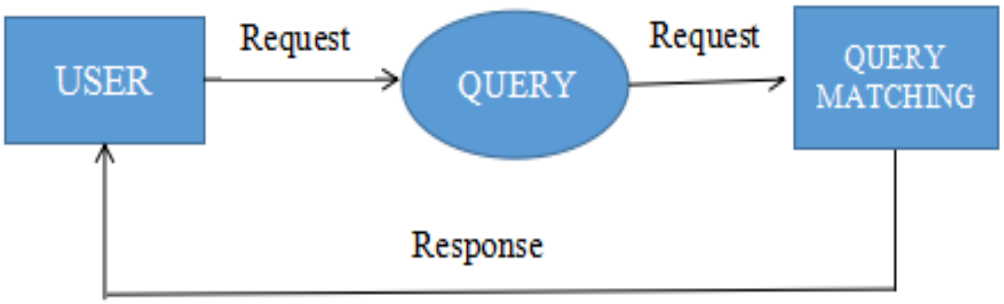


Fig.3.2 User Module

Description

The user enters the query in the chatbot and the chatbot produces the most accurate response based on the trained datasets. Overall, the user module is a crucial part of any chatbot, as it enables the chatbot to interact with users in a natural and intuitive way.

3.2.3 Admin Module

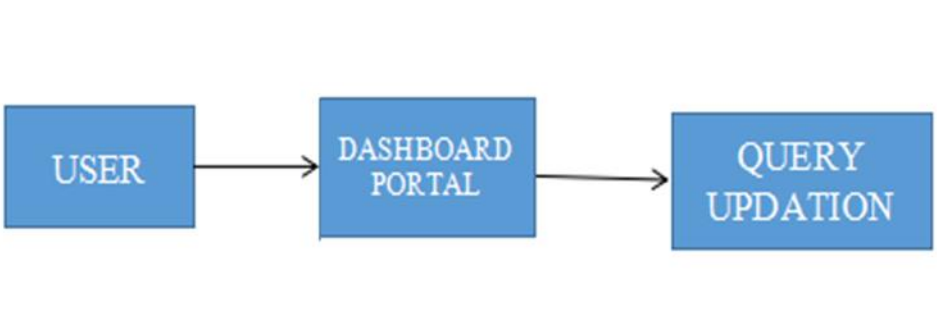


Fig.3.3 Admin Module

Description

The admin module should provide functionality to manage the chatbot's data, such as storing and retrieving user information, conversation history, and analytics data. This can help the chatbot administrator gain insights into how users are interacting with the chatbot and identify areas for improvement. This component allows the chatbot administrator to configure the chatbot's behaviour, such as setting default responses, defining conversation flows, or customizing the chatbot's appearance. Overall, the admin module is a critical component of any chatbot, as it provides the chatbot administrator with the tools they need to manage and optimize the chatbot's behaviour. The admin module in a chatbot is responsible for managing the chatbot's behavior and configurations. It is the interface through which chatbot developers and administrators can make changes to the chatbot's settings and access data and analytics about the chatbot's performance.

3.2.4 Speech Recognition Module

Fig.3.4 Speech Recognition Module

Description

The speech recognition module in a chatbot is responsible for converting spoken language into text that the chatbot can process. This is also known as automatic speech recognition (ASR) or speech-to-text conversion. The speech recognition module typically includes several sub-components that work together to provide accurate and reliable speech recognition. Overall, the speech recognition module is a crucial part of any chatbot that supports voice input, as it enables the chatbot to understand spoken language and interact with users in a more natural and intuitive way.

3.2.5 Language Translation Module

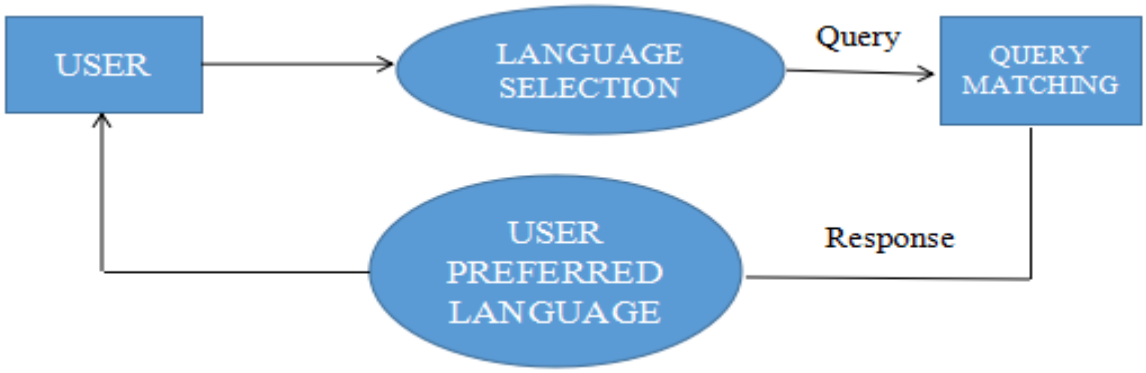


Fig.3.5 Language Translation Module

Description

The user can select the language as per their preference and the enters their query. The chatbot produces the response based on the user preferred language. The multi-language module in a chatbot allows it to support multiple languages, enabling it to interact with users who speak different languages. This component is responsible for identifying the language of the user's input. The contents of the web page can be translated according to the need of the user. A content translation module in a chatbot is responsible for translating content such as responses, prompts, and menus into the language that the user speaks.

3.3 Use Case Diagram

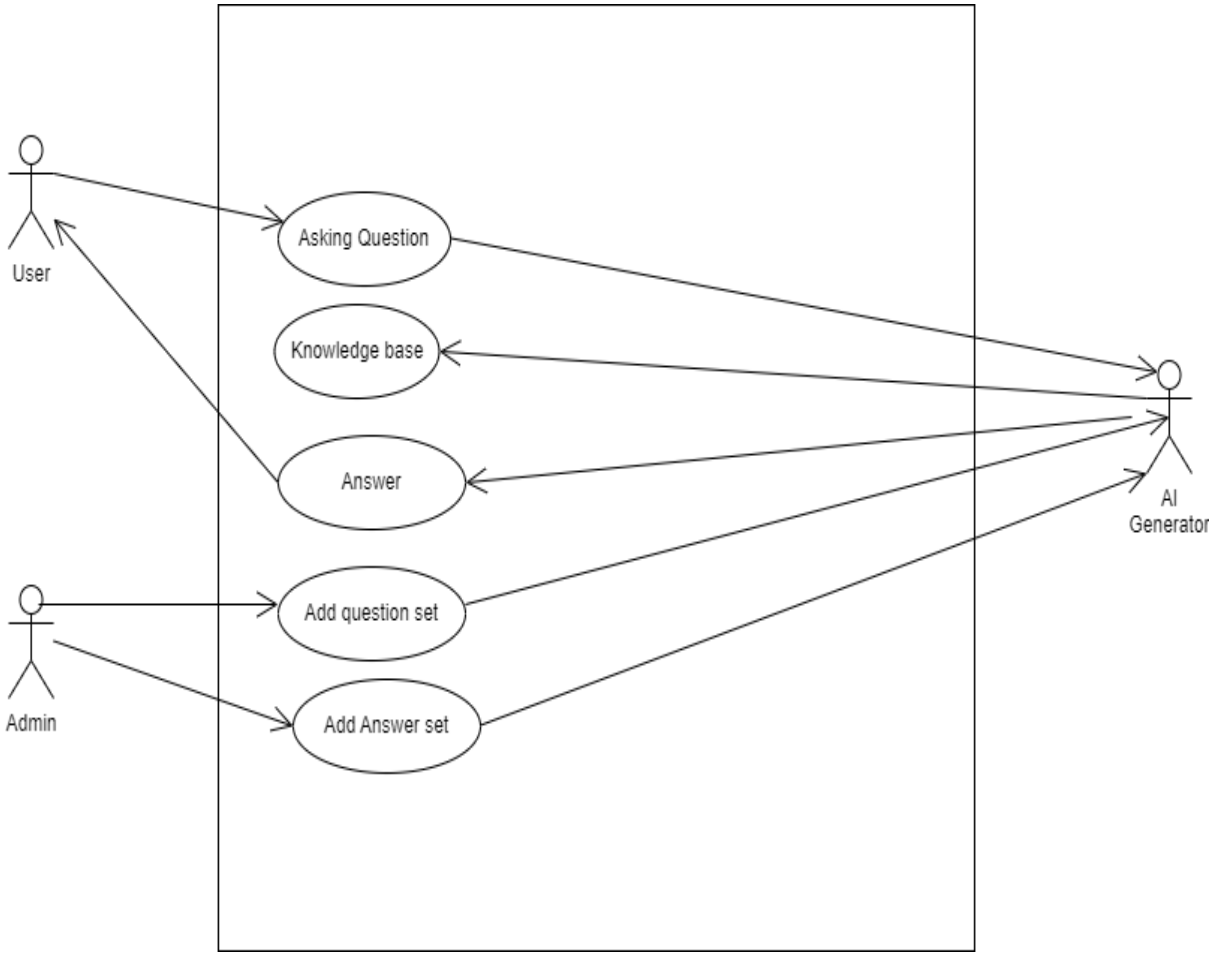


Fig.3.6 Use case Diagram

Description

In this Use case the actors are User, Administrator and AI generator. The User login into the institutional website and they could find the chat bot and there they can ask their queries. Now the queries which was asked by the user was fetch by the AI generator and it checks with its knowledge base. The Knowledge base consists of the predefined queries and its appropriate responses posted by the admin. Now the queries will be matched with the user's query and if there is a match means it will be fetched and send to the AI generator. And the AI generator will now display the required response for the user's query. And Now from the admin side, the admin has the access to post and delete the queries and responses. The Admin can update the queries through the website itself so there is no need for a developer.

3.4 Software Requirements Specification

3.4.1 Requirements

The categories of functional, non-functional, user, and system requirements, among others, are frequently used to group needs in an SRS. Each criterion must be unambiguously stated, specific, and quantifiable.

3.4.1.1 Functional Requirements

Functional specifications outline the precise duties or obligations that the software system must fulfil.

3.4.1.2 Non-Functional Requirements

Non-functional requirements outline the system's quality characteristics, including its performance, dependability, and security.

3.4.2 Functional Requirements

Functional Requirement defines a function of software system and how system must behave when presented with specific inputs or conditions. These may include

calculations, data, manipulation and processing and other specific functionality. In this system following are the functional requirement training retina dataset must be loaded.

3.4.2.1 User-Asking Questions

Table.3.1 Asking Questions

Use case name	Asking Questions
Priority	Essential
Pre-condition	NA
Basic Path	User can ask questions using UI.
Post-condition	Online

3.4.2.2 User-Getting Answers

Table.3.2 Getting Answers

Use case name	Getting Answers
Priority	Essential
Pre-condition	Getting a question
Basic Path	User receives the answer through the Knowledge base.
Post-condition	Online

3.4.2.3 Admin-Add question set

Table.3.3 Adding question set

Use case name	Add question set
Priority	Essential
Pre-condition	New Questions that are posted
Basic Path	Adding questions through JSON
Post-condition	Online

3.4.2.4 Admin-Add Answer set

Table.3.4 Adding Answer set

Use case name	Add answer set
Priority	Essential
Pre-condition	Updating new queries
Basic Path	Adding answers through JSON
Post-condition	Online

3.4.3 Non-Functional Requirements

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviours. They may relate to emergent system properties. Non-functional requirements for this system are specified as follows:

- Responsiveness of the system needs to be appropriate since timely retrieval of sensitive health data is essential.
- The software utilized should be portable so that medical institutions can easily expand to their inter-connected hospitals, spread across locations.
- Privacy of sensitive data should always be maintained and must not be misused in any manner.
- Researchers requesting for data must be from authorized sources. Proof of consent is a responsibility that is borne by the medical institution where the health data is generated.

3.5 Overall Perspective

3.5.1 Product Perspective

Chatbot has found its way into a variety of customer-facing businesses, including e-commerce, food delivery, travel, healthcare, and banking. It’s just a matter of perspective

whether it's a feature or a product. In other words, it can be an item, tool, environment, or even a service that buyers would pay for. A feature, on the other hand, is a component of a product.

3.5.2 Software Interface

a) In Server side with the help of localhost the request (queries) will be processed and forwarded accordingly.

b) Client side An OS which is capable of running a modern web browser which supports JavaScript and HTML5.

3.6 Computer Requirements

3.6.1 Hardware Requirements

Table.3.5 Hardware Requirements

Processor	Intel processor 2.6.0 GHZ
RAM	1GB
Hard Disk	160 GB
Compact Disk	650 MB
Keyboard	Standard Keyboard
Monitor	15'' Color Monitor

3.6.2 Software Requirements

Table.3.6 Software Requirements

Operating System	Windows OS
Frontend	HTML, CSS, Python
Code Editor	Visual Studio Code
Application	Web Application

CHAPTER 4

SYSTEM DESIGN

4.1 Architectural Design

System architecture refers to the overall design and structure of a system, which includes hardware, software, network, and other components. It is a conceptual model that describes the relationships and interactions between different components of a system and how they work together to achieve a common goal. System architecture can be classified into different categories based on the type of system being developed.

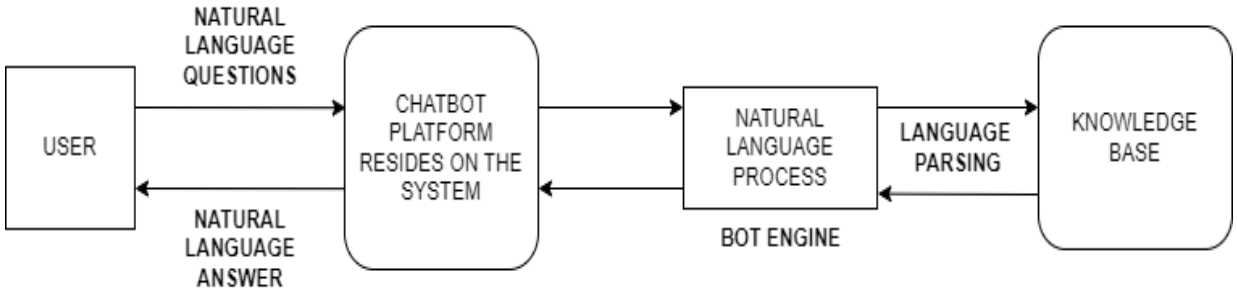


Fig.4.1 Architecture Diagram

The user can able to get a response for their queries using a web based chatbot. The architecture explains that the user logged in to a website and there appears the chatbot where they can post their queries, those queries will be the Natural language questions and taken as a input to the chatbot that resides on the website and those questions where sent to the Bot Engine.

This Bot Engine refers to the backend of the chatbot. The queries will be processed and it will be matched with the knowledge base which is a JSON file consists of the queries and its appropriate responses. In the JSON file the queries and responses are stored in the key value pairs.

If the user's query matches with any key then its appropriate value will be given as output and those responses will be sent to the bot engine and it will be displayed in the chatbot interface as the Natural Language Responses.

4.2 Modules Description

- User Module.
- Admin Module.
- Speech Recognition.
- Language Translation.

4.2.1 User Module

The user module of a chatbot is in charge of managing communications between the chatbot and the user. The user module often acts as the chatbot's interface and is where users enter information so that it can comprehend and reply to their queries. A natural language processing (NLP) engine, a dialogue manager, and a response generator are a few possible parts of the user module. The NLP engine analyses user input and pulls out pertinent data, including the user's purpose, entities referenced, and any context that would be required to comprehend the user's request. The relevant answer is then created by the response generator and delivered to the user after being determined by the dialogue manager using this data. Overall, the user module is critical to the success of a chatbot, as it determines how the chatbot interacts with its users and how effectively it can understand and respond to user requests. Some of the Functions that are required in a User module that make a chatbot effective are:

- **Input Processing:** The user module is responsible for processing user input, which can be in various forms such as text, voice, or images. The input processing component in the user module is responsible for cleaning and normalizing the user input, removing irrelevant information and preparing it for further processing by the NLP engine.

- **Natural Language Processing:** The NLP engine in the user module analyzes the user input and extracts useful information such as the user's intent, entities, and context. The NLP engine uses various techniques such as tokenization, part-of-speech tagging, named entity recognition, and sentiment analysis to understand the user's request.
- **Response Generation:** The response generation component in the user module takes the output from the dialogue manager and generates a response to the user. The response can be in various formats such as text, voice, or visual media.
- **Error Handling:** The user module is also responsible for handling errors and unexpected user input. If the chatbot is unable to understand the user's request, the user module can provide a suitable error message or redirect the user to a help or support section.

4.2.2 Admin Module

The admin module in a chatbot is responsible for managing the chatbot's behavior and configurations. It is the interface through which chatbot developers and administrators can make changes to the chatbot's settings and access data and analytics about the chatbot's performance. Some of the key features that an Admin module should have are:

- **Chatbot Configuration:** The admin module allows chatbot administrators to configure the chatbot's settings, such as the chatbot's welcome message, default responses, and user authentication options.
- **Integration Management:** The admin module allows administrators to manage integrations with other systems or services. This includes configuring API connections and managing data exchange between the chatbot and other systems.
- **Multi-User Collaboration:** The admin module allows multiple users to collaborate on the management of the chatbot. This can be useful for large organizations with

multiple teams or departments responsible for different aspects of the chatbot's development and management.

- **Continuous Improvement:** The admin module provides insights into the chatbot's performance and user behavior. This information can be used to continuously improve the chatbot's features and functionality, and to optimize the user experience over time.

4.2.3 Speech-Recognition

Speech recognition, or speech-to-text, is the ability of a machine or program to identify words spoken aloud and convert them into readable text. Rudimentary speech recognition software has a limited vocabulary and may only identify words and phrases when spoken clearly. More sophisticated software can handle natural speech, different accents and various languages. Many modern devices and text-focused programs have speech recognition functions in them to allow for easier or hands-free use of a device. A software program turns the sound a microphone records into written language that computers and humans can understand, following these four steps:

- Analyze the audio.
- Break it into parts.
- Digitize it into a computer-readable format.
- use an algorithm to match it to the most suitable text representation.

Speech recognition software must adapt to the highly variable and context-specific nature of human speech. The software algorithms that process and organize audio into text are trained on different speech patterns, speaking styles, languages, dialects, accents and phrasings. The software also separates spoken audio from background noise that often accompanies the signal.

To meet these requirements, speech recognition systems use two types of models:

- **Acoustic models:** These represent the relationship between linguistic units of speech and audio signals.

- **Language models:** Here, sounds are matched with word sequences to distinguish between words that sound similar.

It's important to note that speech recognition is not always perfect and can sometimes produce errors. As such, it's important for the chatbot to be able to handle errors gracefully and to provide users with options to correct any errors that may occur during speech recognition.

4.2.4 Language Translation

The Google Translate API is a cloud-based service provided by Google that enables developers to integrate automatic language translation into their applications. The API can be used to translate text from one language to another, detect the language of input text, and perform batch translations of large volumes of text. The API uses machine learning algorithms to perform translations, and it supports over 100 languages, including some rare or less widely spoken languages. To use the Google Translate API, developers must first create a project in the Google Cloud Platform Console and enable the Google Translate API for that project. The API can be accessed through a RESTful interface, which enables developers to integrate it into a wide range of applications and programming languages. There is a cost associated with using the Google Translate API, with pricing based on the number of characters translated per month. Developers can monitor their usage and costs through the Google Cloud Platform Console and set usage limits to control costs and prevent unexpected charges. Some of the key features of the google translate API are:

- **Translation customization:** Developers can customize translations for specific domains or industries using the Translation API's AutoML Translation feature.
- **Language detection:** The Google Translate API also includes language detection functionality, which can automatically detect the language of a given input text.
- **Availability:** The Google Translate API is generally available for use, though there may be occasional service interruptions or maintenance windows

CHAPTER 5

IMPLEMENTATION

5.1 Software Description

Python is a high-level, interpreted programming language that is widely used in various domains such as web development, scientific computing, data analysis, artificial intelligence, machine learning, and more. One of the key features of Python is its easy-to-learn syntax, which makes it accessible to both novice and experienced programmers. It has a large standard library that provides a wide range of modules for tasks such as file I/O, networking, regular expressions, and more. This allows for rapid development and testing, as well as easier debugging and maintenance of code. Python is used for a variety of applications, including web development frameworks such as Django and Flask, scientific computing libraries such as NumPy and Pandas, and machine learning libraries such as TensorFlow and PyTorch. We have imported the packages using PIP and the framework that is used in our project is Flask.

- **PIP:**

  It is a package-management system written in Python and is used to install and manage software packages and also its dependencies during deployment. Pip connects to an online repository of public packages, called the Python Package Index.

- **FLASK :**

  Flask is a great choice for building chatbots that can be accessed via a web interface. To create a Flask application by importing the Flask library and creating an instance of the Flask class.

  // from flask import Flask

  // app = Flask(__name__)

- **NLTK :**

  It provides easy-to-use with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

// import nltk

// from nltk.stem.porter import PorterStemmer

❖ **TORCH :**

PyTorch is a powerful tool for building chatbots with machine learning capabilities, and can enable chatbots to learn and adapt to user input over time.

// import torch

// import torch.nn as nn

// from torch.utils.data import Dataset, DataLoader

Once your chatbot has been trained, you can use it to generate responses to user input. This involves passing the user input through the interface, and converting the output back into human-readable text to generate a response for the user.

❖ **NUMPY:**

NumPy can be a useful tool in building chatbots with machine learning capabilities, and can help to improve the efficiency and performance of the chatbot's architecture.

// import numpy as np

initialize bag with 0 for each word

// bag = np.zeros(len(words), dtype=np.float32)

❖ **PANDAS**

Pandas can be used in web-based chatbots for data storage, cleaning, analysis, and integration with other libraries commonly used in chatbots, such as Flask and NLTK.

// import pandas as pd

// df = df1.replace(np.nan, ' ', inplace=True)

❖ **ODFPY**

odfpy is a Python library that can be used to manipulate OpenDocument Format (ODF) files, which is a popular file format for office applications.

// df = pd.read_excel(fileName, engine='odf')

❖ **PUNKT**

Punkt can be a valuable tool in chatbots that need to process and generate natural language text by providing a pre-trained sentence tokenizer that can help to segment and normalize text. Punkt is a module in the NLTK library that provides a pre-trained sentence tokenizer for natural language processing. Sentence tokenization is the process of splitting a text into individual sentences.

```python
import nltk
nltk.download('punkt')
```

Fig.5.1 PUNKT implementation

5.2 Module Implementation

- **User Module**

When a user first interacts with the chatbot and provide relevant information. This information is then stored in the Knowledge base. As the user interacts more with the chatbot, the user module can use this information to tailor the chatbot's responses. This can help the chatbot to identify areas where it can improve and to provide more personalized and engaging interactions with the user. Overall, the user module is an important component of a chatbot that helps to personalize the chatbot's interactions with users and to improve the overall user experience.

```
$('#start-record-btn').on('click', function (e) {
    if (noteContent.length) {
      noteContent += ' ';
    }
    recognition.start();
  });
  // Sync the text inside the text area with the noteContent variable.
  noteTextarea.on('input', function () {
    noteContent = $(this).val();
  })
```

- **Speech Recognition**

To implement speech recognition in a chatbot, you need to choose a speech recognition library, integrate it into your chatbot, train the model, implement error handling, integrate with the chatbot's NLP module, and provide feedback to the user. This provides an additional layer of convenience and accessibility for users, allowing them to interact with the chatbot using voice commands.

```
  // If false, the recording will stop after a few seconds of silence.
  // When true, the silence period is longer (about 15 seconds),
  // allowing us to keep recording even when the user pauses.
  recognition.continuous = false;
  // This block is called every time the Speech APi captures a line.
  recognition.onresult = function (event) {
    // event is a SpeechRecognitionEvent object.
    // It holds all the lines we have captured so far.
    // We only need the current one.
    var current = event.resultIndex;
```

```
    // console.log(current);
    // Get a transcript of what was said.
    var transcript = event.results[current][0].transcript;
    // Add the current transcript to the contents of our Note.
    // There is a weird bug on mobile, where everything is repeated twice.
    // There is no official solution so far so we have to handle an edge case.
    var mobileRepeatBug = (current == 1 && transcript == event.results[0][0].transcript);
    if (!mobileRepeatBug) {
      noteContent = transcript;
      console.log(noteContent);
      const chatMessage = document.querySelector('#chat-textbox');
      console.log('Hello ', chatMessage);
      chatMessage.value = noteContent;
      document.querySelector('#chat-send-button').click();
    }
  };
```

- **Language Translation**

To implement Google Translate API in a chatbot, you need to sign up for a Google Cloud account, create a new project, enable the Google Translate API, obtain an API key, integrate the API into your chatbot, handle errors, and provide feedback to the user. This provides an additional layer of convenience and accessibility for users who speak different languages.

```
<script type="text/javascript">
    function googleTranslateElementInit() {
      new google.translate.TranslateElement({ pageLanguage: 'en' }, 'google_translate_element');
```

```
    }
  </script>
  <script type="text/javascript"
src="//translate.google.com/translate_a/element.js?cb=googleTranslateElementI
nit">
</script>
```

- **Multimedia Support**

To implement multimedia support in a chatbot, you need to choose a compatible multimedia library, integrate it into your chatbot, handle user input and output and provide appropriate feedback to the user. This enhances the user experience by making the conversation more engaging and interactive.

```
{
    "tag": "pictures",
    "patterns": [
        "admission images", "admission pics", "joining details",
        "admission img","admission pic","ad","admit"
    ],
    "responses": [
        "<img src='/static/docs/assets/img/adm.jpg' height='220px'
width='300px'>",
        "<img src='/static/docs/assets/img/ad1.jpg' height='300px'
width='240px'>"

    ],
    "context_set": ""
},
```

CHAPTER 6

TESTING

6.1 Testing Process

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 Software Testing Strategies

Testing involves

- Unit Testing
- Functional Testing
- Acceptance Testing
- Validation Testing

6.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.2.3 Acceptance Testing

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

6.2.4 Validation Testing

At the culmination of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected, and a final series of software tests-validation testing begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by the customer.

6.3 Test Cases

Table 6.1 User Asking Queries

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Asking Questions	Admission Details	Admission Dates & Details	Admission Dates & Details	PASS

Table 6.2 User Getting Responses

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Getting Response	Admission Details	Admission Dates & Details	Admission Dates & Details	PASS

Table 6.3 Admin Updating details

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Posting Query with responses	Updating Admission Date	Admission Date updated	Admission Date updated	PASS

Table 6.4 Language Translation

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Translating one language to another	Switch from English to Tamil	Language changed to Tamil	Language changed to Tamil	PASS

Table 6.5 Voice Recognition

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	User Voice Recognition	Querying with voice	Voice translated to text and response can be heard	Voice translated to text and response can be heard	PASS

Table 6.6 Multimedia Support

S.NO	TEST CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	RESULT
1	Providing related multimedia based on query	Admission Details	Admission Related Brochure	Admission Related Brochure	PASS

CHAPTER 7
CONCLUSION

The INFOBOT is user friendly in nature. The user can access the web page through their browser in where the chatbot is residing. The chatbot will be available 24x7. So the user can use the chatbot at any time. When the user asking the questions, the chatbot fetches the answer from the Knowledge base and provides the answer in the web page. Now the user will get a response from the chatbot as text as well as audio format. The user can able to change the language of the content as per their need. There are around 110+ languages available for user convince based their requirements the user can change the chatbot language. Here the admins has a separate login. Every admin is provided with a unique username and password to login. Admins can able to update the queries and their related answers to the knowledge base. This system doesn't allow to share any offensive words in order to provide good relationship and environment to use the chatbot. The updating of queries is simple and it overcomes the delay in communication. The response time of our chatbot to the user queries is 0.2 milli seconds. And the loss time for our chatbot to the user queries is 0.03 milli seconds.

CHAPTER 8
FUTURE ENHANCEMENT

The future of chatbots is even more exciting. The enhancement that makes the chatbot even more unique as well as well performing are,

Integrating the chatbot with other services like a CRM system, customer support ticketing system, or social media channels can improve the user experience by streamlining workflows and providing more personalized responses. For example, if the chatbot is integrated with a CRM system, it can provide personalized responses based on the user's past interactions with the company.

And by Implementing emotional intelligence techniques like sentiment analysis and emotional recognition can help the chatbot detect and respond to user emotions. This can improve the chatbot's ability to provide empathetic and supportive responses to users, increasing user engagement and satisfaction.

And finally, by Implementing robust security measures to protect user data and prevent unauthorized access to the chatbot's system.

APPENDICES

APPENDIX-1

SOURCE CODE - CHATBOT DEPLOYMENT

base.html

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}" />
<link rel="stylesheet" href="style.css" />
<head>
  <meta charset="UTF-8" />
  <title>Infobot</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
    integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCO
mLASjC" crossorigin="anonymous" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css" />
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.13.0/css/all.min.css" rel="stylesheet" />
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <style>
    .chat-logo {
      fill: darkblue;
    }
  </style>
```

```
</head>
<body style="height: 80vh;">
  <iframe src="http://saranathan.ac.in/" height="850" width="1695"></iframe>
  <div class="container">
    <div class="chatbox">
      <div class="chatbox__support">
        <div class="chatbox__header">
          <div class="chatbox__image--header">
            <img src="https://img.icons8.com/color/48/000000/circled-user-female-skin-type-5--v1.png" alt="image" />
          </div>
          <div class="chatbox__content--header">
            <h4 class="chatbox__heading--header">InfoBot</h4>
            <p class="chatbox__description--header" style="
            margin-right: 20px;
        ">Chat Support</p>
            <div id="google_translate_element"></div>
          </div>
        </div>
        <div class="chatbox__messages">
          <div></div>
        </div>
        <div class="chatbox__footer">
          <input id="chat-textbox" type="text" placeholder="Write a message..." />
          <button id="chat-send-button" class="chatbox__send--footer send__button">Send</button>
```

```
        <button data-voice="0" class="chatbox__send--footer send__button" id="voice-btn">Voice</button>
       </div>
      </div>
      <div class="chatbox__button">
       <button id="chatbox-button">
        <img src="{{ url_for('static', filename='images/chatlogo.png') }}" class="chat-logo" />
       </button>
      </div>
    </div>
  </div>
  <div class="container">
    <!--  id="hide-speech" -->
    <div class="app" style="visibility: hidden; height: 0;">
      <div class="input-single">
       <textarea id="note-textarea" placeholder="Create a new note by typing or using voice recognition."
        rows="6"></textarea>
      </div>
      <button id="start-record-btn" title="Start Recording">Start Recognition</button>
    </div>
  </div>
  <!-- <div style="visibility: hidden;"> -->
  <div style="visibility: hidden;">
    Select Voice: <select id='voiceList'></select>
```

```
    <br><br>
    <input id='txtInput' /> <br><br>
    <button id='btnSpeak'>Speak!</button>
  </div>
  <script>
    var txtInput = document.querySelector('#txtInput');
    var voiceList = document.querySelector('#voiceList');
    var btnSpeak = document.querySelector('#btnSpeak');
    var synth = window.speechSynthesis;
    var voices = [];
    PopulateVoices();
    if (speechSynthesis !== undefined) {
      speechSynthesis.onvoiceschanged = PopulateVoices;
    }
    btnSpeak.addEventListener('click', () => {
      var toSpeak = new SpeechSynthesisUtterance(txtInput.value);
      console.log(txtInput.value);
      voiceBtn.style.background = "#0d6efd"
      var selectedVoiceName = voiceList.selectedOptions[0].getAttribute('data-
name');
      voices.forEach((voice) => {
        if (voice.name === selectedVoiceName) {
          toSpeak.voice = voice;
        }
      });
      synth.speak(toSpeak);
    });
```

```
    function PopulateVoices() {
      voices = synth.getVoices();
      var selectedIndex = voiceList.selectedIndex = 5;
      voiceList.innerHTML = '';
      voices.forEach((voice) => {
        var listItem = document.createElement('option');
        listItem.textContent = voice.name;
        listItem.setAttribute('data-lang', voice.lang);
        listItem.setAttribute('data-name', voice.name);
        voiceList.appendChild(listItem);
      });
      voiceList.selectedIndex = selectedIndex;
    }
  </script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
  <script>
    const voiceBtn = document.querySelector('#voice-btn');
    console.log(voiceBtn);
    const hideBox = document.querySelector('#start-record-btn');
    voiceBtn.addEventListener('click', () => {
      voiceBtn.dataset.voice = 1;
      console.log(voiceBtn.dataset);
      voiceBtn.style.background = 'blue'
      hideBox.click();
    })
    try {
```

```
    var SpeechRecognition = window.SpeechRecognition ||
window.webkitSpeechRecognition;
    var recognition = new SpeechRecognition();
  }
  catch (e) {
    console.error(e);
    $('.no-browser-support').show();
    $('.app').hide();
  }
  var noteTextarea = $('#note-textarea');
  // console.log(noteTextarea);
  var instructions = $('#recording-instructions');
  var notesList = $('ul#notes');
  var noteContent = '';
  // Get all notes from previous sessions and display them.
  var notes = getAllNotes();
  renderNotes(notes);
  /*-----------------------------
      Voice Recognition
  ------------------------------*/

  // If false, the recording will stop after a few seconds of silence.
  // When true, the silence period is longer (about 15 seconds),
  // allowing us to keep recording even when the user pauses.
  recognition.continuous = false;
  // This block is called every time the Speech APi captures a line.
  recognition.onresult = function (event) {
```

```
    // event is a SpeechRecognitionEvent object.
    // It holds all the lines we have captured so far.
    // We only need the current one.
    var current = event.resultIndex;
    // console.log(current);
    // Get a transcript of what was said.
    var transcript = event.results[current][0].transcript;
    // Add the current transcript to the contents of our Note.
    // There is a weird bug on mobile, where everything is repeated twice.
    // There is no official solution so far so we have to handle an edge case.
    var mobileRepeatBug = (current == 1 && transcript ==
event.results[0][0].transcript);
    if (!mobileRepeatBug) {
      noteContent = transcript;
      console.log(noteContent);
      const chatMessage = document.querySelector('#chat-textbox');
      console.log('Hello ', chatMessage);
      chatMessage.value = noteContent;
      document.querySelector('#chat-send-button').click();
    }
  };
  /*-----------------------------
      App buttons and input
  -----------------------------*/
  $('#start-record-btn').on('click', function (e) {
    if (noteContent.length) {
      noteContent += ' ';
```

```
    }
    recognition.start();
  });
  // Sync the text inside the text area with the noteContent variable.
  noteTextarea.on('input', function () {
    noteContent = $(this).val();
  })
  $('#save-note-btn').on('click', function (e) {
    recognition.stop();
    if (!noteContent.length) {
      instructions.text('Could not save empty note. Please add a message to your note.');
    }
    else {
      // Save note to localStorage.
      // The key is the dateTime with seconds, the value is the content of the note.
      saveNote(new Date().toLocaleString(), noteContent);

      // Reset variables and update UI.
      noteContent = '';
      renderNotes(getAllNotes());
      noteTextarea.val('');
      instructions.text('Note saved successfully.');
    }
  })
  notesList.on('click', function (e) {
```

```
  e.preventDefault();
  var target = $(e.target);
  // Listen to the selected note.
  if (target.hasClass('listen-note')) {
    var content = target.closest('.note').find('.content').text();
    readOutLoud(content);
  }
  // Delete note.
  if (target.hasClass('delete-note')) {
    var dateTime = target.siblings('.date').text();
    deleteNote(dateTime);
    target.closest('.note').remove();
  }
});
/*----------------------------
     Speech Synthesis
----------------------------*/
function readOutLoud(message) {
  var speech = new SpeechSynthesisUtterance();
  // Set the text and voice attributes.
  speech.text = message;
  speech.volume = 1;
  speech.rate = 1;
  speech.pitch = 1;
  window.speechSynthesis.speak(speech);
}
```

```
/*----------------------------
     Helper Functions
----------------------------*/
function renderNotes(notes) {
  var html = '';
  if (notes.length) {
    notes.forEach(function (note) {
      html += `<li class="note">
    <p class="header">
      <span class="date">${note.date}</span>
      <a href="#" class="listen-note" title="Listen to Note">Listen to Note</a>
      <a href="#" class="delete-note" title="Delete">Delete</a>
    </p>
    <p class="content">${note.content}</p>
  </li>`;
    });
  }
  else {
    html = '<li><p class="content">You don\'t have any notes yet.</p></li>';
  }
  notesList.html(html);
}

function saveNote(dateTime, content) {
  localStorage.setItem('note-' + dateTime, content);
}
```

```
  function getAllNotes() {
    var notes = [];
    var key;
    for (var i = 0; i < localStorage.length; i++) {
      key = localStorage.key(i);
      console.log(i)
      console.log(key)
      if (key.substring(0, 5) == 'note-') {
        notes.push({
          date: key.replace('note-', ''),
          content: localStorage.getItem(localStorage.key(i))
        });
      }
    }
    console.log(notes)
    return notes;
  }


  function deleteNote(dateTime) {
    localStorage.removeItem('note-' + dateTime);
  }
</script>
<script type="text/javascript">
  function googleTranslateElementInit() {
    new google.translate.TranslateElement({ pageLanguage: 'en' },
'google_translate_element');
```

```
    }
  </script>
  <script type="text/javascript"
src="//translate.google.com/translate_a/element.js?cb=googleTranslateElementInit"></script>
  <script>
    $SCRIPT_ROOT = {{ request.script_root | tojson }};
  </script>
  <script type="text/javascript" src="{{ url_for('static', filename='app.js') }}"></script>
  <script>
    const chatboxButton = document.querySelector("#chatbox-button");
    const chatMessage = document.querySelector('.chatbox__messages');
    console.log(chatboxButton);
    // const msg = 'Hello,how may I help you?';
    window.addEventListener('load', () => {
      chatboxButton.click();
      chatMessage.innerHTML += 'Greetings !! '
      chatMessage.innerHTML += 'Hi, I am ChatBot. You can select your prefered Language above'
      let count = 1;
      const hidFunc = setInterval(() => {
        const gtrans = document.querySelector('.skiptranslate');
        const gBox = document.querySelector(".goog-te-gadget");
        const chatboxHeader = document.querySelector(".chatbox__header");
        const selectBox = gBox.querySelector('div');
        if (count === 1) {
```

```
        gBox.innerHTML = '';
        chatboxHeader.append(selectBox);
      }
      if (gtrans) {
        gtrans.style.display = 'none';
        gtrans.style.visibility = 'hidden';
        document.body.classList.add('white-bar')
        document.body.style.top = '0';
        // clearInterval(hidFunc);
      }
      // clearInterval(hidFunc);
      count++;
    }, 10)
  })
 </script>
</body>
</html>
```

app.py

```
from urllib import response
from flask import Flask, render_template, request, jsonify
from chat import get_response
app = Flask(__name__)
@app.get("/")
def index_get():
   return render_template("base.html")
@app.post("/predict")
```

```
def predict():
    text = request.get_json().get("message")
    response = get_response(text)
    message = {"answer": response}
    return jsonify(message)


if __name__ == "__main__":
    app.run(debug= True)
```

chat.py

```
import random
import json
import torch
from model import NeuralNet
from nltk_utils import bag_of_words, tokenize
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
with open('intents.json', 'rb') as json_data:
    intents = json.load(json_data)
FILE = "data.pth"
data = torch.load(FILE)
input_size = data["input_size"]
hidden_size = data["hidden_size"]
output_size = data["output_size"]
all_words = data['all_words']
tags = data['tags']
model_state = data["model_state"]
model = NeuralNet(input_size, hidden_size, output_size).to(device)
```

```
model.load_state_dict(model_state)
model.eval()
bot_name = "Sam"
def get_response(msg):
    sentence = tokenize(msg)
    X = bag_of_words(sentence, all_words)
    X = X.reshape(1, X.shape[0])
    X = torch.from_numpy(X).to(device)
    output = model(X)
    _, predicted = torch.max(output, dim=1)
    tag = tags[predicted.item()]
    probs = torch.softmax(output, dim=1)
    prob = probs[0][predicted.item()]
    if prob.item() > 0.75:
        for intent in intents['intents']:
            if tag == intent["tag"]:
                return random.choice(intent['responses'])
        return "I do not understand..."
if __name__ == "__main__":
    print("Let's chat! (type 'quit' to exit)")
    while True:
        # sentence = "do you use credit cards?"
        sentence = input("You: ")
        if sentence == "quit":
            break
        resp = get_response(sentence)
```

```
        print(resp)
```

model.py

```
import torch
import torch.nn as nn
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(hidden_size, hidden_size)
        self.l3 = nn.Linear(hidden_size, num_classes)
        self.relu = nn.ReLU()

    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)
        # no activation and no softmax at the end
        return out
```

nltk_utils.py

```
import numpy as np
import nltk
# nltk.download('punkt')
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()
```

```
def tokenize(sentence):
    """
    split sentence into array of words/tokens
    a token can be a word or punctuation character, or number
    """
    return nltk.word_tokenize(sentence)


def stem(word):
    """
    stemming = find the root form of the word
    examples:
    words = ["organize", "organizes", "organizing"]
    words = [stem(w) for w in words]
    -> ["organ", "organ", "organ"]
    """
    return stemmer.stem(word.lower())


def bag_of_words(tokenized_sentence, words):
    """
    return bag of words array:
    1 for each known word that exists in the sentence, 0 otherwise
    example:
    sentence = ["hello", "how", "are", "you"]
```

```
    words = ["hi", "hello", "I", "you", "bye", "thank", "cool"]
    bog   = [  0 ,    1 ,    0 ,   1 ,    0 ,    0 ,      0]
    """
    # stem each word
    sentence_words = [stem(word) for word in tokenized_sentence]
    # initialize bag with 0 for each word
    bag = np.zeros(len(words), dtype=np.float32)
    for idx, w in enumerate(words):
        if w in sentence_words:
            bag[idx] = 1
    return bag
```

train.py

```
import numpy as np
import random
import json
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from nltk_utils import bag_of_words, tokenize, stem
from model import NeuralNet


with open('intents.json', 'rb') as f:
    intents = json.load(f)
all_words = []
tags = []
xy = []
```

```
# loop through each sentence in our intents patterns
for intent in intents['intents']:
    tag = intent['tag']
    # add to tag list
    tags.append(tag)
    for pattern in intent['patterns']:
        # tokenize each word in the sentence
        w = tokenize(pattern)
        # add to our words list
        all_words.extend(w)
        # add to xy pair
        xy.append((w, tag))
# stem and lower each word
ignore_words = ['?', '.', '!']
all_words = [stem(w) for w in all_words if w not in ignore_words]
# remove duplicates and sort
all_words = sorted(set(all_words))
tags = sorted(set(tags))

print(len(xy), "patterns")
print(len(tags), "tags:", tags)
print(len(all_words), "unique stemmed words:", all_words)

# create training data
X_train = []
y_train = []
```

```
for (pattern_sentence, tag) in xy:
    # X: bag of words for each pattern_sentence
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)
    # y: PyTorch CrossEntropyLoss needs only class labels, not one-hot
    label = tags.index(tag)
    y_train.append(label)
X_train = np.array(X_train)
y_train = np.array(y_train)
# Hyper-parameters
num_epochs = 1000
batch_size = 8
learning_rate = 0.001
input_size = len(X_train[0])
hidden_size = 8
output_size = len(tags)
print(input_size, output_size)
class ChatDataset(Dataset):
    def __init__(self):
        self.n_samples = len(X_train)
        self.x_data = X_train
        self.y_data = y_train
    # support indexing such that dataset[i] can be used to get i-th sample
    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]
    # we can call len(dataset) to return the size
```

```
    def __len__(self):
        return self.n_samples
dataset = ChatDataset()
train_loader = DataLoader(dataset=dataset,
                          batch_size=batch_size,
                          shuffle=True,
                          num_workers=0)
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = NeuralNet(input_size, hidden_size, output_size).to(device)
# Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
# Train the model
for epoch in range(num_epochs):
    for (words, labels) in train_loader:
        words = words.to(device)
        labels = labels.to(dtype=torch.long).to(device)
        # Forward pass
        outputs = model(words)
        # if y would be one-hot, we must apply
        # labels = torch.max(labels, 1)[1]
        loss = criterion(outputs, labels)
        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

```
    if (epoch+1) % 100 == 0:
        print (f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')
print(f'final loss: {loss.item():.4f}')
data = {
"model_state": model.state_dict(),
"input_size": input_size,
"hidden_size": hidden_size,
"output_size": output_size,
"all_words": all_words,
"tags": tags
}
FILE = "data.pth"
torch.save(data, FILE)
print(f'training complete. file saved to {FILE}')
```

APPENDIX-2

SOURCE CODE - ADMIN SIDE

app.py

```
from flask import Flask, redirect, url_for, request , render_template
import data
app = Flask(__name__, template_folder="templates")
# @app.route('/')
# def success():
#     return render_template('index.html')
@app.route('/', methods=["GET", "POST"])
def login():
```

```
    if(request.method == 'POST'):
        tag = request.form['tag']
        pattern = request.form['pattern']
        res = request.form['res']
        value = data.insertData(tag, pattern, res)
        print(value)
        # return redirect(url_for('login'))
        return render_template('admin.html')
    else:
        tag = request.args.get('tag')
        pattern = request.args.get('pattern')
        res = request.form.get('res')
        value = data.insertData(tag, pattern, res)
        print(value)
        # return redirect(url_for('login'))
        return render_template('admin.html')
@app.route('/files', methods=["GET", "POST"])
def files():
    if request.method == "POST":
        file = request.files['file']
        data.excelToJson(file)
    return render_template('upload.html')
    # return redirect(url_for('admin'))
if __name__ == "__main__":
    app.run(debug=True)
```

data.py

```
import json
from turtle import clear
import pandas as pd
import numpy as np
def writeJson(data, fileName):
    with open(fileName, 'w') as f:
        json.dump(data, f)
def insertData(tag, pattern, response):
    # print(f'Pattern : {pattern}\nResponse : {response}')
    with open('demo.json') as file:
        final = json.load(file)
        try:
            temp = final['intent']
            if len(temp)<=0:
                myDict = {"tag": tag, "patterns": [], "responses": []}
                if type(pattern) == type(float('nan')):
                    return -1
                else:
                    myDict['patterns'].append(pattern)
                if type(response) == type(float('nan')):
                    return -1
                else:
                    myDict['responses'].append(response)
                data = final['intent']
                data.append(myDict)
```

```
        writeJson(final, "demo.json")
        return 'Data updated successfully'
    else:
        for i in range(len(temp)):
            data = final['intent'][i]
            if (('tag' in data.keys()) and (data['tag']==tag)):
                if(pattern not in data['patterns']):
                    if type(pattern) == type(float('nan')):
                        return -1
                    else:
                        data['patterns'].append(pattern)
                    if type(response) == type(float('nan')):
                        return -1
                    else:
                        data['responses'].append(response)
                writeJson(final, "demo.json")
                return 'Data updated successfully'
            else:
                myDict = {"tag": tag, "patterns": [], "responses": []}
                if type(pattern) == type(float('nan')):
                    return -1
                else:
                    myDict['patterns'].append(pattern)
                if type(response) == type(float('nan')):
                    return -1
                else:
```

```
                    myDict['responses'].append(response)
                data = final['intent']
                data.append(myDict)
            writeJson(final, "demo.json")
            return 'Data updated successfully'
        # return 'Data updated successfully'
    # temp.append(ls)
    # print(temp)
    except KeyError as e:
        print(f'Error occered')
def excelToJson(fileName):
    df = pd.read_excel(fileName, engine='odf')
    # df = df1.replace(np.nan, ' ', inplace=True)
    for i in df.index:
        a=insertData(df['Tags'][i], df['Patterns'][i], df['Response'][i])
        print(a)
# excelToJson('values.ods')
```

APPENDIX-3

OUTPUT - SCREENSHOTS

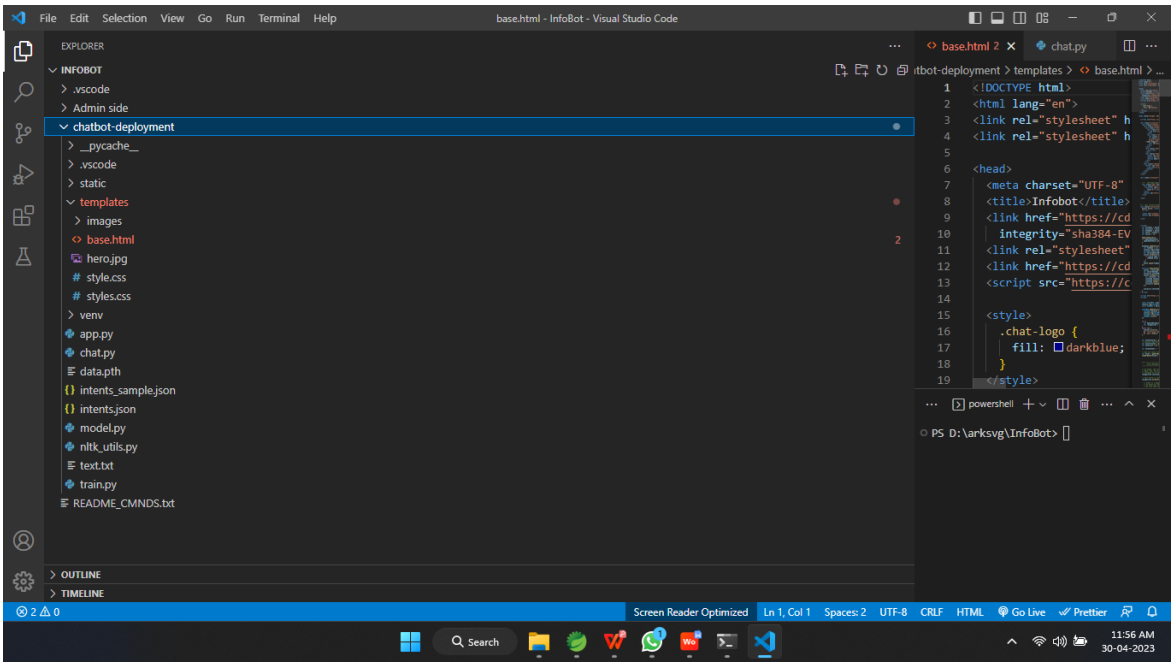


Fig.8.1 Chatbot Directory

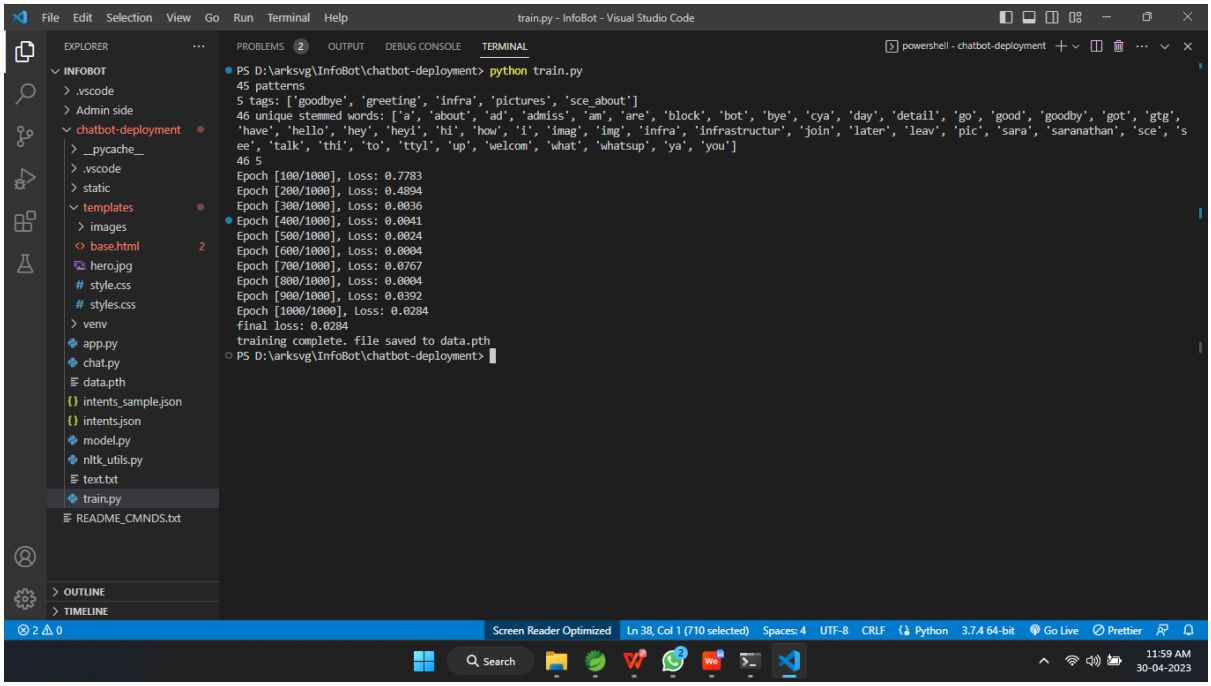


Fig.8.2 Epochs Trained

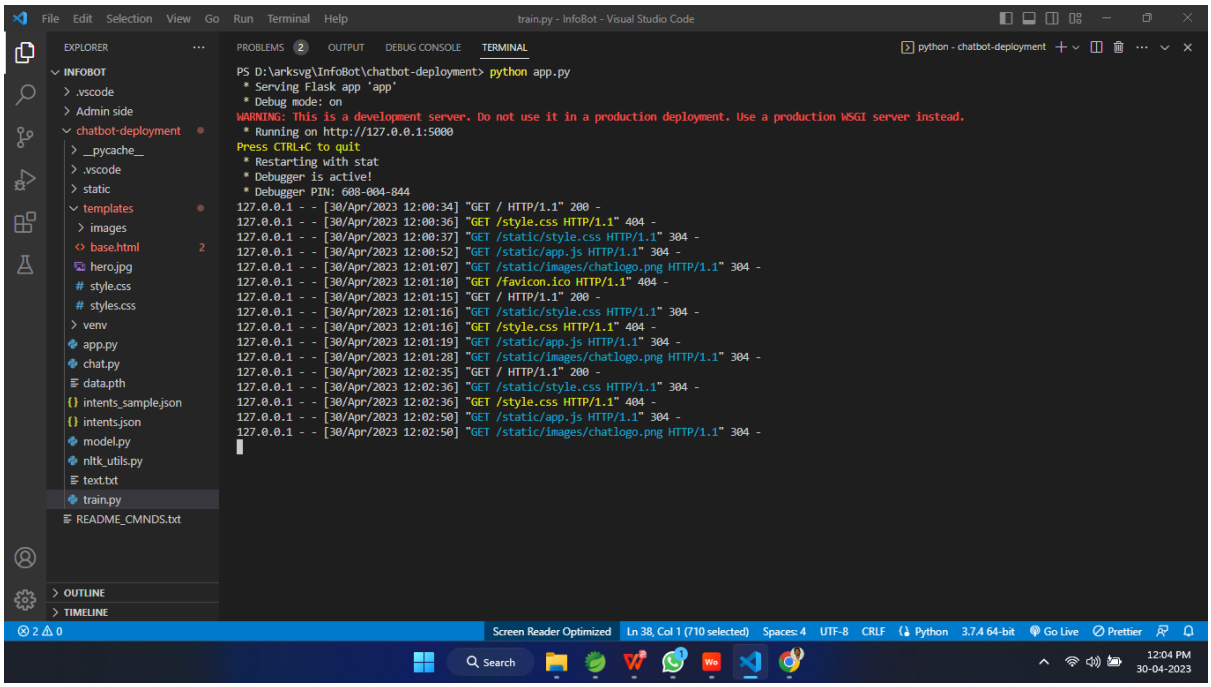


Fig.8.3 Server Loading

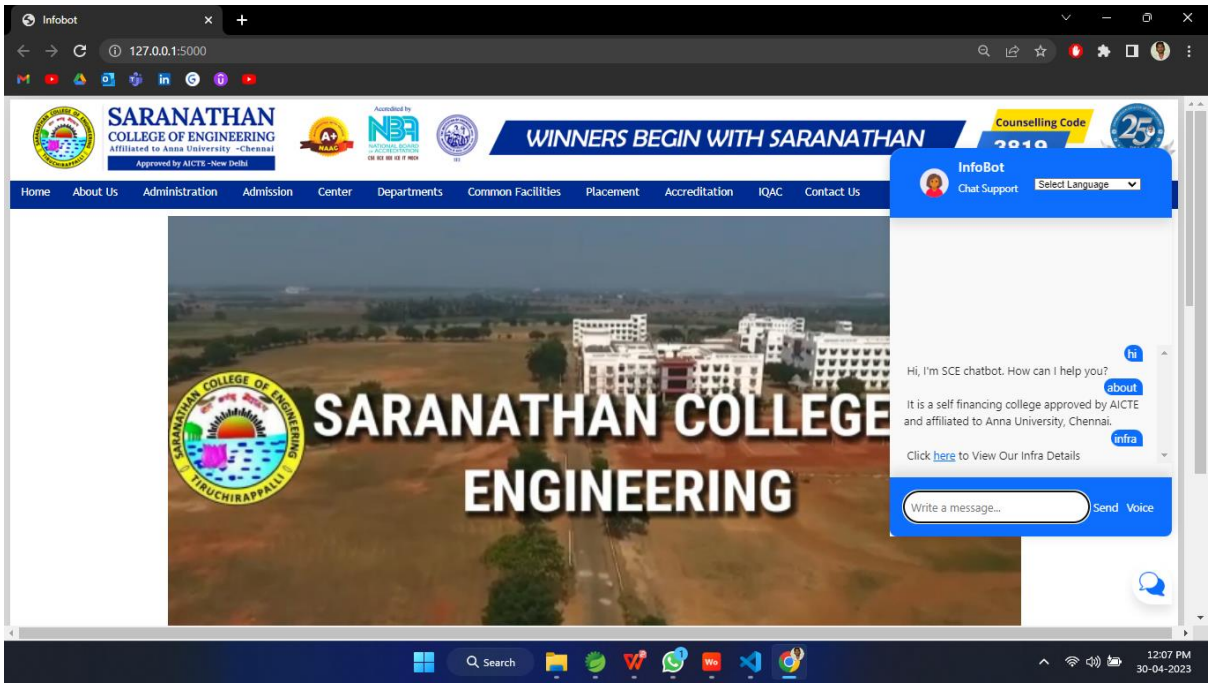


Fig.8.4 Output of User module

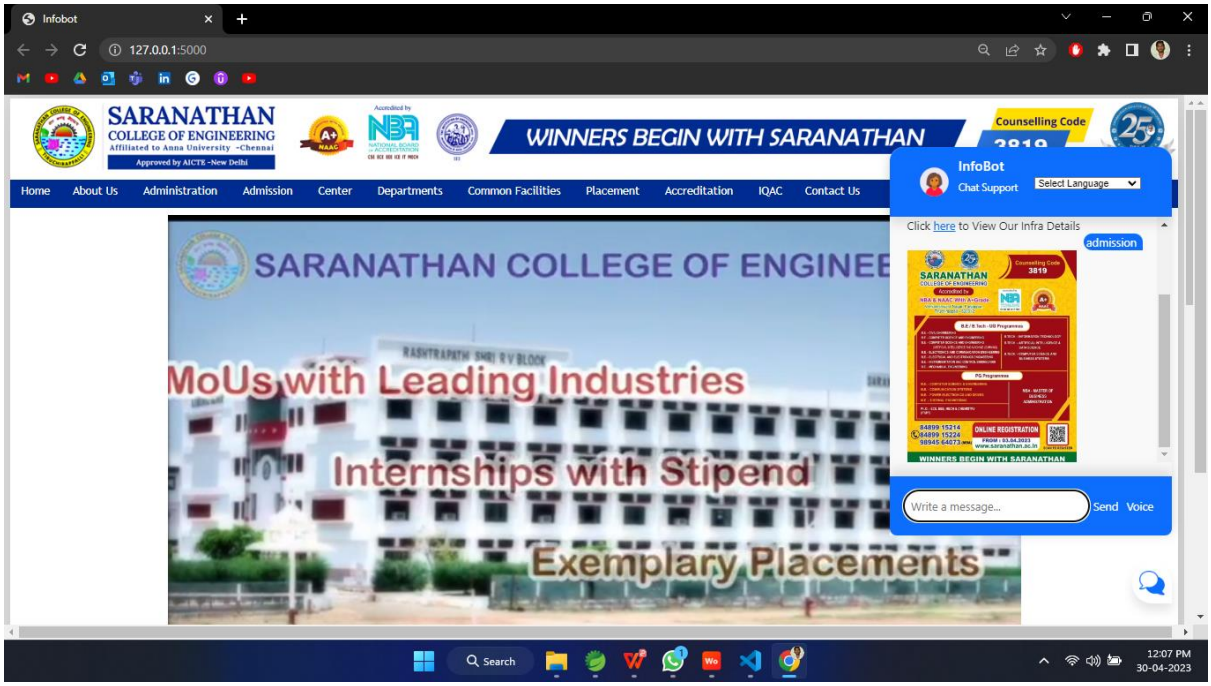


Fig.8.5 Multimedia Support

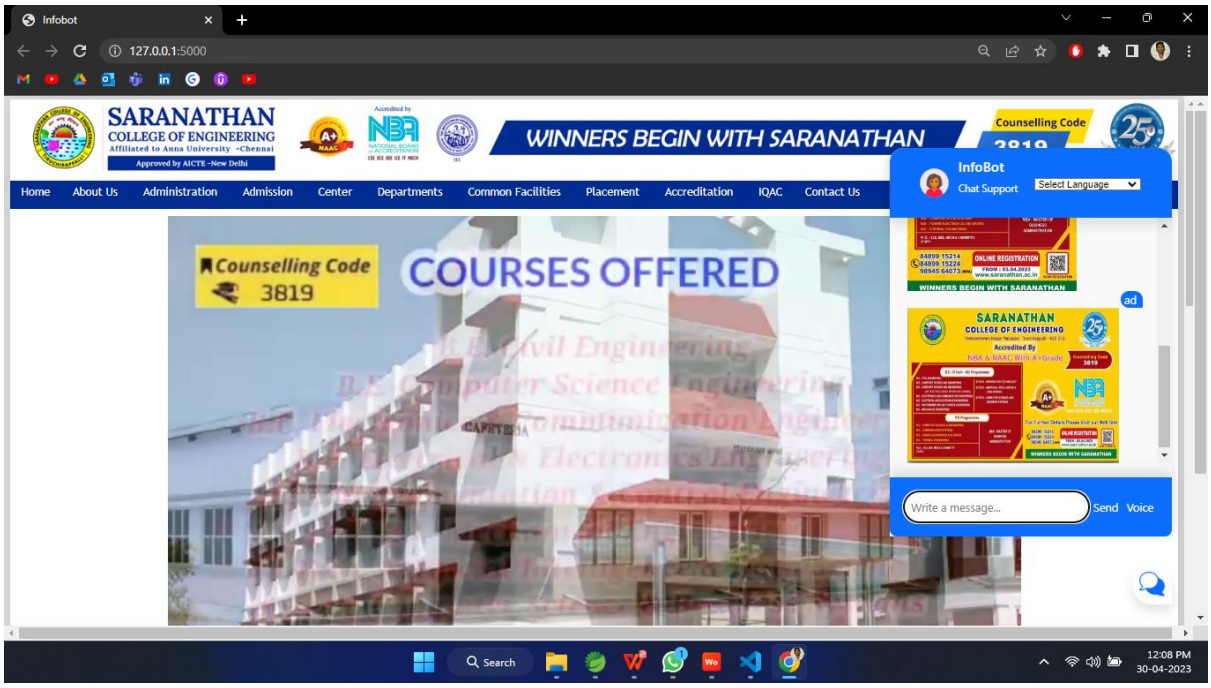


Fig.8.6 Image with details

Fig.8.7 Choosing Language for translation

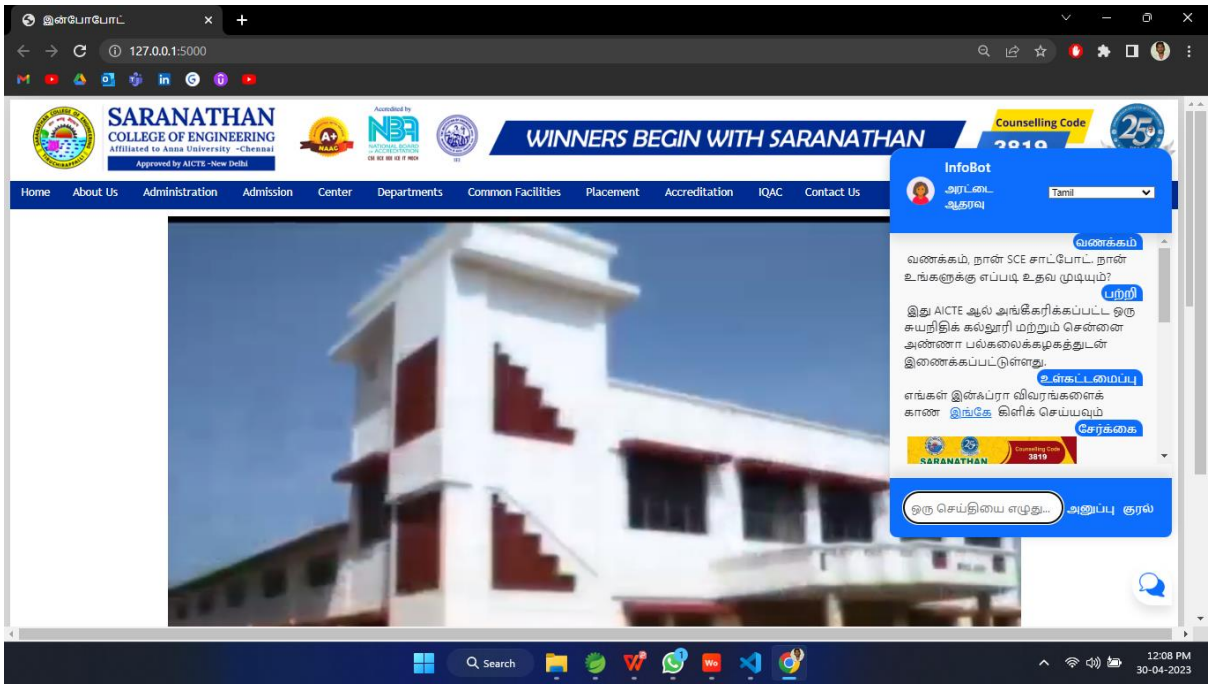


Fig.8.8 Content after translation

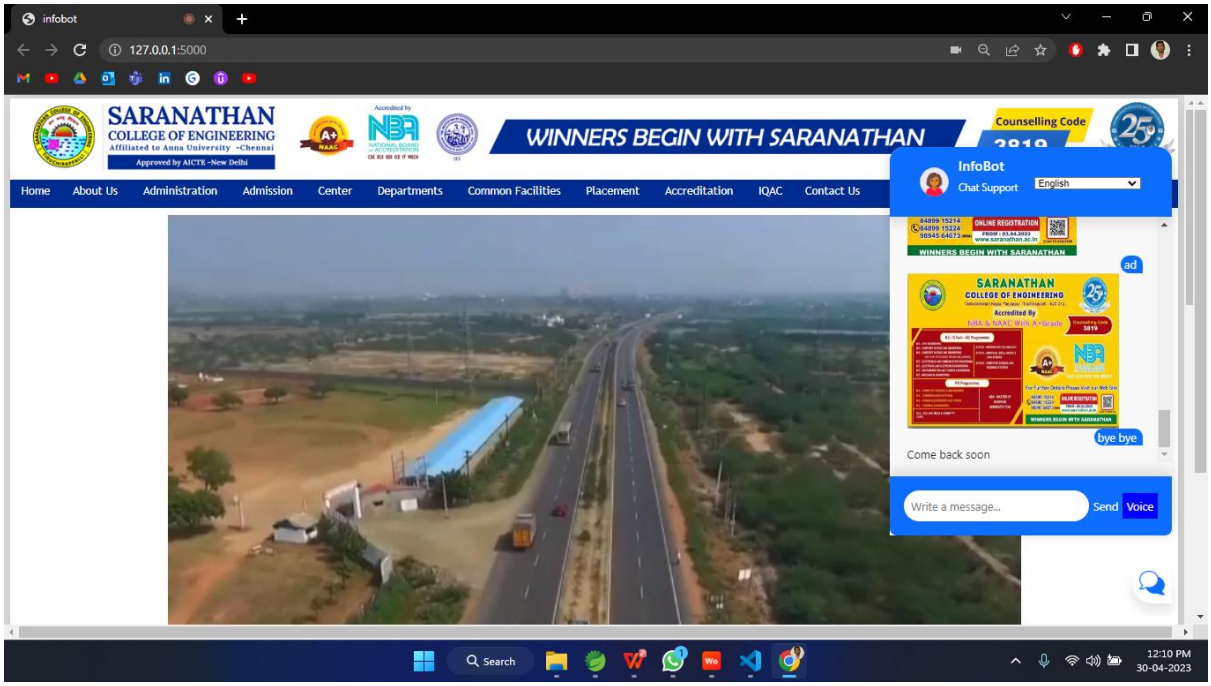


Fig.8.9 Speech recognition output

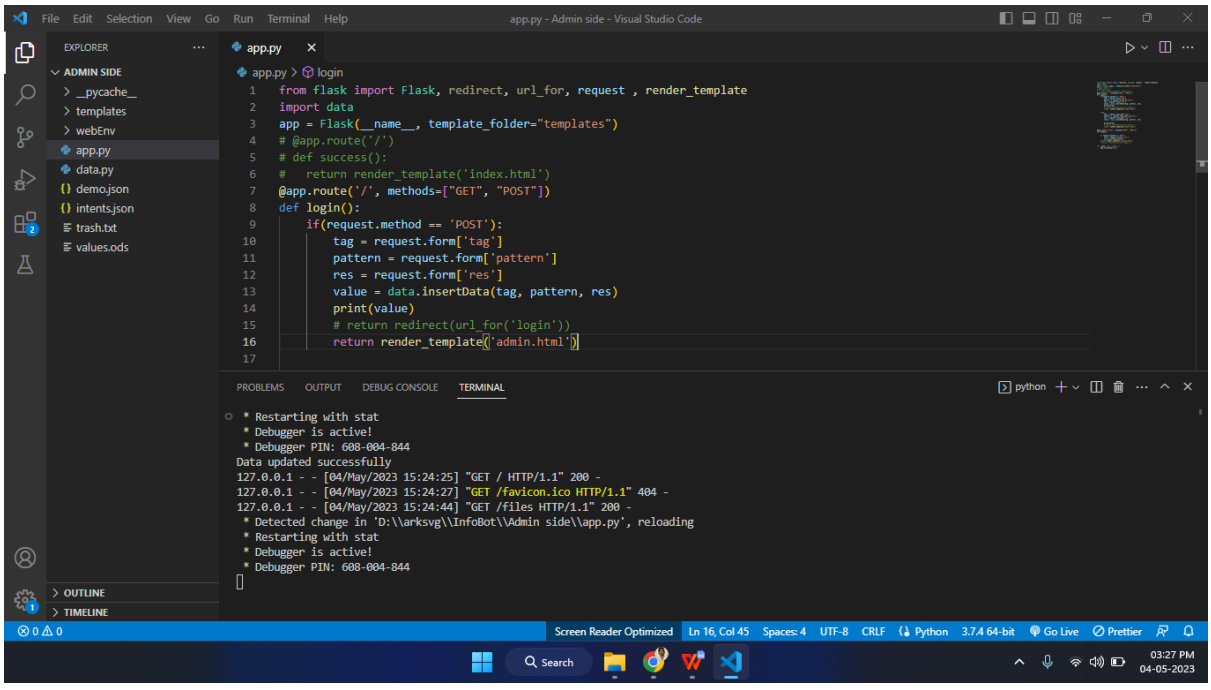


Fig.8.10 Admin module code

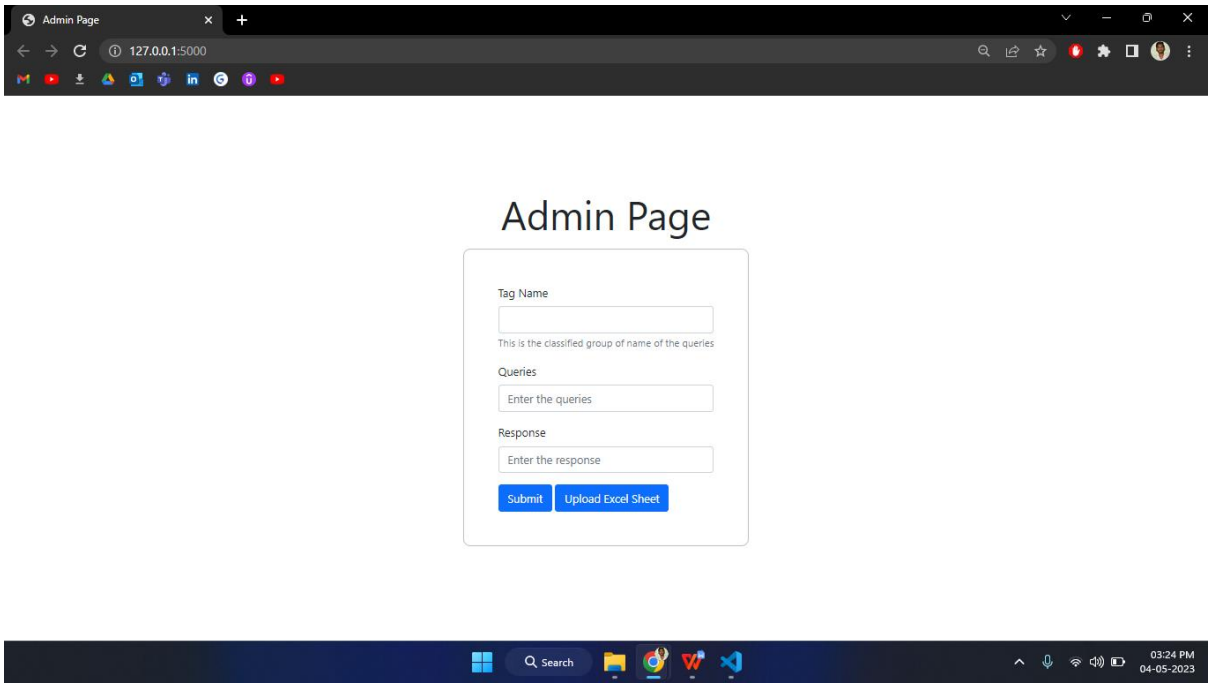


Fig.8.11 Output of Admin module

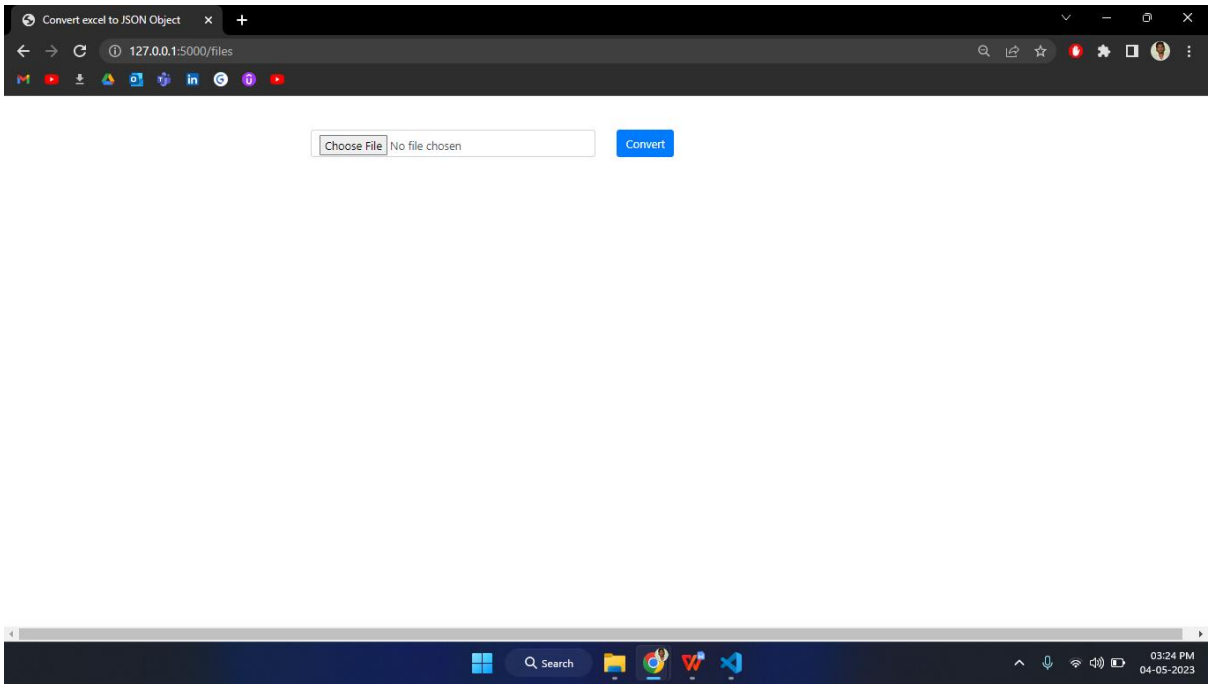


Fig.8.12 Admin module input page

REFERENCES

[1] Achmad Ramaditiya, Aris Munawar, Octarina Nur Samijayani, Suci Rahmatia.Date of Conference: 29-30 June 2021.Implementation Chatbot Whatsapp using Python Programming for Broadcast and Reply Message Automatically.

[2] Adamopoulou and L. Moussiades, "An overview of chatbot technology", IFIP International Conference on Artificial Intelligence Applications and Innovations, pp. 373-383, 2020.

[3] Adamopoulou and L. Moussiades, "Chatbots: History technology and applications", Machine Learning with Applications, vol. 2, pp. 100006, 2020.

[4] Ankit Kumar, Hrithik Paul, Kamred Udham Singh, Mohd Asif Shah, Sanjay Chakraborty, Saroj Kumar Pandey, Sayani Ghatak.Date of Publication: 06 December 2022.An AI-Based Medical Chatbot Model for Infectious Disease Prediction.

[5] Ed-Douibi, J. L. C. Izquierdo, G. Daniel and J. Cabot, "A Model-based Chatbot Generation Approach to Converse with Open Data Sources", arXiv preprint arXiv:2007, vol. 10503, 2020.

[6] Giancarlo Sperli.SAC '20: Proceedings of the 35th Annual ACM Symposium on Applied Computing March 2020.https://doi.org/10.1145/3341105.3374129.

[7] Gurasis Singh, Gurram Gopinadh, Manik Rakhra, Muthumula Navaneeswar Reddy, Narasimha Sai Addepalli, Shaik Aliraja, Vennapusa Siva Ganeshwar Reddy.Date of Conference: 28-30 April 2021.E-Commerce Assistance with a Smart Chatbot using Artificial Intelligence.

[8] Khan and M. R. Rabbani, "Chatbot as islamic finance expert (caife) when finance meets artificial intelligence", Proceedings of the 2020 4th International Symposium on Computer Science and Intelligent Control, pp. 1-5, 2020.

[9] Mohammad Monirujjaman Khan.Date of Conference: 14-16 December2020.Development of An e-commerce Sales Chatbot.

[10] Norihide Kitaoka,Takahiro Kinouchi,"A response generation method of chat-bot system using input formatting and reference resolution",Date of Conference: 28-29 September 2022,Date Added to IEEE Xplore: 02 November 2022,INSPEC Accession Number: 22212574,DOI: 10.1109/ICAICTA56449.2022.9932928