CHATBOTAI USING PYTHON

A PROJECT REPORT

Submitted by

PRAKASH PAUDEL.P	**(211420205108)**
VIGNESH VARUN .J	**(211420205177)**
DHANASEKAR.S	**(211420205040)**

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE, POONAMALLEE
ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“CHATBOT AI USING PYTHON”** is the bonafide work of **“PRAKASH PAUDEL.P (211420205108), VIGNESH VARUN.J (211420205177) DHANASEKAR.S (211420205040)”** who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Dr. M. HELDA MERCY M.E., Ph.D.,	**Mrs. MUTHULAKSHMI M.Tech.,(Ph.D)**
HEAD OF THE DEPARTMENT	**ASSOCIATE PROFESSOR (SUPERVISOR)**
Department of Information Technology	Department of Information Technology
Panimalar Engineering College	Panimalar Engineering College
Poonamallee, Chennai - 600 123	Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on ___________________________

DECLARATION

I hereby declare that the project report entitled "**CHATBOT AI USING PYTHON"** which is being submitted in partial fulfilment of the requirement of the course leading to the award of the 'Bachelor of Technology in Information Technology' in **Panimalar Engineering College, An Autonomous institution Affiliated to Anna University- Chennai** is the result of the project carried out by me under the guidance and supervision of **Mrs. K.MUTHULAKSHMI, M.TECH.,(Ph.D) Associate Professor in the Department of Information Technology**. I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

(**Prakash paudel.P**)

(Vignesh varun.J)

Date:

(Dhanasekar.S)

Place**:** Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date:

DR.D.**KARUNKUZHALI M.Tech.,Ph.D,**

Place: Chennai

(Associate Professor/ IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co- operation and support from many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Honorable Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors, Mrs. C. VIJAYA RAJESHWARI and Dr.C.SAKTHI KUMAR, M.E.,Ph.D** and **Dr.SARANYA SREE SAKTHIKUMAR., B.E., M.B.A.,Ph.D** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our staff in charge, **DR.D.KARUNKUZHALI M.Tech.,Ph.D,** Associate Professor, Department of Information Technology for her guidance throughout the course of our project. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

ABSTRACT

A chatbot is an artificial intelligence (AI) programme that simulates human-like dialogue with users through text or voice interfaces using natural language processing (NLP). As businesses and organisations seek to automate customer service and boost operational efficiency, chatbots have grown in popularity in recent years.Although chatbots have been around since the 1960s, they weren't practicable or widely employed until the internet age and the development of machine learning and artificial intelligence (AI) technology.

Today, chatbots are employed in a wide range of sectors, including e-commerce, healthcare, and banking.Chatbots can be programmed to carry out a variety of functions, including customer service, completing purchases, and responding to commonly asked inquiries. For a smooth user experience, they can also be combined with other technologies, like voice assistants like Amazon's Alexa or Google Assistant.Customer experience improvement is one of the main advantages of employing chatbots. Chatbots can be available around-the-clock to respond to client inquiries, offering rapid and effective solutions that can foster repeat business. By automating routine processes and lowering the need for human involvement, they can also assist enterprises in saving time and money.

With improvements in machine learning and deep learning algorithms, chatbot AI technology is continually developing, enabling more complex and individualised interactions. In order to make personalised recommendations and increase user engagement, chatbots can now analyse user data. However, there are drawbacks to chatbot AI as well. Making sure chatbots can effectively interpret and respond to user queries is one of the main concerns. NLP systems occasionally have trouble deciphering user intent, which results in inaccurate or irrelevant responses. To overcome this difficulty, chatbot developers are continuously enhancing their algorithms and putting a lot of data to use while training their chatbots.

To sum up, chatbot AI technology is an area that is fast developing and has enormous potential to enhance customer experience and streamline processes in a variety of businesses. Chatbots are set to become even more common as companies and organisations continue to invest in AI technologies, giving customers an easy and tailored way to communicate with companies and organisations.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Artificial Intelligence (AI) increasingly integrates our daily lives with the creation and analysis of intelligent software and hardware, called intelligent agents. Intelligent agents can do a variety of tasks ranging from labor work to sophisticated operations. A chatbot is a typical example of an AI system and one of the most elementary and widespread examples of intelligent Human-Computer Interaction (HCI).

It is a computer program, which responds like a smart entity when conversed with through text or voice and understands one or more human languages by Natural Language Processing (NLP). In the lexicon, a chatbot is defined as "A computer program designed to simulate conoversation with human users, especially over the Internet" . Chatbots are also known as smart bots, interactive agents, digital assistants, or artificial conversation entities. Chatbots can mimic human conversation and entertain users but they are not built only for this.

They are useful in applications such as education, information retrieval, business, and e-commerce . They became so popular because there are many advantages of chatbots for users and developers too. Most implementations are platform-independent and instantly available to users without needed installations. Contact to the chatbot is spread through a user's social graph without leaving the messaging app the chatbot lives in, which provides and guarantees the user's identity. Moreover, payment services are integrated into the messaging system and can be used safely and reliably and a notification system re-engages inactive users. Chatbots are integrated with group conversations or shared just like any other contact, while multiple conversations can be carried forward in parallel. Knowledge in the use of one chatbot is easily transferred to the usage of other chatbots, and there are limited data requirements. Communication reliability, fast and uncomplicated development iterations, lack of version fragmentation, and limited design efforts for the interface are some of the advantages for developers too .

1.1 OBJECTIVES

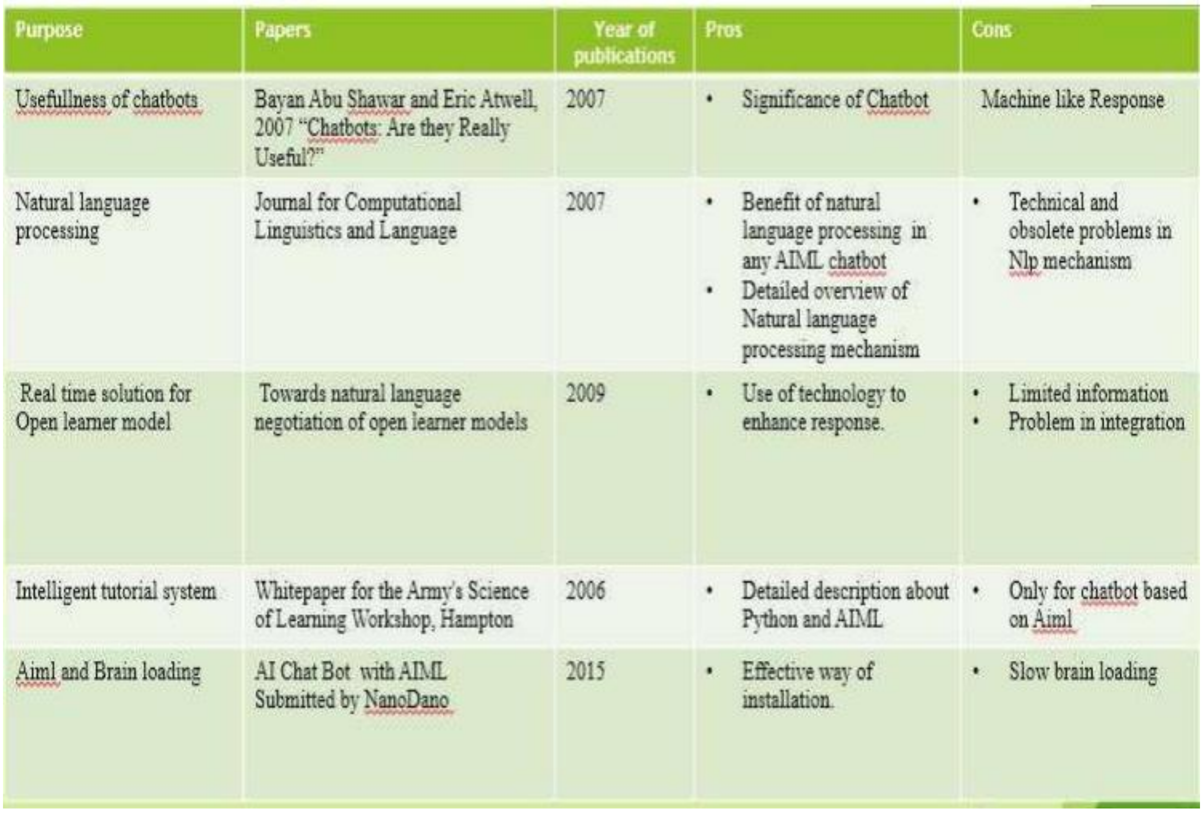
The objective of chatbot AI is to provide a better user experience by leveraging artificial intelligence technologies such as machine learning and natural language processing. Chatbot AI can be used to improve customer experience by providing quick and efficient responses to customer inquiries, streamlining operations, and reducing the need for human intervention. Additionally, it can provide personalized recommendations and solutions, leading to increased engagement and sales. Chatbot AI can also enhance a brand's image by creating a consistent brand image and tone across all interactions. By reducing wait times, handling multiple user inquiries simultaneously, and being more cost-effective than human customer service representatives, chatbot AI can increase efficiency and reduce costs, leading to better business outcomes. Ultimately, the objective of chatbot AI is to provide a more efficient, engaging, and personalized user experience that can drive growth and success for businesses.

1.2 Scope

The scope of chatbots is expanding rapidly as they become an essential tool for businesses of all sizes and industries. Chatbots can be used to automate repetitive tasks, provide quick and efficient customer service, and personalize interactions with users.Some common use cases for AI chatbots include customer support, sales and marketing, appointment scheduling, human resources, and entertainment and gaming. Each of these use cases may require different functionalities and capabilities for the chatbot to be effective and AI chatbot is generally broader and more advanced than a traditional chatbot due to the use of natural language processing and machine learning algorithms. While a traditional chatbot may rely on a set of predefined rules or responses, an AI chatbot has the ability to understand and respond to natural language input in a more sophisticated manner.

2. Literature Survey

2.1 literature survey table



Purpose	Papers	Year of publications	Pros	Cons
Usefullness of chatbots	Bayan Abu Shawar and Eric Atwell, 2007 "Chatbots: Are they Really Useful?"	2007	• Significance of Chatbot	Machine like Response
Natural language processing	Journal for Computational Linguistics and Language	2007	• Benefit of natural language processing in any AIML chatbot • Detailed overview of Natural language processing mechanism	• Technical and obsolete problems in Nlp mechanism
Real time solution for Open learner model	Towards natural language negotiation of open learner models	2009	• Use of technology to enhance response.	• Limited information • Problem in integration
Intelligent tutorial system	Whitepaper for the Army's Science of Learning Workshop, Hampton	2006	• Detailed description about Python and AIML	• Only for chatbot based on Aiml
Aiml and Brain loading	AI Chat Bot with AIML Submitted by NanoDano	2015	• Effective way of installation.	• Slow brain loading

Fig 1:literature survey table

2.2 Natural Language Processing

Natural Language Processing (NLP) is the study of letting computers understand human languages. Without NLP, human language sentences are just a series of meaningless symbols to computers. Computers don't recognize the words and don't understand the grammars. NLP can be regard as a "translator", who will translate human languages to computer understandable information. Traditionally, users need to follow well-defined procedures accurately, in order to interact with computers. For example, in Linux systems, all commands must be precise. A single replace of one character or even a space can have significant difference. However, the emergence of NLP is changing the way of interacting. Apple Siri and Microsoft Cortana have made it possible to give command in everyday languages and is changing the way of interacting.

Speech recognition, also called speech-to-text, is the task of reliably converting voice data into text data. Speech recognition is required for any application that follows voice commands or answers spoken questions. What makes speech recognition especially challenging is the way people talk—quickly, slurring words together, with varying emphasis and intonation, in different accents, and often using incorrect grammar.

Part of speech tagging, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context. Part of speech identifies 'make' as a verb in 'I can make a paper plane,' and as a noun in 'What make of car do you own?'

Word sense disambiguation is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context. For example, word sense disambiguation helps distinguish the meaning of the verb 'make' in 'make the grade' (achieve) vs. 'make a bet' (place).

Named entity recognition, or NEM, identifies words or phrases as useful entities. NEM identifies 'Kentucky' as a location or 'Fred' as a man's name.

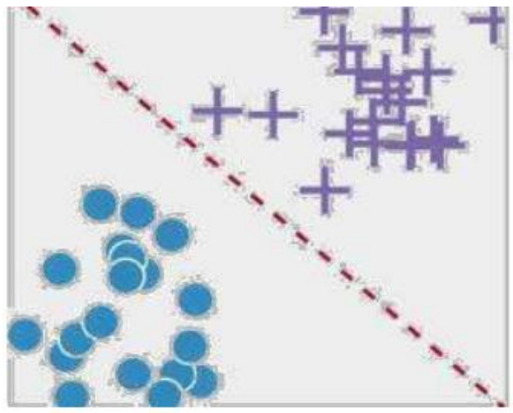
Co-reference resolution is the task of identifying if and when two words refer to the same entity. The most common example is determining the person or object to which a certain pronoun refers (e.g., 'she' = 'Mary'), but it can also involve identifying a metaphor or an idiom in the text (e.g., an instance in which 'bear' isn't an animal but a large hairy person).

Sentiment analysis attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text.

Natural language generation is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

2.3 Machine Learning.

Machine Learning (ML) is an area of computer science that "gives computers the ability to learn without being explicitly programmed". The parameter of the formulas is calculated from the data, rather than defined by the programmer. Two most common usage of ML is Classification and Regression. As shown in figure, Classification means to categorize different types of data, while Regression means to find a way to describe the data. Basic ML program will have two stages, fittingand predicting.In the fitting stage, the program will be given a large set (at least thousands) of data. The program will try to adjust its parameter based on some statistical models, in order to make it “fit” the input data best. In the predicting stage, the program will give a prediction for a new input based on the parameters it just calculated out. For example, the famous Iris flower dataset contains the measurement of several features of three different species of flowers, such as the length of sepals and petals. A well-defined ML program can learn the pattern behind this feature and give prediction accordingly.


Classification figure

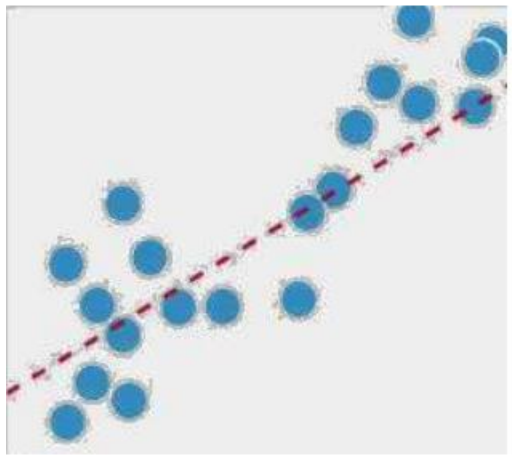

Regressing

Fig 2: Machine learning diagrams

2.4 Chatbot and Machine learning

Machine learning chatbots works using artificial intelligence. User need not to be more specific while talking with a bot because it can understand the natural language, not only commands. This kind of bots get continuously better or smarter as it learns from past conversations it had with people. Here is a simple example which illustrate how they work. The following is a conversation between a human and a chatbot: Human: “I need a flight from San Jose to New York.” Bot: “Sure! When would you like to travel?” Human: “From Dec 20, 2016 to Jan 28, 2017.” Bot: “Great! Looking for flights.” In order to achieve the ultimate goal, I have taken an iterative approach and divided my work into four major deliverables.

These deliverables not only helped me in understanding the code structure of Yioop but also enhances Yioop’s functionality. In the rest of the report, I will be discussing about the four deliverables. To understand more on chatbot service, I had implemented a Facebook Messenger Weather Bot in deliverable 1, which is discussed in next section. The purpose of deliverable 2 is to introduce chatbots to the Yioop. I have added Bot Configuration settings which is used to add bot users in Yioop. In the next deliverable, I have added a functionality where the user will be able to call bots in a group thread.

Activation of bots will happen by calling respective callback URL which is already configured that helps bots to have a conversation with users. More details on this is discussed in deliverable 3 section. As a deliverable 4, I have created a weather bot i.e, a web application in php that calls yahoo API to get weather information. The last section of the report contains the conclusion and future work. I have implemented a Facebook Messenger Bot to get an overview of how chatbot is build. During this implementation, I understood the flow of control for a chatbot service with other services which is explained below.

In order to create a Facebook Messenger Bot, a developer needs to be authenticated and approved by Facebook to converse with the public and the web server for security reasons. For a Facebook Messenger Bot, I have created a simple web application using Node.js by installing the necessary dependencies using npm. I ran this locally. I also downloaded and installed ngrok and started it - npm run ngrok. This launched a Forwarding URL to the local running server, that means any requests to Forwarding URL will hit the locally running server. This url is used as a 3 | P a g e Callback URL in Facebook App which will be explained further. To set up the Facebook App, I have created a Facebook Page and Facebook App using my Facebook account. While setting up a Webhook in the app settings, I have given the Forwarding URL as Callback URL and added code for verification.The access token in page settings is stored as environment variable as it will be used in integration. In order to make webhook to receive messages from this page, the app is subscribed to the page created. To set up the bot to handle the POST calls at webhook, I have created a webhook endpoint in the sample application.

2.5 Artificial Intelligence

AI was coined by John McCarthy, an American computer scientist, in 1956 at The Dartmouth Conference where the discipline was born. Today, it is an umbrella term that encompasses everything from robotic process automation to actual robotics. It has gained prominence recently due, in part, to big data, or the increase in speed, size and variety of data businesses are now collecting. AI can perform tasks such as identifying patterns in the data more efficiently than humans, enabling businesses to gain more insight out of their data.

2.6 Types of artificial intelligence

AI can be categorized in any number of ways, but here are two examples. The first classifies AI systems as either weak AI or strong AI. Weak AI, also known as narrow AI, is an AI system that is designed and trained for a particular task. Virtual personal assistants, such as Apple's Siri, are a form of weak AI. Strong AI, also known as artificial general intelligence, is an AI system with generalized human cognitive abilities so that when presented with an unfamiliar task, it has enough intelligence to find a solution. The Turing Test, developed by mathematician Alan Turing in 1950, is a method used to determine if a computer can actually think like a human, although the method is controversial. The second example is from Arend Hintze, an assistant professor of integrative biology and computer science and engineering at Michigan State University. He categorizes AI into four types, from the kind of AI systems that exist today to sentient systems, which do not yet exist.

Type 1:

Reactive machines. An example is Deep Blue, the IBM chess program that beat Garry Kasparov in the 1990s. Deep Blue can identify pieces on the chess board and make predictions, but it has no memory and cannot use past experiences to inform future ones. It analyzes possible moves -- its own and its opponent -- and chooses the most strategic move. Deep Blue and Google's AlphaGO were designed for narrow purposes and cannot easily be applied to another situation.

Type 2:

Limited memory. These AI systems can use past experiences to inform future decisions. Some of the decisionmaking functions in autonomous vehicles have been designed this way. Observations used to inform actions happening in the not-so-distant future, such as a car that has changed lanes. These observations are not stored permanently.

Type 3:

Theory of mind. This is a psychology term. It refers to the understanding that others have their own beliefs, desires and intentions that impact the decisions they make. This kind of AI does not yet exist.

Type 4:

Self-awareness. In this category, AI systems have a sense of self, have consciousness. Machines with self-awareness understand their current state and can use the information to infer what others are feeling. This type of AI does not yet exist.

2.7 CLASSIFICATIONS OF CHATBOT

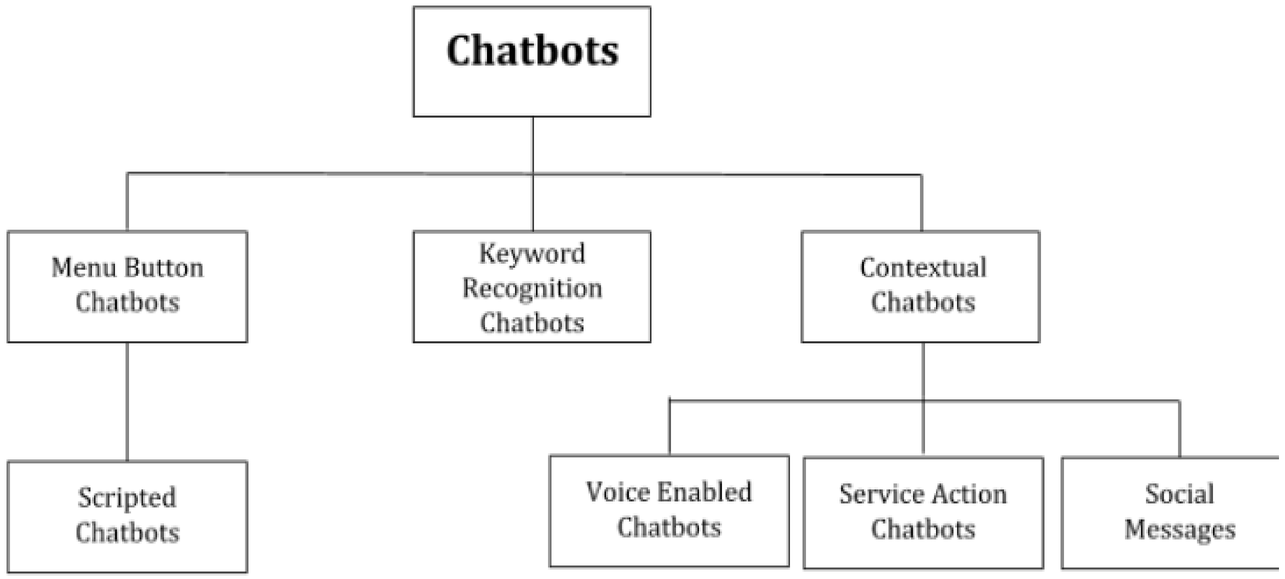


Fig 3: classifications of chatbot

2.8 PREFERENCES OF CHATBOT

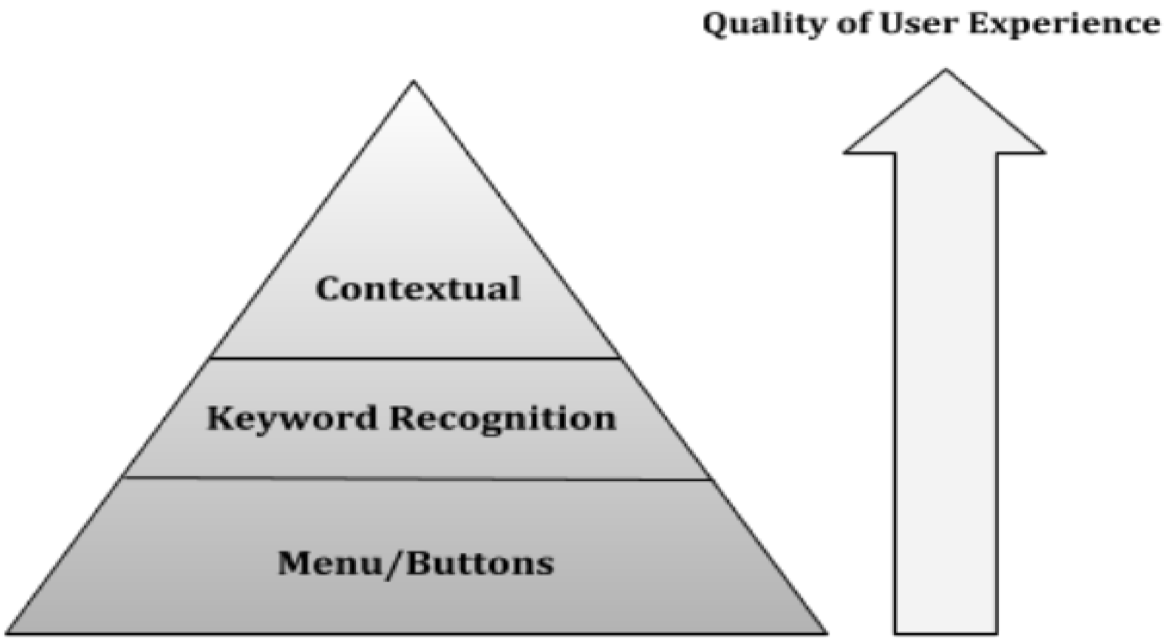


Fig 4: preferences of chatbot

2.9 TYPES OF CHATBOT

Chatbots can be classified using different parameters: the knowledge domain, the service provided, the goals, the input processing and response generation method, the human-aid, and the build method. Classification based on the knowledge domain considers the knowledge a chatbot can access or the amount of data it is trained upon. Open domain chatbots can talk about general topics and respond appropriately, while closed domain chatbots are focused on a particular knowledge domain and might fail to respond to other questions.

Classification based on the service provided considers the sentimental proximity of the chatbot to the user, the amount of intimate interaction that takes place, and it is also dependent upon the task the chatbot is performing. Interpersonal chatbots lie in the domain of communication and provide services such as Restaurant booking, Flight booking, and FAQ bots. They are not companions of the user, but they get information and pass them on to the user. They can have a personality, can be friendly, and will probably remember information about the user, but they are not obliged or expected to do so. Intrapersonal chatbots exist within the personal domain of the user, such as chat apps like Messenger, Slack, and WhatsApp. They are companions to the user and understand the user like a human does.Inter-agent chatbots become omnipresent while all chatbots will require some inter-chatbot communication possibilities. The need for protocols for inter-chatbot communication has already emerged. Alexa-Cortana integration is an example of inter-agent communication. Classification based on the goals considers the primary goal chatbots aim to achieve. Informative chatbots are designed to provide the user with information that is stored beforehand or is available from a fixed source, like FAQ chatbots. Chat based/Conversational chatbots talk to the user, like another human being, and their goal is to respond correctly to the sentence they have been given.Task-based chatbots perform a specific task such as booking a flight or helping somebody. These chatbots are intelligent in the context of asking for information and understanding the user's input. Restaurant booking bots and FAQ chatbots are examples of Task-based chatbots.

Classification based on the input processing and response generation method takes into account the method of processing inputs and generating responses. There are three models used to produce the appropriate responses: rule-based model, retrieval based model, and generative model . Rule-based model chatbots are the type of architecture which most of the first chatbots have been built with, like numerous online chatbots. They choose the system response based on a fixed predefined set of rules, based on recognizing the lexical form of the input text without creating any new text answers. The knowledge used in the chatbot is humanly hand-coded and is organized and presented with conversational patterns . A more comprehensive rule database allows the chatbot to reply to more types of user input. However, this type of model is not robust to spelling and grammatical mistakes in user input. Most existing research on rule-based chatbots studies response selection for single-turn conversation, which only considers the last input message. In more human like chatbots, multi-turn response selection takes into consideration previous parts of the conversation to select a response relevant to the whole conversation context . A little different from the rule-based model is the retrieval-based model, which offers more flexibility as it queries and analyzes available resources using APIs A retrieval-based chatbot retrieves some response candidates from an index before it applies the matching approach to the response selection .

The generative model generates answers in a better way than the other three models, based on current and previous user messages. These chatbots are more human-like and use machine learning algorithms and deep learning techniques. However, there are difficulties in building and training them . Another classification for chatbots considers the amount of human-aid in their comoponents. Human-aided chatbots utilize human computation in at least one element from the chatbot. Crowd workers, freelancers, or full-time employees can embody their intelligence in the chatbot logic to fill the gaps caused by limitations of fully automated chatbots. While human computation, compared to rule-based algorithms and machine learning, provides more flexibility and robustness, still, it cannot process a given piece of information as fast as a machine, which makes it hard to scale to more user requests . Chatbots can also be classified according to the permissions provided by their development platform. Development platforms can be of open-source, such as RASA, or can be of proprietary code such as development platforms typically offered by large companies such as Google or IBM.

Open-source platforms provide the chatbot designer with the ability to intervene in most aspects of implementation. Closed platforms, typically act as black boxes, which may be a significant disadvantage depending on the project requirements. However, access to state-of-the-art technologies may be considered more immediate for large companies. Moreover, one may assume that chatbots developed based on large companies’ platforms may be benefited by a large amount of data that these companies collect. Of course, chatbots do not exclusively belong to one category or another, but these categories exist in each chatbot in varying proportions.

3. System Design

3.1 Proposed Architecture design for ChatbotAI

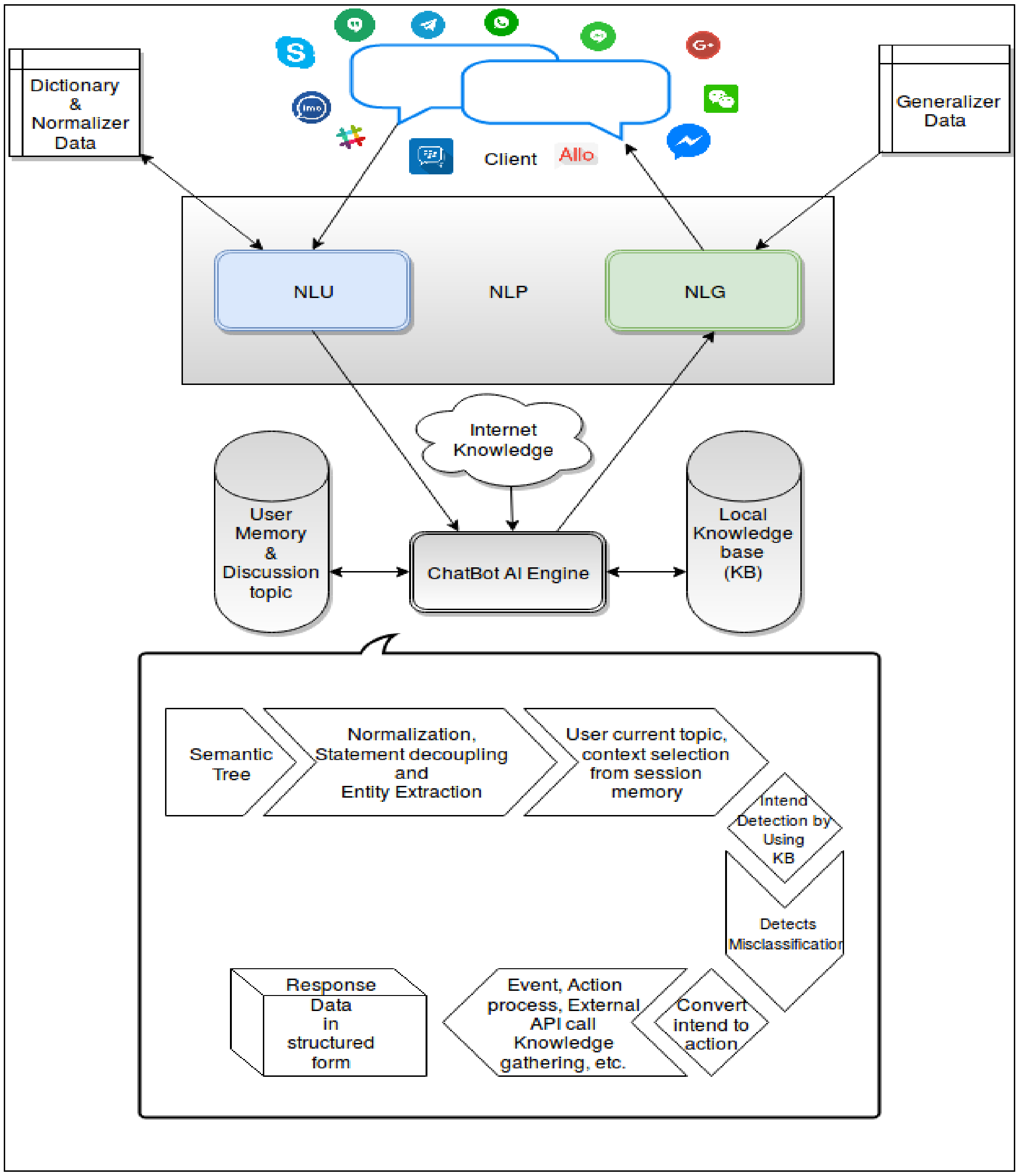

Dictionary & Normalizer Data
Client
Allo
Generalizer Data
NLU
NLP
NLG
Internet Knowledge
User Memory & Discussion topic
ChatBot AI Engine
Local Knowledge base (KB)
Semantic Tree
Normalization, Statement decoupling and Entity Extraction
User current topic, context selection from session memory
Intend Detection by Using KB
Detects Misclassification
Convert intend to action
Event, Action process, External API call Knowledge gathering, etc.
Response Data in structured form

Fig 5: proposed architecture design of chatbot

3.2 Genral chatbot architecture

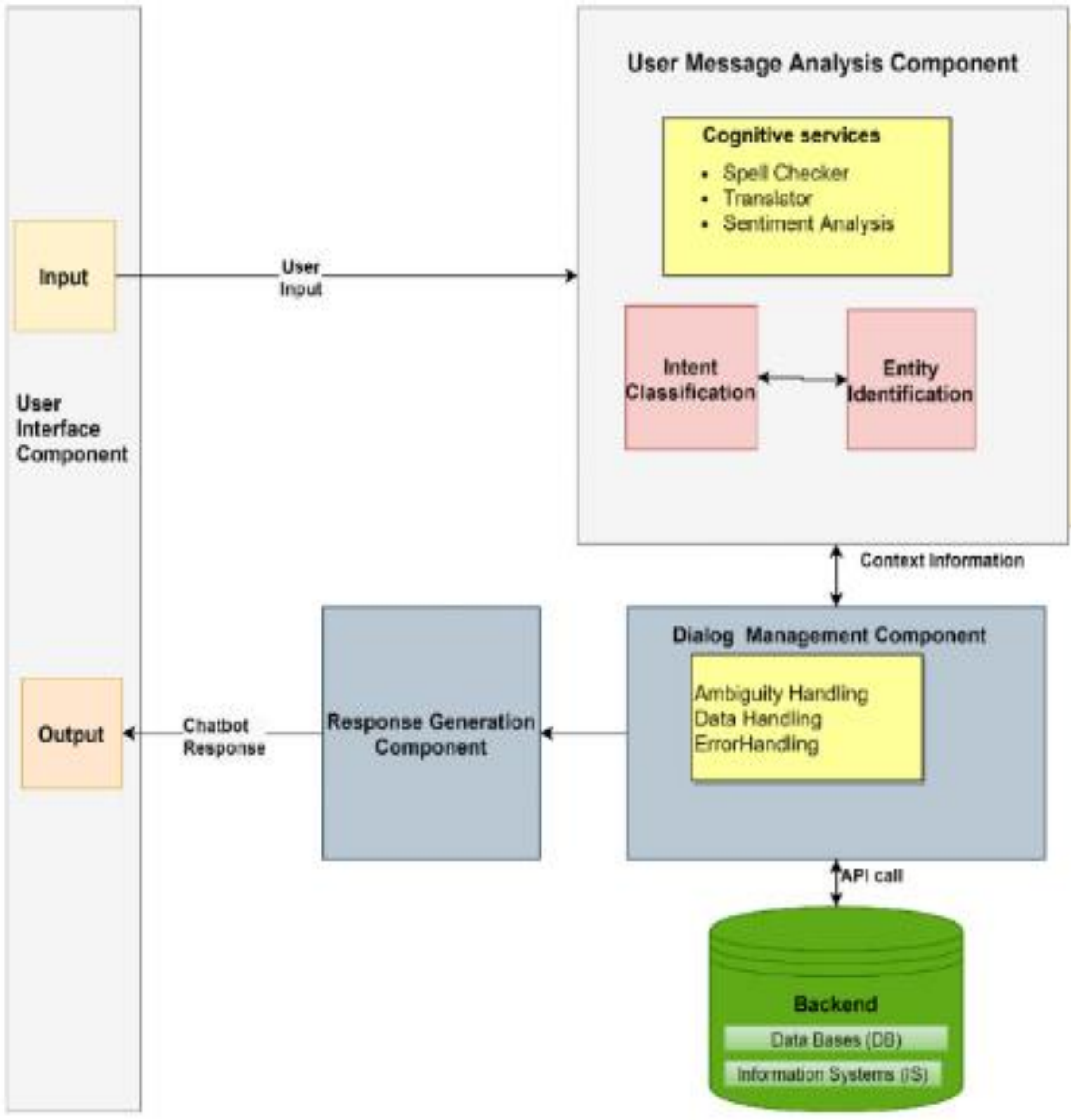


Fig 6: genral architecture for chatbot

3.3 DATAFLOW DIAGRAM LEVEL 0:

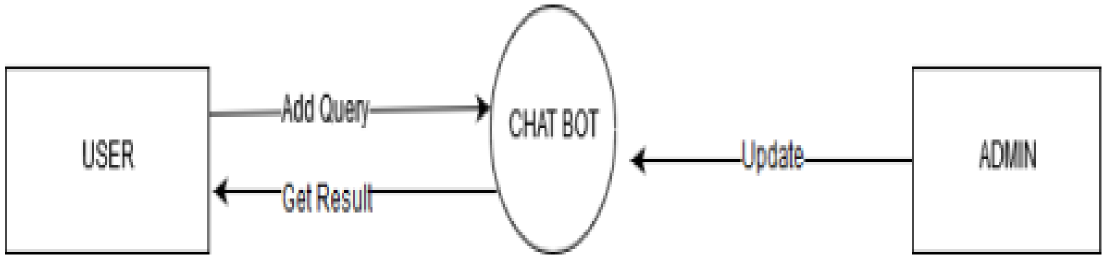


Fig 7 :Zero level diagram of DFD

3.4 DATAFLOW DIAGRAM LEVEL 1:

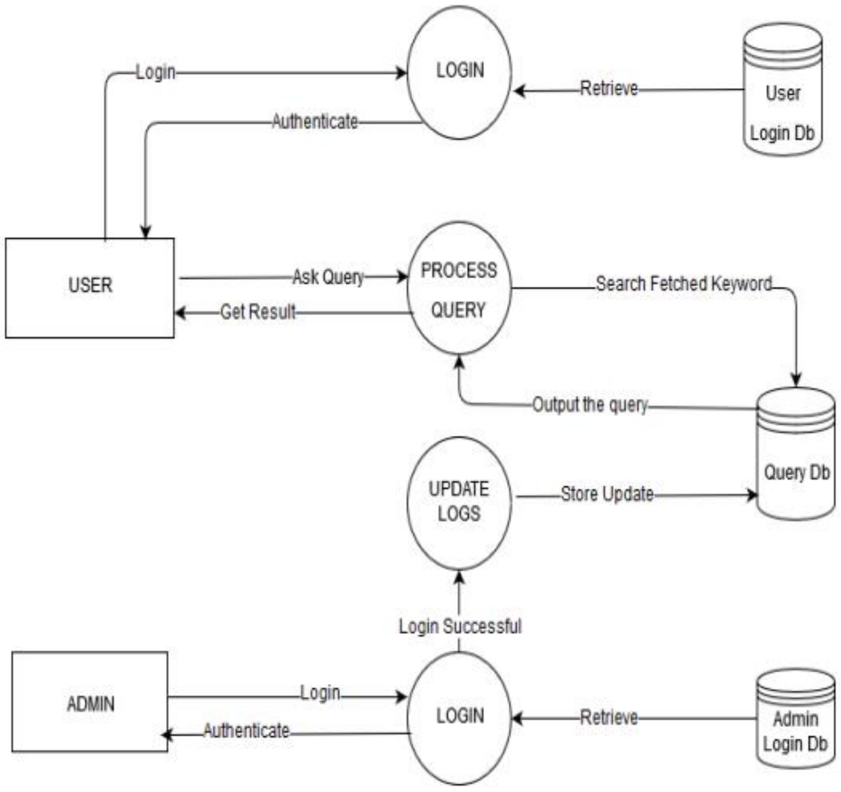


Fig 8: level one DFD diagram

3.5 DATAFLOW DIAGRAM LEVEL 2:

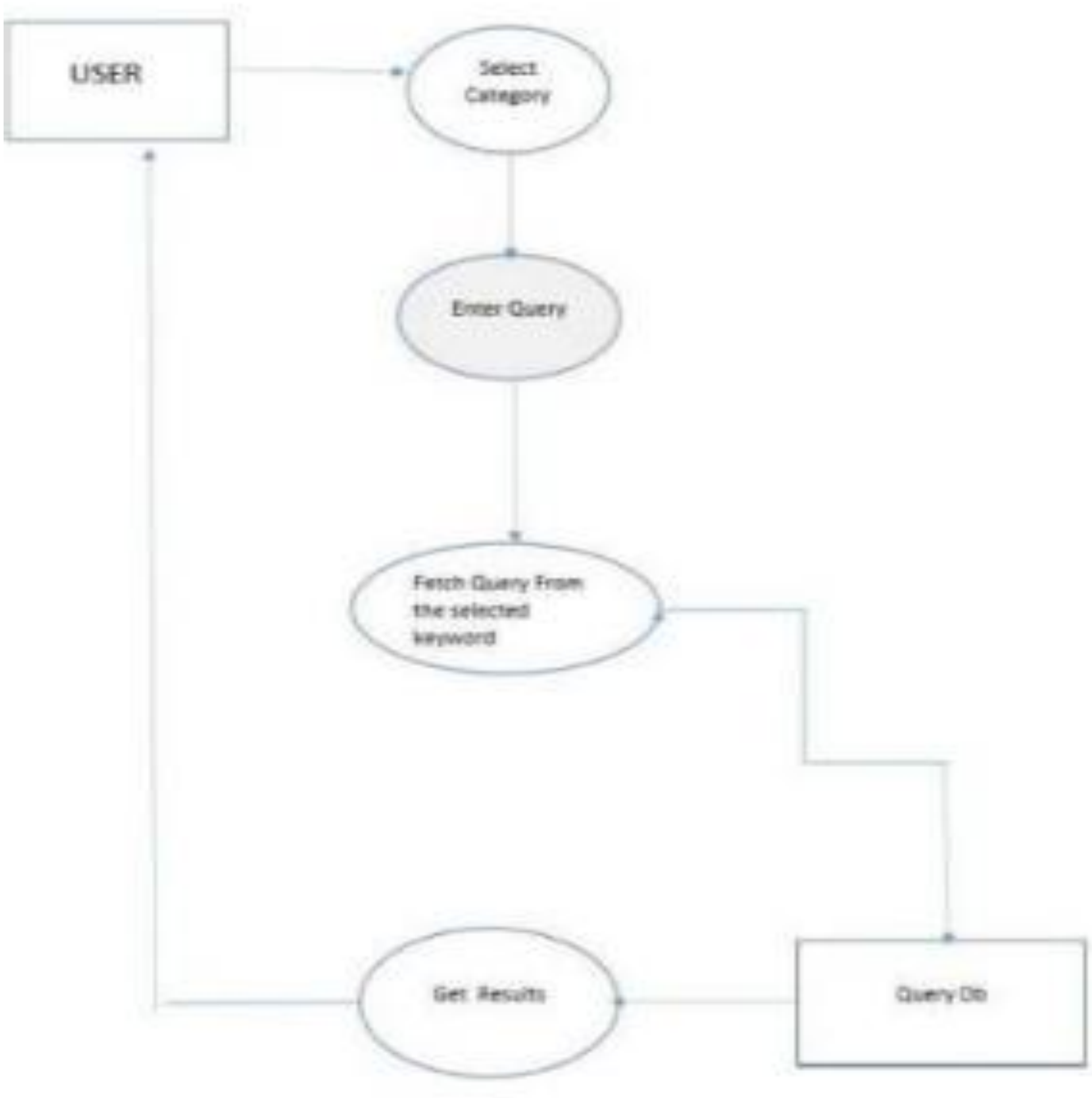


Fig 9: Level Two DFD Diagram

3.6 USECASE DIAGRAM FOR CHATBOT

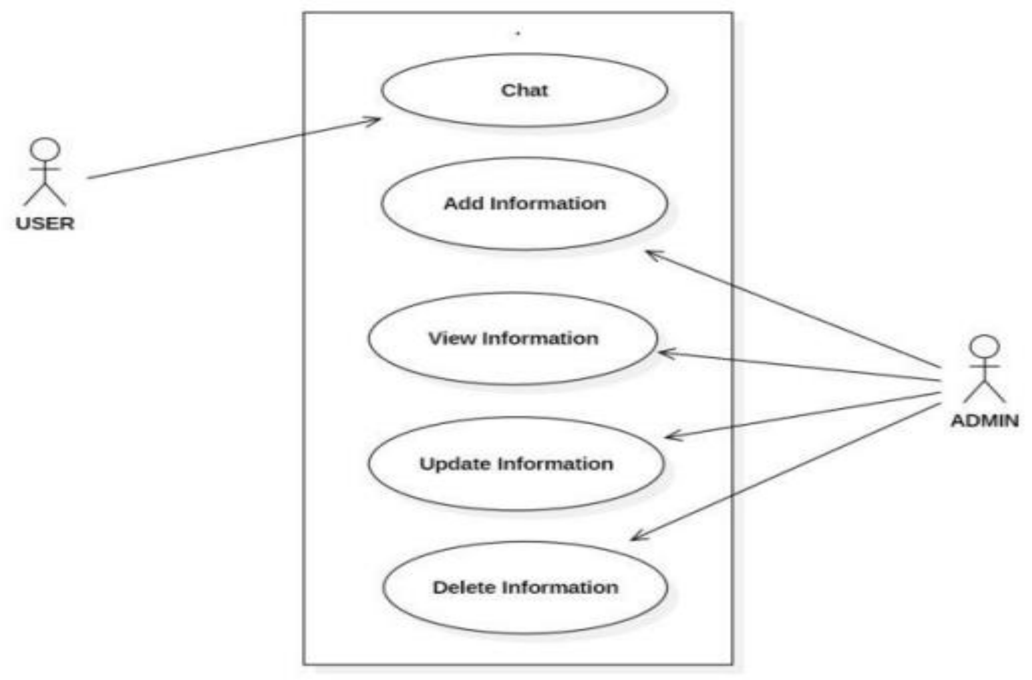


Fig 10 :Usecase diagram of user and admin roles

3.7 Usecase diagram for context identification

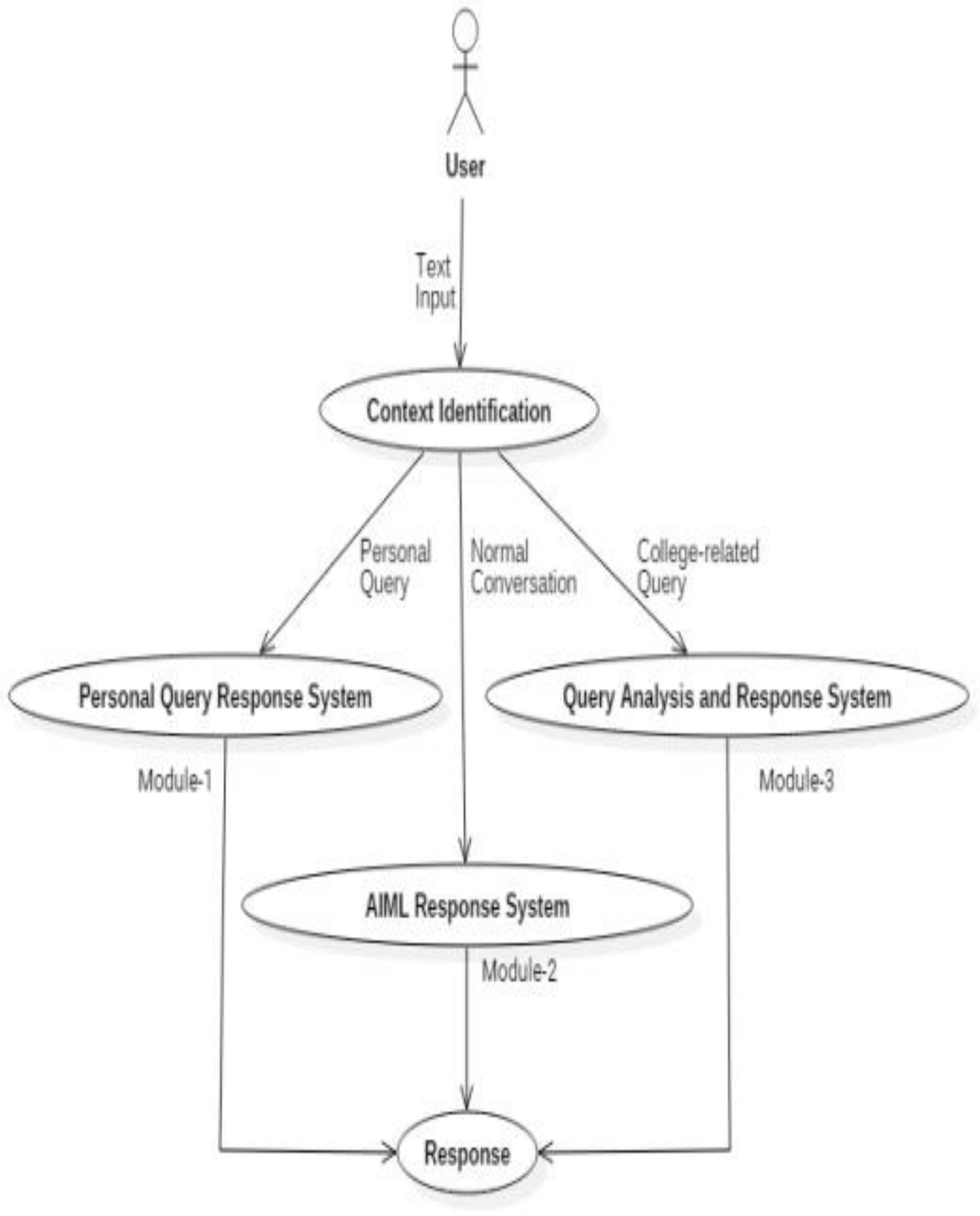


Fig 11: Usecase diagram for context identification

3.8 CLASS DIAGRAM FOR CHATBOT

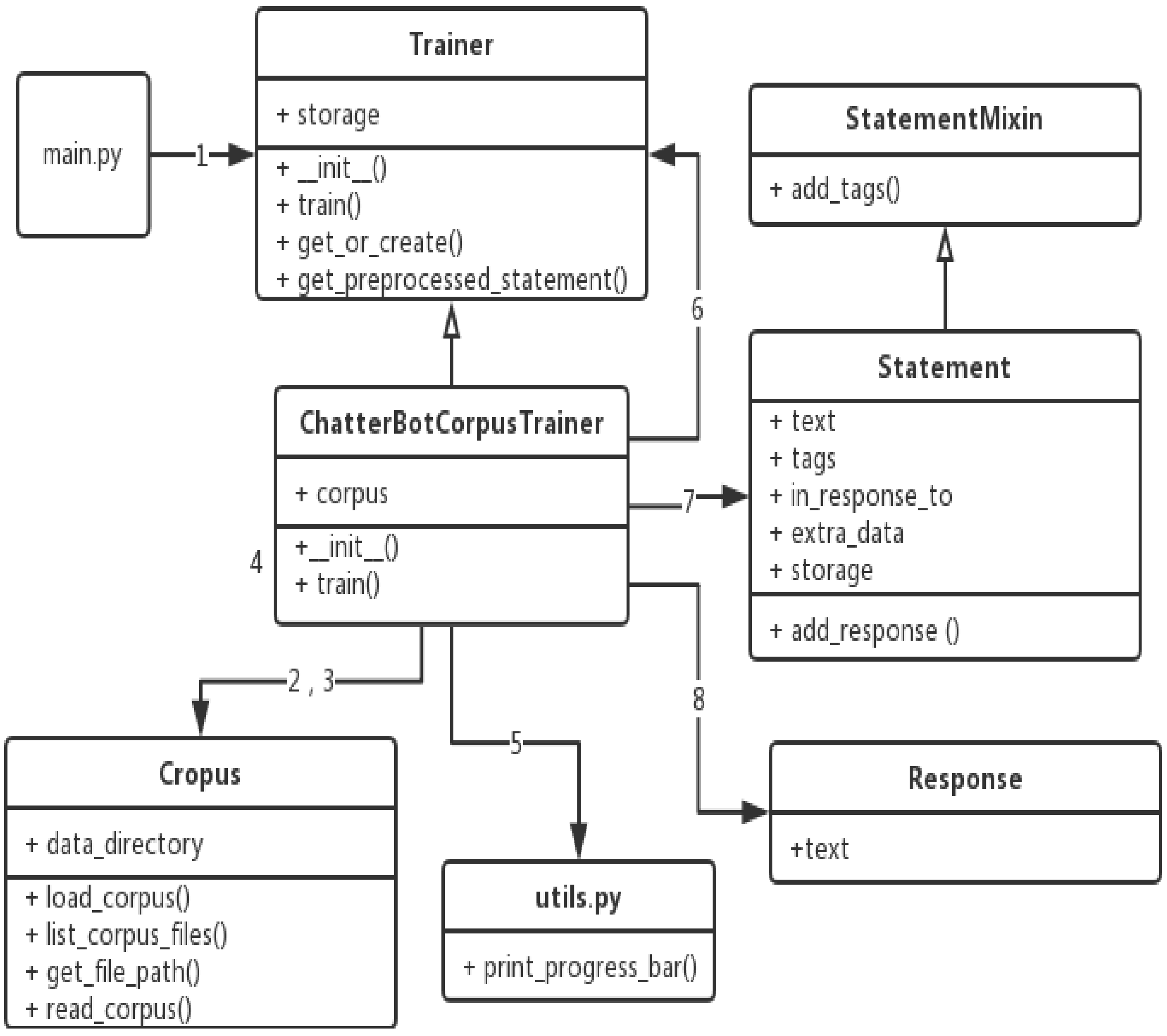


Fig 12: Class diagram for chatbot

3.9 Sequence diagram for chatbot

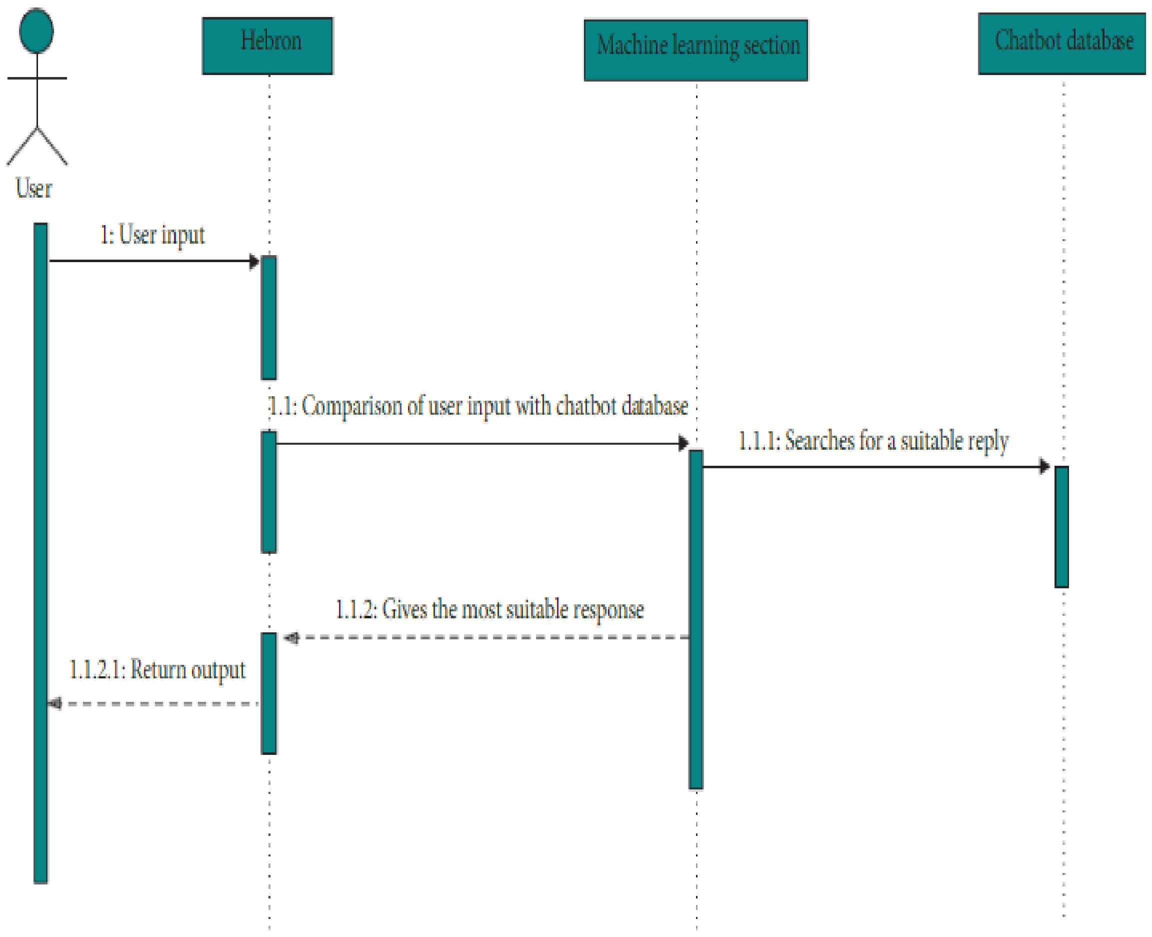

User
Hebron
Machine learning section
Chatbot database
1: User input
1.1: Comparison of user input with chatbot database
1.1.1: Searches for a suitable reply
1.1.2: Gives the most suitable response
1.1.2.1: Return output

Fig 13: sequence diagram for chatbot

4. Requirements Specification

4.1 Hardware Specification

To run the project required hardware specifications are

1.supported cpu models

i) intel core i7
ii) intel core i3
iii)Intel Core i5

2.Memory
i) Minimum 4gb RAM is required
3. CPU
i) 2x 64-bit 2.8GHZ 8.00
4.Disk storage
i)250 GB

4.2 Software Specification

1.supported operating system versions
i) Windows 8/8.1/10/11 (64-bit)
ii) macOS 10.15 (Catalina)
iii) Any 64-bit Linux distribution that supports Gnome, KDE, or Unity DE
2.Latest version of visual studio
3. python – v3 (version 3)
4. Rest API

5. IMPLEMENTATION

5.1 Python Implementation

```
from os import path
from tkinter import Tk, Canvas, Frame, Label, \
    ALL, Button, Entry, END, Scrollbar, N, S, E, W, LEFT, PhotoImage
from tkinter.constants import DISABLED, NORMAL, RIGHT
from threading import Thread, Event
from time import sleep



class ChatGUI:
    def __init__(self, callback, first_message="welcome to ChatBotAI",
terminate="quit"):
        """
        callback: (function) Bot callback function
        first_message: (Str) first string message show to user
        terminate: (str) string message shown user

        Initialize the tkinter window and start the mainloop.

        Canvas is used to create the window. Each message is created as
        """
        # media path for bot images
        self.data_path = path.join(path.dirname(path.dirname(path.abspath(__file__))),
"media")

        # instance args
        self.callback = callback
        self.terminate = terminate

        # initialize tkinter start
        self.root = Tk()
        self.root.title("Sample ChatBot")
        self.root.protocol("WM_DELETE_WINDOW", self.close_handler)
        # this removes the maximize
        self.root.resizable(0, 0)
        self.canvas = Canvas(self.root, width=800, height=500, bg="white")
        self.canvas.grid(row=0, column=0)
        self.canvas_scroll_y = Scrollbar(self.root, orient="vertical",
command=self.canvas.yview)
        self.canvas_scroll_y.grid(row=0, column=1, sticky=(N, S, W, E))
        self.canvas.configure(yscrollcommand=self.canvas_scroll_y.set)

        self.user_input_box = Entry(self.root)
        self.user_input_box.grid(row=1, column=0, padx=5, pady=10, ipady=8,
ipadx=290, sticky=W)
        self.user_input_box.bind("<Return>", self.user_input_handler)
        self.user_input_box.bind("<Shift_L><Return>", self.user_input_box_handler)
        send_image = PhotoImage(file=path.join(self.data_path, "send.png"))
        self.send_button = Button(self.root, image=send_image,
                                  command=lambda: self.user_input_handler(None))
        self.send_button.grid(row=1, column=0, sticky=E)

        self.bot_image = PhotoImage(file=path.join(self.data_path, "robot.png"))
        self.user_image = PhotoImage(file=path.join(self.data_path, "user.png"))
        # initialize tkinter stop

        # get the last bubble objects to move them up for next bubbles
        self.last_bubble = None

        # bubble thread objects
        self.user_thread = None
        self.bot_thread = None
        self.thread_event = Event()

        if first_message:
            self.add_bot_message(first_message)

        self.root.mainloop()

    def close_handler(self):
        """
        When the close button of MainWindow pressed we need to kill the active threads
        before closing the window.
        """
        if self.user_thread and self.user_thread.is_alive():
            self.bot_thread._tstate_lock.release_lock()
            self.user_thread._stop()
        if self.bot_thread and self.bot_thread.is_alive():
            self.bot_thread._tstate_lock.release_lock()
            self.bot_thread._stop()
        self.root.destroy()

    def show_bubble(self, message="", bot=True):
        """
        message: (str) Bubble message shown in canvas
        bot: (bool) Shown the bubble based on this value

        Add the bubble to canvas.

        Previous canvas are moved based on the last bubble height and new bubbles are
added.
        Color, arrow(draw_triangle), image(add_icon) and Position of the Bubble is
configures here

        UserBubble is added to right side of the canvas.
        BotBubble is added to left side of the canvas.
        """
        if self.last_bubble:
            self.canvas.move(ALL, 0, -(self.last_bubble.winfo_height() + 10))
        bg_color = "light blue" if bot else "light grey"
        color = "black"  # if bot else "white"
        frame = Frame(self.canvas, bg=bg_color)
        self.last_bubble = frame

        widget = self.canvas.create_window(50 if bot else 700, 440, window=frame,
anchor='nw' if bot else 'ne')

        chat_label = Label(frame, text=message, wraplength=600, justify=LEFT if bot
else RIGHT, font=("Helvetica", 12),
                           bg=bg_color, fg=color)
        chat_label.pack(anchor="w" if bot else 'e', side=LEFT if bot else RIGHT,
pady=10, padx=10)

        self.root.update_idletasks()
        self.canvas.create_polygon(self.draw_triangle(widget, bot), fill=bg_color,
outline=bg_color)
        self.add_icon(widget, bot)

    def add_icon(self, widget, bot=True):
        """
        Add the image to given widget.
        Currently this based bot and user bubble positions.

        If it's moved we need to update the x1 and y1 values.
        """
        x1, y1, x2, y2 = self.canvas.bbox(widget)
        if bot:
            self.canvas.create_image(x1 - 72, y2, image=self.bot_image, anchor=W)
        else:
            self.canvas.create_image(x2 + 72, y2, image=self.user_image, anchor=E)

    def draw_triangle(self, widget, bot=True):
        """
        Draw the triangles in the bubble widget.
        """
        x1, y1, x2, y2 = self.canvas.bbox(widget)
        if bot:
            return x1, y2 - 10, x1 - 10, y2, x1, y2
        return 700, y2 - 10, 700, y2, 710, y2

    def add_user_message(self, message):
        """
        create a user bubble and disable the input box until bot bubble is shown.
        Moreover, update the scroll location of the canvas
        """
        self.user_input_box.config(state=DISABLED)
        self.send_button.config(state=DISABLED)
        self.show_bubble(message, bot=False)
        # need to update canvas scroll region after content modified
        self.canvas.configure(scrollregion=self.canvas.bbox("all"))
        self.thread_event.set()

    def add_bot_message(self, message):
        """
        create a bot bubble and enable the input box after message shown in the canvas.
        Moreover, update the scroll location of the canvas
        """
        self.show_bubble(message, bot=True)
        self.user_input_box.config(state=NORMAL)
        self.send_button.config(state=NORMAL)
        # need to update canvas scroll region after content modified
        self.canvas.configure(scrollregion=self.canvas.bbox("all"))

    def process_message(self, message):
        """
        Call the bot handler and add the result to bot_message
        """
        bot_message = self.callback(message)
        while not self.thread_event.is_set():
            sleep(0.1)
        self.add_bot_message(bot_message)

    def user_input_handler(self, event):
        """
        User InputBox widget
        """
        message = self.user_input_box.get()

        if not message:
            return

        if message == self.terminate:
            self.close_handler()
            return

        self.thread_event.clear()
        self.user_thread = Thread(target=self.add_user_message, args=(message,))
        self.bot_thread = Thread(target=self.process_message, args=(message,))
        self.user_thread.start()
        self.bot_thread.start()

        self.user_input_box.delete(0, END)

    def user_input_box_handler(self, event):
        """
        Helper method to add the newline in the InputBox
        """
        self.user_input_box.insert(END, "\n")
```

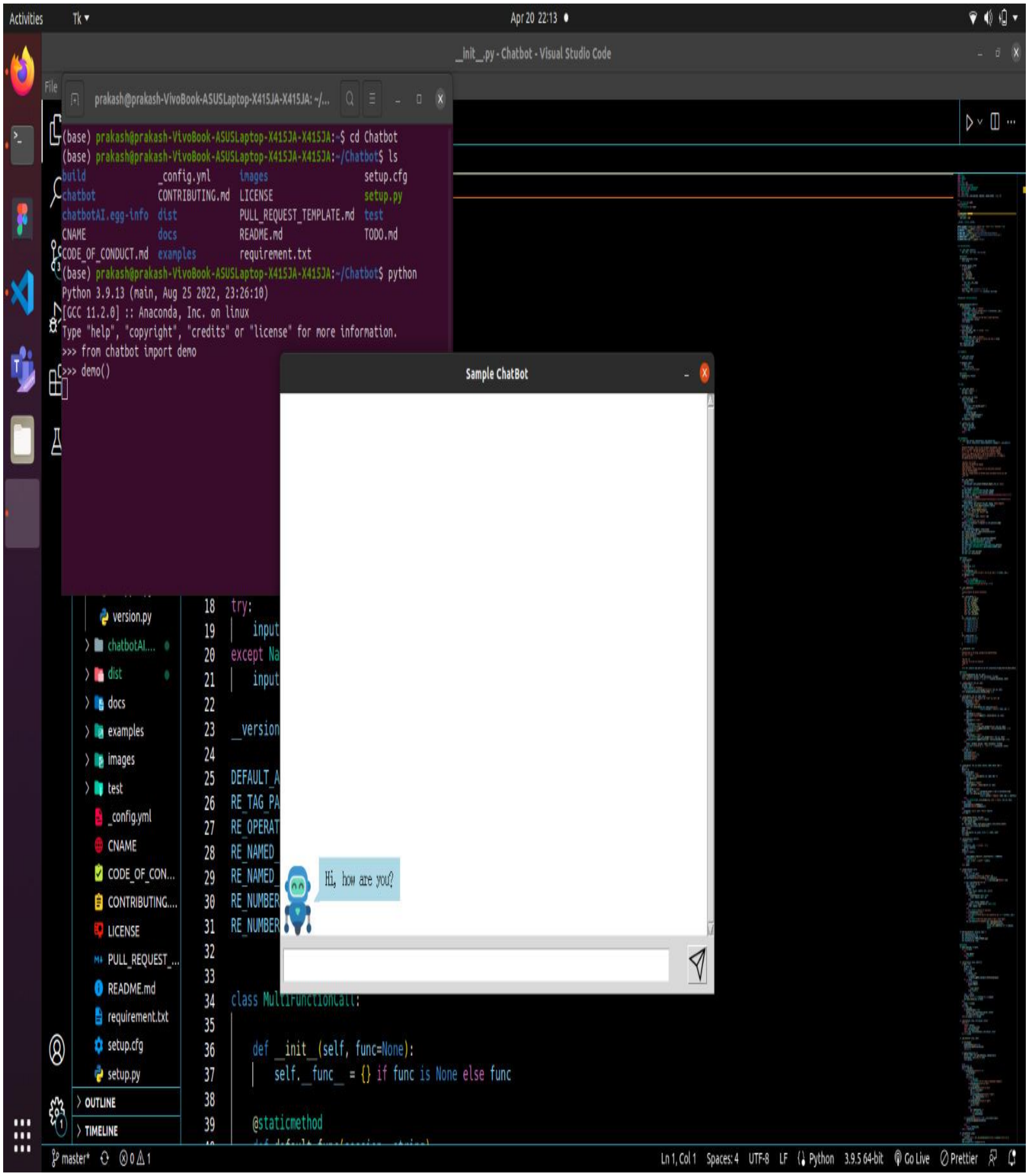
5.2 Mapper Class

```
class Session:

    def __init__(self, chat, session_id):
        self.__chat = chat
        self.session_id = session_id

    @property
    def conversation(self):
        return self.__chat._conversation[self.session_id]

    @conversation.setter
    def conversation(self, value):
        self.__chat._conversation[self.session_id] = value

    @property
    def memory(self):
        return self.__chat._memory[self.session_id]

    @memory.setter
    def memory(self, value):
        self.__chat._memory[self.session_id] = value

    @property
    def attr(self):
        return self.__chat._attr[self.session_id]

    @attr.setter
    def attr(self, value):
        self.__chat._attr[self.session_id] = value
    @property
    def topic(self):
        return self.__chat._topic[self.session_id]

    @topic.setter
    def topic(self, value):
        self.__chat._topic[self.session_id] = value


class Conversation(list):

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.__bot_message = []
        self.__user_message = []

    def append_bot_message(self, message):
        self.__bot_message.append(message)
        self.append(message)

    def append_user_message(self, message):
        self.__user_message.append(message)
        self.append(message)

    def get_bot_message(self, index):
        return self.__bot_message[index]

    def get_user_message(self, index):
        return self.__user_message[index]


class SessionHandler:

    def __init__(self, _class, **kwargs):
        self._class = _class
        self.__data = {key: _class(value) for key, value in kwargs.items()}

    def __getitem__(self, key):
        return self.__data[key]
```

```
def __setitem__(self, sender_id, val):
    self.__data[sender_id] = self._class(val)

def update(self, *args, **kwargs):
    data = dict(*args, **kwargs)
    for key, val in data.items():
        self.__data[key] = self._class(val)

def __delitem__(self, *args, **kwargs):
    return self.__data.__delitem__(*args, **kwargs)

def __contains__(self, *args, **kwargs):
    return self.__data.__contains__(*args, **kwargs)

def __iter__(self):
    return self.__data.__iter__()

def __len__(self):
    return self.__data.__len__()

def __repr__(self, *args, **kwargs):
    return self.__data.__repr__()

def __sizeof__(self, *args, **kwargs):
    return self.__data.__sizeof__()

def __str__(self, *args, **kwargs):
    return self.__data.__str__()

def clear(self):
    return self.__data.clear()

def copy(self):
    return SessionHandler(self._class, **self.__data)

def fromkeys(self, *args):
    return self.__data.fromkeys(*args)

def get(self, *args):
    return self.__data.get(*args)
```

```
def items(self):
    return self.__data.items()

def keys(self):
    return self.__data.keys()

def pop(self, *args):
    return self.__data.pop(*args)

def popitem(self):
    return self.__data.popitem()

def setdefault(self, *args, **kwargs):
    return self.__data.setdefault(*args, **kwargs)

def values(self):
    return self.__data.values()
```

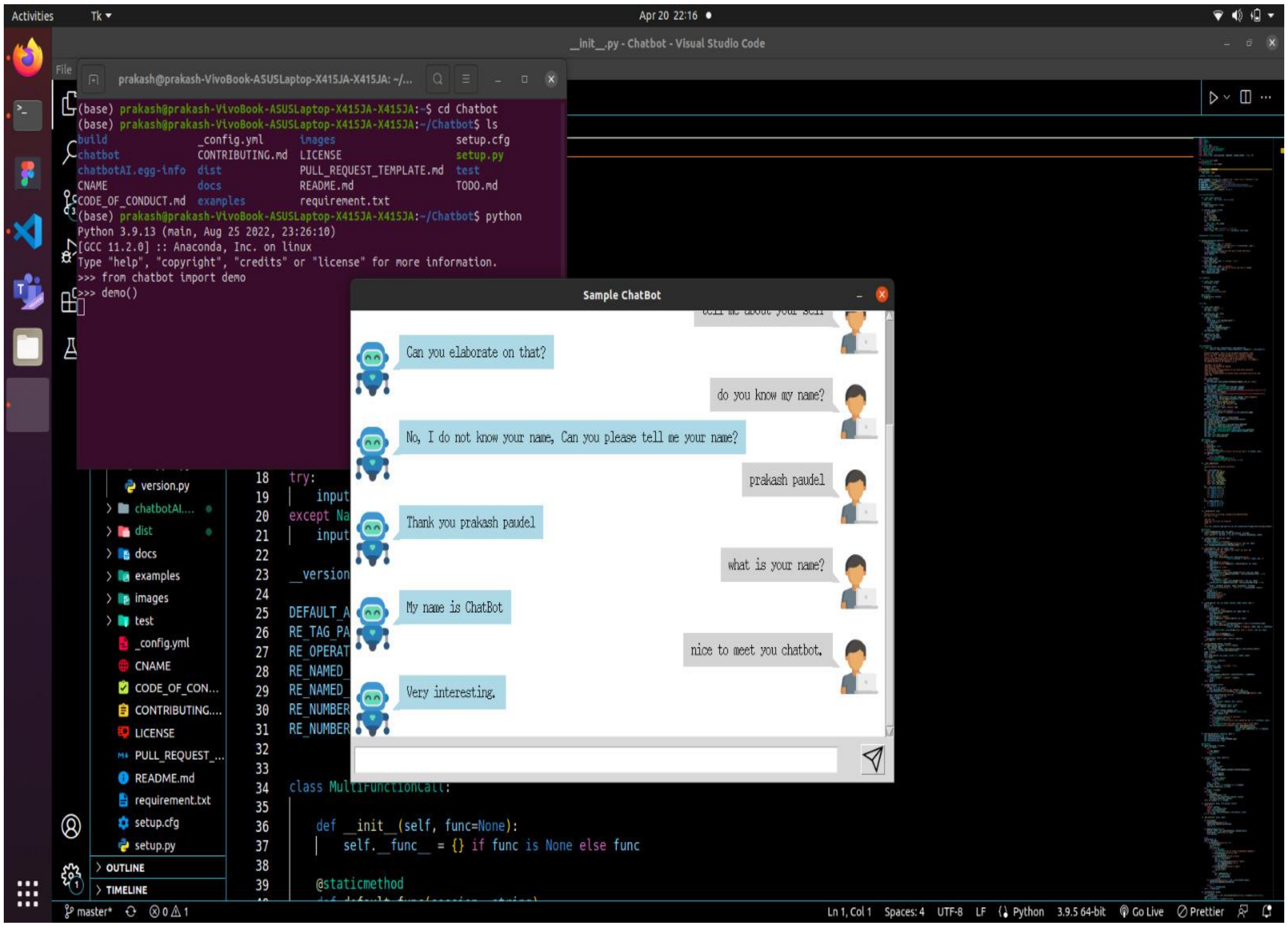
5.3 Output Screenshot



```
(base) prakash@prakash-VivoBook-ASUSLaptop-X415JA-X415JA:~$ cd Chatbot
(base) prakash@prakash-VivoBook-ASUSLaptop-X415JA-X415JA:~/Chatbot$ ls
build              _config.yml      images                    setup.cfg
chatbot            CONTRIBUTING.md  LICENSE                   setup.py
chatbotAI.egg-info dist             PULL_REQUEST_TEMPLATE.md  test
CNAME              docs             README.md                 TODO.md
CODE_OF_CONDUCT.md examples         requirement.txt
(base) prakash@prakash-VivoBook-ASUSLaptop-X415JA-X415JA:~/Chatbot$ python
Python 3.9.13 (main, Aug 25 2022, 23:26:10)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from chatbot import demo
>>> demo()
```

```
class MultiFunctionCall:

    def __init__(self, func=None):
        self.__func__ = {} if func is None else func

    @staticmethod
```

Fig 14: Output Screenshot



```
(base) prakash@prakash-VivoBook-ASUSLaptop-X415JA-X415JA:~$ cd Chatbot
(base) prakash@prakash-VivoBook-ASUSLaptop-X415JA-X415JA:~/Chatbot$ ls
build                _config.yml      images                    setup.cfg
chatbot              CONTRIBUTING.md  LICENSE                   setup.py
chatbotAI.egg-info   dist             PULL_REQUEST_TEMPLATE.md  test
CNAME                docs             README.md                 TODO.md
CODE_OF_CONDUCT.md   examples         requirement.txt
(base) prakash@prakash-VivoBook-ASUSLaptop-X415JA-X415JA:~/Chatbot$ python
Python 3.9.13 (main, Aug 25 2022, 23:26:10)
[GCC 11.2.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from chatbot import demo
>>> demo()
```

Sample ChatBot

Can you elaborate on that?

do you know my name?

No, I do not know your name, Can you please tell me your name?

prakash paudel

Thank you prakash paudel

what is your name?

My name is ChatBot

nice to meet you chatbot.

Very interesting.

```
    def __init__(self, func=None):
        self.__func__ = {} if func is None else func

    @staticmethod
```

Fig 15 : Output Screenshot

6. CONCLUSION

chatbot AI can be used for a variety of applications, such as customer service, e-commerce, and education. It can provide personalized recommendations, answer frequently asked questions, and assist with transactions. Overall, chatbot AI is a promising technology that can improve customer engagement and provide more efficient services. As the technology advances, we can expect to see more sophisticated and intelligent chatbots that can understand and respond to more complex user requestsIn the simplest of terms, an AI chatbot is a software program built and deployed with the help of conversational AI to assist users and make bot-human interactions, natural, engaging, and free-flowing. Additionally, chatbots can handle multiple conversations simultaneously, which can significantly reduce the response time for customers.

A) This chatbot can handle simple queries,

B) Chatbot AI has its ability to operate 24/7 without any breaks or time off, which makes it a cost-effective solution for businesses.

C) It has the potential to improve customer service and engagement, but it should be used in conjunction with human interaction to provide the best possible customer experience.

7 . REFERENCE

[1] Yuhua Li, David McLean, Zuhair A. Bandar, James D. O'Shea, Keeley Crockett, "Sentence Similarity Based on Semantic Nets and Corpus Statistics", IEEE Transactions on Knowledge and Data Engineering, Volume 18 - No. 8, August 2006.

[2] Emanuela Haller, Traian Rebedea, "Designing a Chat-bot that Simulates an Historical Figure", IEEE Conference Publications, July 2013.

[3] Pratik Slave, Vishruta Patil, Vyankatesh Gaikwad, Girish Wadhwa, "College Enquiry Chat Bot", International Journal on Recent and Innovation Trends in Computing and Communication, Volume 5, Issue 3, March 2015.

[4] "AIML Based Voice Enabled Artificial Intelligent Chatterbot", International Journal of u- and e- Service, Science and Technology Volume 8 - No. 2, 2015.

[5] Amey Tiwari, Rahul Talekar, Prof. S. M. Patil, "College Information Chatbot System", International Journal of Engineering Research and General Science, Volume 2, Issue 2, April 2017.

[6] Rachit Kulkarni, Ankit Methwani, Nakul Pawar, Charmi Valecha, Pooja Shetty, "College Chat-bot", International Journal of Advanced Research in Computer Engineering & Technology, Volume 6, Issue 4, April 2017.

[7] Chaitrali S. Kulkarni, Amruta U. Bhavsar, Savita R. Pingale, Prof. Satish S. Kumbhar, "BANK CHATBOT - An Intelligent Assistant System Using NLP and Machine Learning", International Research Journal of Engineering and Technology, Volume 4, Issue 5, May 2017.

[8] Prof. K. Bala, Mukesh Kumar, Sayali Hulawale, Sahil Pandita, "Chat-Bot For College Management System Using A.I", International Research Journal of Engineering and Technology, Volume 4, Issue 11, Nov 2017