CONTENTS

CONTENTS

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Introduction to Voice Assistants

Voice assistants are artificial intelligence (AI) systems that enable users to interact with devices and perform tasks using natural language voice commands. Voice assistants have become increasingly popular in recent years, with many people using them to control smart devices, access information, and perform a variety of tasks on their smartphones, smart speakers, and other devices.

Voice assistants use natural language processing (NLP) algorithms and machine learning techniques to understand and respond to user requests. They can be activated using a specific trigger word or phrase, such as "Hey Siri" or "Ok Google," and can perform a wide range of tasks, such as answering questions, setting reminders, playing music, or controlling smart home devices.

Voice assistants have the potential to make many everyday tasks more convenient and efficient, as they allow users to interact with devices and systems using their voice rather than requiring them to use a physical interface or input commands manually. However, voice assistants also raise privacy and security concerns due to the sensitive personal data that they may collect, store, and process.

Overall, voice assistants are an emerging and rapidly evolving technology that has the potential to transform how people interact with devices and systems, and they will likely continue to play an important role in the development of AI and the internet of things (IoT).

1.2 Aim and purpose

Purpose of virtual assistant is to being capable of voice interaction, music playback, making to-do lists, setting alarms, streaming podcasts, playing audiobooks, and providing weather, traffic, sports, and other real-time information, such as news. Virtual assistants enable users to speak natural language voice commands in order to operate the device and its apps.

There is an increased overall awareness and a higher level of comfort demonstrated specifically by millennial consumers. In this ever-evolving digital world where speed, efficiency, and convenience are constantly being optimized, it’s clear that we are moving towards less screen interaction.

CHAPTER 2

LITERATURE SURVEY

2.1 Survey on Virtual Assistant: Google Assistant, Siri, Cortana, Alexa

Authors: Amrita S. Tulshan and Sudhir Namdeorao Dhage

Virtual assistant is boon for everyone in this new era of 21st century. It has paved way for a new technology where we can ask questions to machine and can interact with IVAs as people do with humans. This new technology attracted almost whole world in many ways like smart phones, laptops, computers etc. Some of the significant VPs are like Siri, Google Assistant, Cortana, and Alexa. Voice recognition, contextual understanding and human interaction are the issues which are not solved yet in this IVAs. So, to solve those issues100 users participated a survey for this research and shared their experiences. All users' task was to ask questions from the survey to all personal assistants and from their experiences this research paper came up with the actual results. According to that results many services were covered by these assistants but still there are some improvements required in voice recognition, contextual under-standing and hand free interaction. After addressing these improvements in IVAs will definitely increased its use is the main goal for this research paper.

2.2 Survey On Smart Virtual Voice Assistant

Authors: Manjusha Jadhav, Krushna Kalyankar, Gnaesh Narkhede and Swapnil Kharose

In this modern era, day to day life became smarter & interlinked with technology. We already know some voice assistant like Google, Siri etc. Now in our voice assistant system, it can act as your smart friend, daily schedule manager, to do writer, calculator & search tool. This project works on speech input & give output through speech & text on screen. This assistant attaches with the world wide web to provid result that the user required. Natural language processing algorithm helps machines to engage in communication using natural human language in many forms.

2.3 Survey on Personal Voice Assistant

Authors: S. Lahari, A. Naveen, G. Sarath Chandra

Digitization brings new possibilities to ease our daily life activities by the means of assistive technology. Amazon Alexa, Apple Siri, Microsoft Cortana, Samsung Bixby, to name only a few were successful in the age of smart personal assistants (spas).A voice assistant is defined a digital assistant that combines artificial intelligence, machine learning Speech Recognition, Natural Language Processing (NLP), Speech Synthesis and various actuation mechanisms to sense and influence the environment. We use different NLP techniques to convert Speech to text (STT), then process the text, convert Text to Speech (TTS), add various functionalities. However, SPA research seems to be highly fragmented among different disciplines, such as computer science, human-computer-interaction and information systems, which leads to 'reinventing the wheel approaches' and thus impede progress and conceptual clarity. In this paper, we present an exhaustive, integrative literature review to build a solid basis for future research. Hence, we contribute by providing a consolidated, integrated view on prior research and lay the foundation for an SPA classification scheme.

2.4 Survey on Personal Desktop Virtual Voice Assistant using Python

Authors: Prof. Suresh V. Reddy, Chandresh Chhari, Prajwal Wakde, Nikhil Kamble

In today's develop generation, How cool is it to build your own personal assistants like Alexa or Siri? It's not very complex and may be effortlessly performed in Python. Personal virtual assistants are capturing numerous attentions lately. Chat bots are not unusual in maximum business web sites. The predominant agenda of our voice help makes human beings clever and supply immediate and computed effects. The fundamental mission of a voice assistant is to reduce using enter gadgets like keyboard, mouse, touch pens, and so forth. This will lessen both the hardware fee and space taken by it.

2.5 Smart Home Voice Assistants: A Literature Survey of User Privacy and Security Vulnerabilities

Authors: Prof. Suresh V. Reddy, Chandresh Chhari, Prajwal Wakde, Nikhil Kamble

Intelligent voice assistants are internet-connected devices, which listen to their environment and react to spoken user commands in order to retrieve information from the internet, control appliances in the household, or notify the user of incoming messages, reminders, and the like. With their increasing ubiquity in smart homes, their application seems only limited by the imagination of developers, who connect these off-the-shelf devices to existing apps, online services, or appliances. However, since their inherent nature is to observe the user in their home, their ubiquity also raises concern of security and user privacy. To justify the trust placed into the devices, the devices must be secure from unauthorized access and the back-end infrastructure tasked with speech-to-text analysis, command interpretation, and connection to other services and appliances must maintain confidentiality of data. To investigate existing possible vulnerabilities, approaches to mitigate them, as well as general considerations in this emerging field, we supplement the findings of a recent study with results from a systematic literature review. We were able to compile a list of six main types of user privacy vulnerabilities, partially confirming previous findings, but also finding additional issues. We discuss these vulnerabilities, their associated attack vectors, and possible mitigations users can take to protect themselves.

CHAPTER 3

EXISTING SYSTEM

3.1 Overview of existing system

A virtual voice assistant is a software program that utilizes natural language processing and voice recognition technologies to understand and respond to spoken commands and queries. It allows users to interact with their devices, applications, and services using voice commands, and can perform a wide range of tasks such as making phone calls, scheduling appointments, setting reminders, and providing information. Some popular examples of virtual voice assistants include Amazon Alexa, Google Assistant, and Apple Siri. These AI-powered systems can be integrated with other devices and services to create a more seamless and convenient user experience.

3.2 Disadvantages of existing system

1. **Performance**: Voice assistants may have limitations in terms of their performance, such as the speed at which they can process and respond to user requests, or the complexity of tasks that they can handle.
2. **Privacy and security**: Voice assistants may not always clearly communicate their data collection and sharing practices to users, which could raise concerns about transparency and consent.
3. **Customization**: Voice assistants may not offer users a high degree of customization or control over their functionality, which could limit their usefulness and appeal to users.
4. **Accuracy**: Voice assistants may not always accurately understand or respond to user requests and queries, which can lead to frustration and a poor user experience.
5. **Capabilities**: Voice assistants may not support all tasks or functions that users may want to perform, and they may not be able to integrate with all devices or systems.

CHAPTER 4

PROPOSED SYSTEM

4.1 Features

- It can get some real time information such as news headlines, weather report, IP address, Internet speed, and system stats.
- It can also get entertaining contents such as jokes, latest movies or TV series, and playing songs and videos in YouTube.
- It can also generate an image from given text and can also send an email.
- It can perform system operations such as opening/closing/switching tabs, copying/pasting/deleting/selecting the text, creating a new file, taking screen shot, minimizing/maximizing/switching/closing windows.
- It can also get brief information on any topic, perform arithmetic operations, and answer any general knowledge question.
- It can perform google search, find map or distance between two places on google maps.
- We can also get the chat history along with date & time of the query.
- It can also open any installed app and some websites, we can also take notes with help of assistant.

4.2 Advantages of Proposed System

- It can create an image from the given text.
- Send email to specified mail id by dictating the message.
- It is free of cost.
- Modification can be done quickly and easily.
- It maintains privacy as user data is not shared with third parties.

4.3 System Architecture

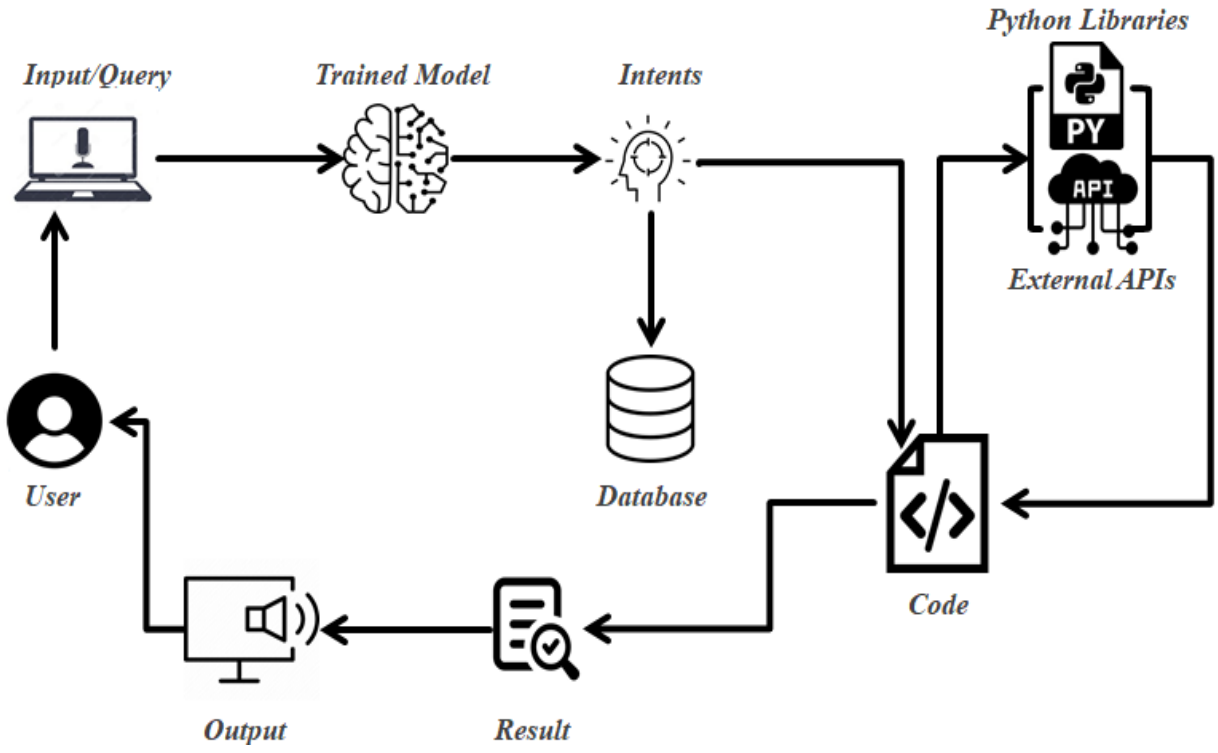


Figure 4.4.1 System Architecture (Voice Assistant)

CHAPTER 5

SYSTEM DESIGN

5.1 Requirement Analysis

In order to effectively design and develop a system, it is important to understand and document the requirements of the system. The process of gathering and documenting the requirements of a system is known as requirement analysis. It helps to identify the goals of the system, the stakeholders and the constraints within which the system will be developed. The requirements serve as a blueprint for the development of the system and provide a reference point for testing and validation.

- **Hardware Requirements**
  - Processor – 2.3 GHz or more
  - RAM – 4 GB or more
  - Disk Space – 50 GB or more
  - Input Devices – Microphone & Keyboard
  - Output Devices – Speaker & Monitor
  - Internet Connection

- **Software Requirements**
  - Python 3.9 or later
  - Python packages
    - SpeechRecognition==3.8.1
    - tensorflow==2.10.0
    - Keras==2.10.0
    - scikit-learn==1.1.2
  - APIs
    - News API
    - WolframAlpha API
    - OpenWeatherMap API
    - TMDB API
    - DreamStudio API

5.2 Data Flow Diagram

- **Level 0 data flow diagram**

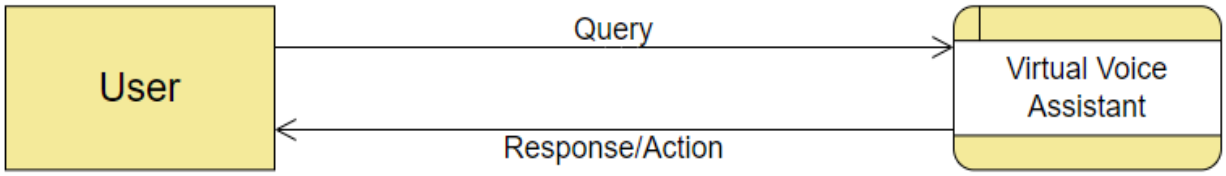


Figure 5.2.1 Level 0 Data Flow Diagram

- **Level 1 data flow diagram**

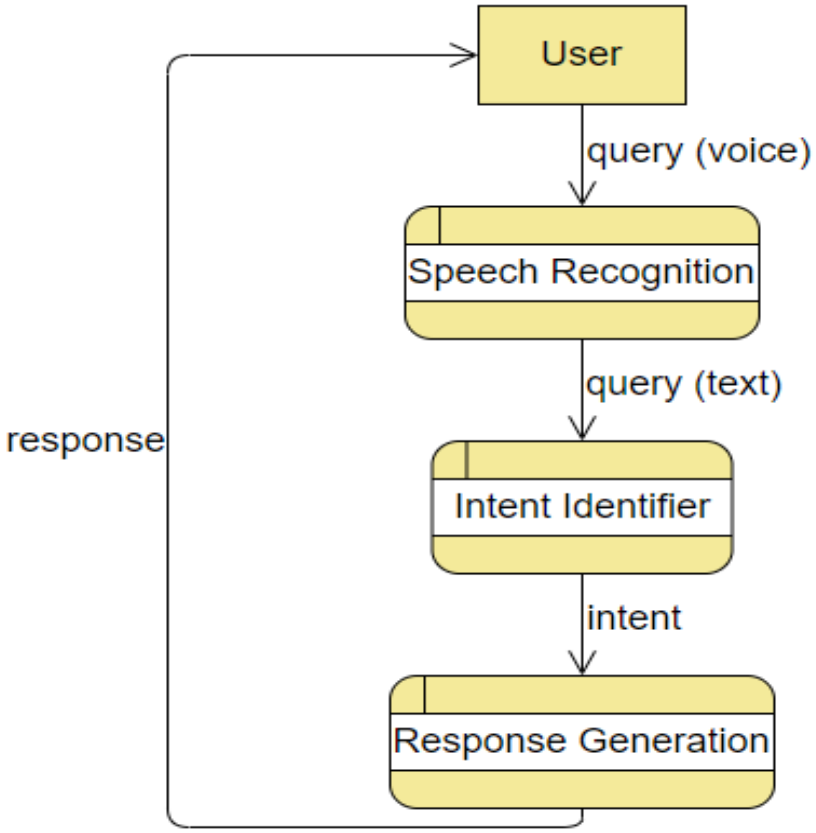


Figure 5.2.2 Level 1 Data Flow Diagram

- **Level 2 data flow diagram**

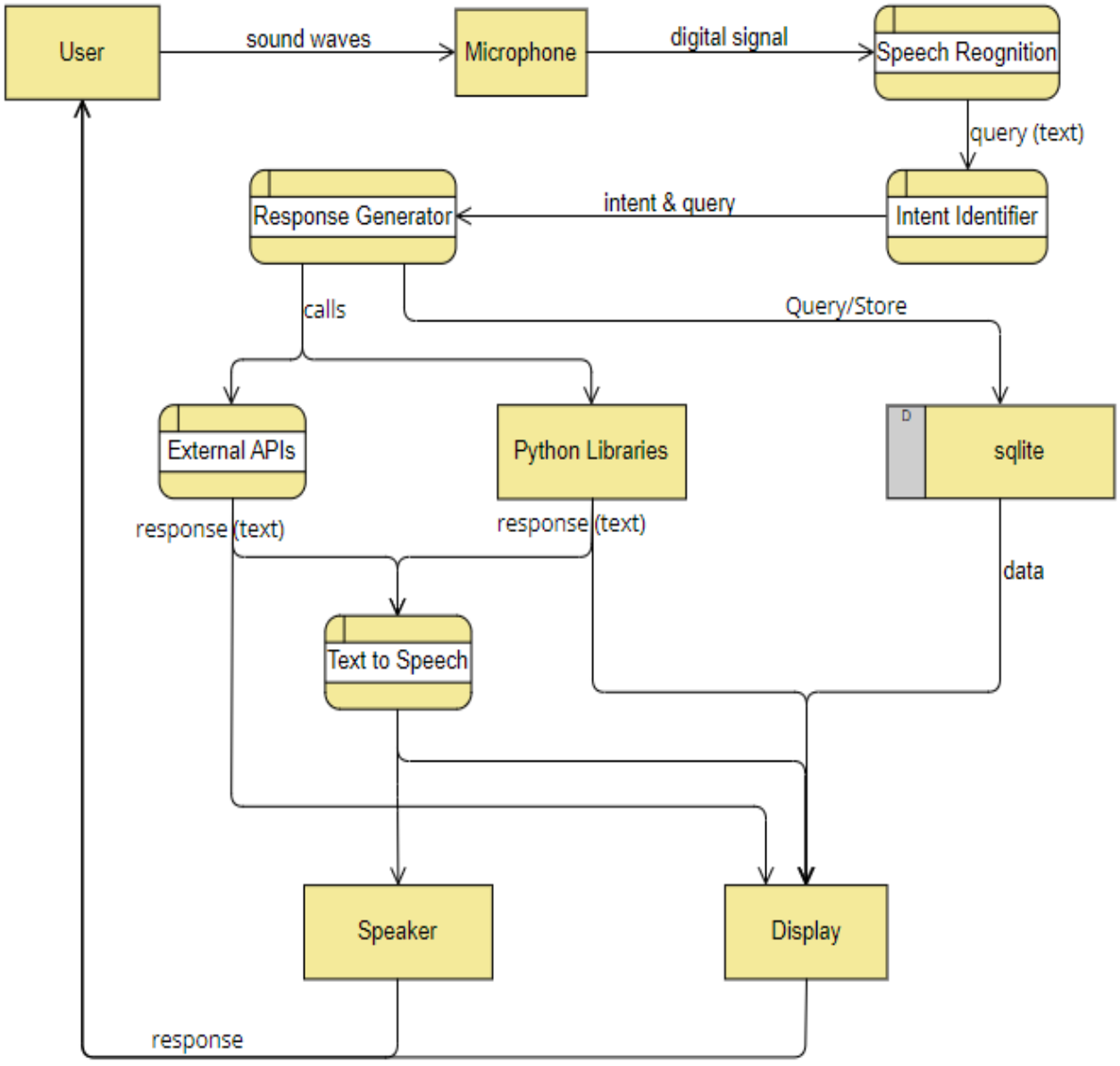


Figure 5.2.3 Level 2 Data Flow Diagram

5.3 ER Diagram

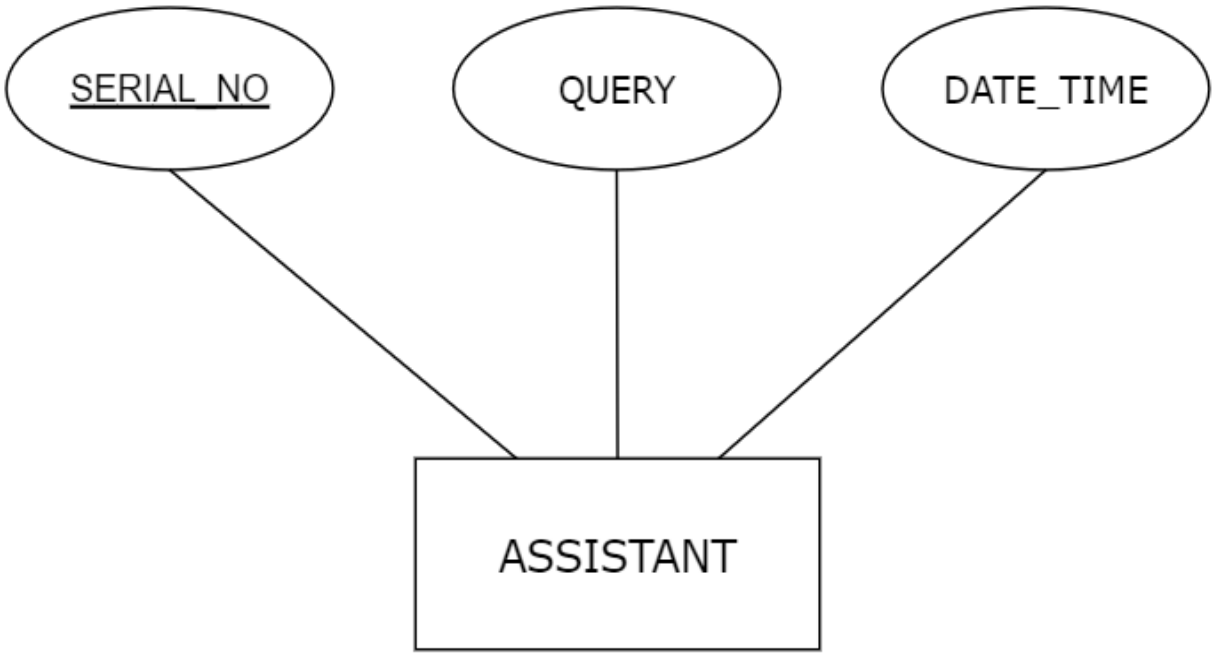


Figure 5.3.1 ER Diagram (Voice Assistant)

5.4 Use Case Diagram

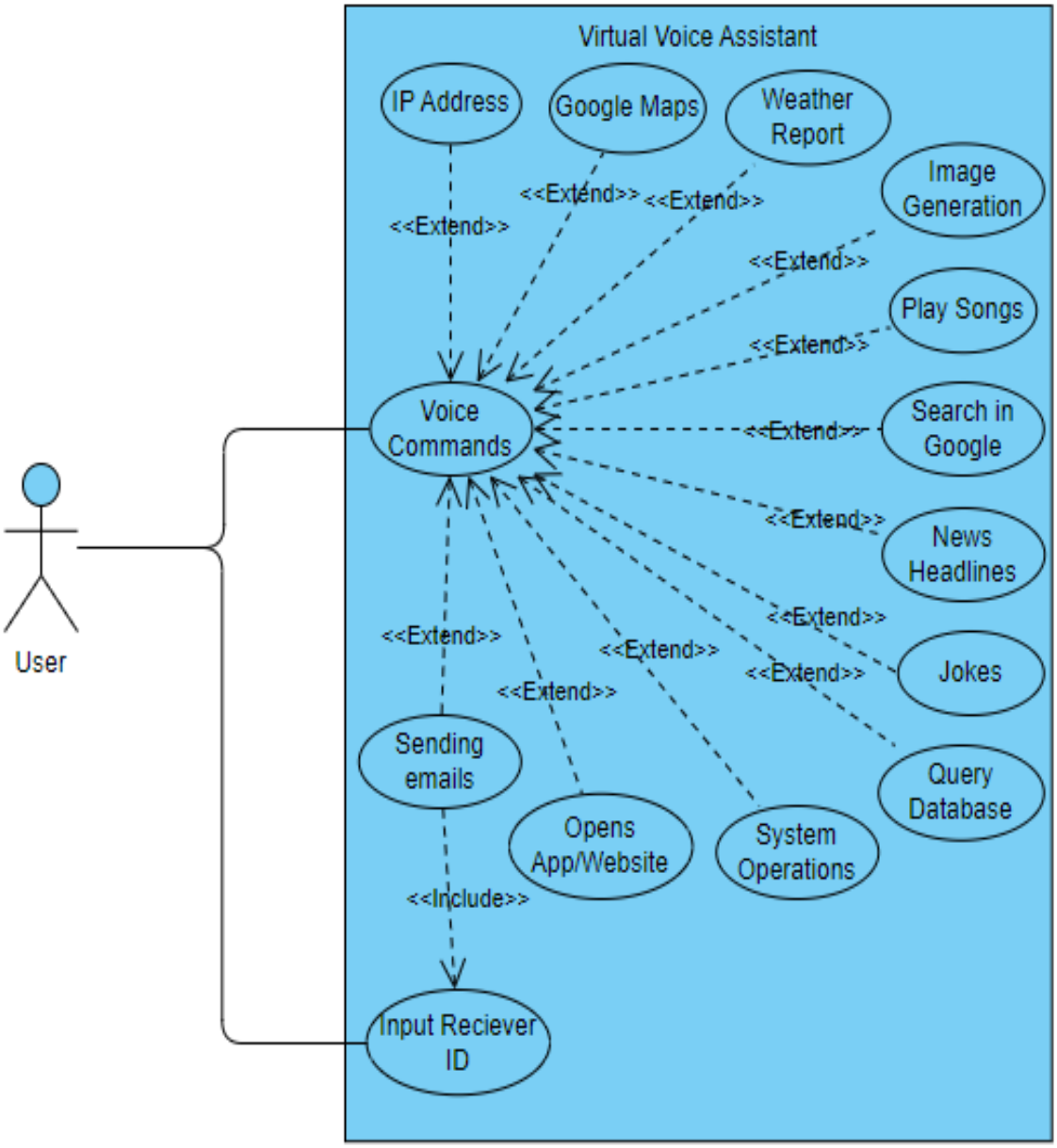


Figure 5.4.1 Use Case Diagram (Voice Assistant)

CHAPTER 6

CODING

main.py

```
try:
    # importing prebuilt modules
    import os
    import logging
    import pyttsx3
    logging.disable(logging.WARNING)
    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # disabling warnings
for gpu requirements
    from keras_preprocessing.sequence import pad_sequences
    import numpy as np
    from keras.models import load_model
    from pickle import load
    import speech_recognition as sr
    import sys
    sys.path.insert(0, os.path.expanduser('~') +
"/PycharmProjects/Virtual_Voice_Assistant")
    # sys.path.insert(0,
os.path.expanduser('~')+"/Virtual_Voice_Assistant") # adding voice
assistant directory to system path
    # importing modules made for assistant
    from database import *
    from image_generation import generate_image
    from gmail import send_email
    from API_functionalities import *
    from system_operations import *
    from browsing_functionalities import *
except (ImportError, SystemError, Exception, KeyboardInterrupt) as
e:
    print("ERROR OCCURRED WHILE IMPORTING THE MODULES")
    exit(0)

recognizer = sr.Recognizer()

engine = pyttsx3.init()
engine.setProperty('rate', 185)

sys_ops = SystemTasks()
tab_ops = TabOpt()
win_ops = WindowOpt()

# load trained model
```

```
model = load_model('..\\Data\\chat_model')

# load tokenizer object
with open('..\\Data\\tokenizer.pickle', 'rb') as handle:
    tokenizer = load(handle)

# load label encoder object
with open('..\\Data\\label_encoder.pickle', 'rb') as enc:
    lbl_encoder = load(enc)

def speak(text):
    print("ASSISTANT -> " + text)
    try:
        engine.say(text)
        engine.runAndWait()
    except KeyboardInterrupt or RuntimeError:
        return

def chat(text):
    # parameters
    max_len = 20
    while True:
        result =
model.predict(pad_sequences(tokenizer.texts_to_sequences([text]),

truncating='post', maxlen=max_len), verbose=False)
        intent =
lbl_encoder.inverse_transform([np.argmax(result)])[0]
        return intent

def record():
    with sr.Microphone() as mic:
        recognizer.adjust_for_ambient_noise(mic)
        recognizer.dynamic_energy_threshold = True
        print("Listening...")
        audio = recognizer.listen(mic)
        try:
            text = recognizer.recognize_google(audio,
language='us-in').lower()
        except:
            return None
    print("USER -> " + text)
    return text

def listen_audio():
    try:
        while True:
            response = record()
            if response is None:
```

```
                continue
            else:
                main(response)
    except KeyboardInterrupt:
        return

def main(query):
        add_data(query)
        intent = chat(query)
        done = False
        if ("google" in query and "search" in query) or ("google"
in query and "how to" in query) or "google" in query:
            googleSearch(query)
            return
        elif ("youtube" in query and "search" in query) or "play"
in query or ("how to" in query and "youtube" in query):
            youtube(query)
            return
        elif "distance" in query or "map" in query:
            get_map(query)
            return
        if intent == "joke" and "joke" in query:
            joke = get_joke()
            if joke:
                speak(joke)
                done = True
        elif intent == "news" and "news" in query:
            news = get_news()
            if news:
                speak(news)
                done = True
        elif intent == "ip" and "ip" in query:
            ip = get_ip()
            if ip:
                speak(ip)
                done = True
        elif intent == "movies" and "movies" in query:
            speak("Some of the latest popular movies are as
follows :")
            get_popular_movies()
            done = True
        elif intent == "tv_series" and "tv series" in query:
            speak("Some of the latest popular tv series are as
follows :")
            get_popular_tvseries()
            done = True
        elif intent == "weather" and "weather" in query:
            city = re.search(r"(in|of|for) ([a-zA-Z]*)", query)
            if city:
```

```
                city = city[2]
                weather = get_weather(city)
                speak(weather)
            else:
                weather = get_weather()
                speak(weather)
            done = True
        elif intent == "internet_speedtest" and "internet" in
query:
            speak("Getting your internet speed, this may take some
time")
            speed = get_speedtest()
            if speed:
                speak(speed)
                done = True
        elif intent == "system_stats" and "stats" in query:
            stats = system_stats()
            speak(stats)
            done = True
        elif intent == "image_generation" and "image" in query:
            speak("what kind of image you want to generate?")
            text = record()
            speak("Generating image please wait..")
            generate_image(text)
            done = True
        elif intent == "system_info" and ("info" in query or
"specs" in query or "information" in query):
            info = systemInfo()
            speak(info)
            done = True
        elif intent == "email" and "email" in query:
            speak("Type the receiver id : ")
            receiver_id = input()
            speak("Tell the subject of email")
            subject = record()
            speak("tell the body of email")
            body = record()
            success = send_email(receiver_id, subject, body)
            if success:
                speak('Email sent successfully')
            else:
                speak("Error occurred while sending email")
            done = True
        elif intent == "select_text" and "select" in query:
            sys_ops.select()
            done = True
        elif intent == "copy_text" and "copy" in query:
            sys_ops.copy()
            done = True
```

```
        elif intent == "paste_text" and "paste" in query:
            sys_ops.paste()
            done = True
        elif intent == "delete_text" and "delete" in query:
            sys_ops.delete()
            done = True
        elif intent == "new_file" and "new" in query:
            sys_ops.new_file()
            done = True
        elif intent == "switch_tab" and "switch" in query and
"tab" in query:
            tab_ops.switchTab()
            done = True
        elif intent == "close_tab" and "close" in query and "tab"
in query:
            tab_ops.closeTab()
            done = True
        elif intent == "new_tab" and "new" in query and "tab" in
query:
            tab_ops.newTab()
            done = True
        elif intent == "close_window" and "close" in query:
            win_ops.closeWindow()
            done = True
        elif intent == "switch_window" and "switch" in query:
            win_ops.switchWindow()
            done = True
        elif intent == "minimize_window" and "minimize" in query:
            win_ops.minimizeWindow()
            done = True
        elif intent == "maximize_window" and "maximize" in query:
            win_ops.maximizeWindow()
            done = True
        elif intent == "screenshot" and "screenshot" in query:
            win_ops.Screen_Shot()
            done = True
        elif intent == "stopwatch":
            pass
        elif intent == "wikipedia" and ("tell" in query or "about"
in query):
            description = tell_me_about(query)
            if description:
                speak(description)
            else:
                googleSearch(query)
            done = True
        elif intent == "math":
            answer = get_general_response(query)
            if answer:
```

```
                speak(answer)
                done = True
        elif intent == "open_website":
            completed = open_specified_website(query)
            if completed:
                done = True
        elif intent == "open_app":
            completed = open_app(query)
            if completed:
                done = True
        elif intent == "note" and "note" in query:
            speak("what would you like to take down?")
            note = record()
            take_note(note)
            done = True
        elif intent == "get_data" and "history" in query:
            get_data()
            done = True
        elif intent == "exit" and ("exit" in query or "terminate"
in query or "quit" in query):
            exit(0)
        if not done:
            answer = get_general_response(query)
            if answer:
                speak(answer)
            else:
                speak("Sorry, not able to answer your query")
        return


if __name__ == "__main__":
    try:
        listen_audio()
    except:
        print("EXITED")
```

Full code available on GitHub.

CHAPTER 7

TESTING

7.1 Test Results

7.1.1 Test Case 1

Test Case No.	1
Test Type	Functional Test
Name of Test	Verify weather report feature
Test Case Description	The objective of this test case is to verify that the virtual voice assistant is able to fetch and deliver the weather report for a specified city or the current device's location.
Input	Speak “what’s the weather in CITY” OR Speak “what’s the weather today”
Expected Output	The virtual voice assistant should be able to deliver accurate weather reports for different cities or the current device's location.
Actual Output	The virtual voice assistant responds with news latest headlines.
Result	Pass
Comments	Working properly.

7.1.2 Test Case 2

Test Case No.	2
Test Type	Functional Test
Name of Test	Verify image generation feature
Test Case Description	The objective of this test case is to verify that the virtual voice assistant is able to generate an image based on the user's prompt.
Input	Speak “generate an image” After invoking the feature a text prompt must be given “(text prompt to generate image)”
Expected Output	“What kind of image you want to generate?” “Generating image please wait…”

	The virtual voice assistant should display the generated image.
Actual Output	The virtual voice assistant generates image as expected.
Result	Pass
Comments	Working properly.

7.1.3 Test Case 3

Test Case No.	3
Test Type	Functional Test
Name of Test	Verify email sending feature
Test Case Description	The objective of this test case is to verify that the virtual voice assistant is able to send emails to a specified recipient.
Input	Speak “send email” Provide the recipient email address, subject, and body of the email.
Expected Output	If the email is sent successfully, the virtual voice assistant should respond with "Email sent successfully". If there is an error while sending the email, the virtual voice assistant should respond with "Error occurred while sending email".
Actual Output	The virtual voice assistant responds as expected.
Result	Pass
Comments	Working properly.

7.1.4 Test Case 4

Test Case No.	4
Test Type	Functional Test
Name of Test	Verify math feature
Test Case Description	The objective of this test case is to verify that the virtual voice assistant is able to solve simple mathematical equations.
Input	Speak “what is the sum of 255 and 365”
Expected Output	620

Actual Output	The virtual voice assistant calculates as expected.
Result	Pass
Comments	Working properly. Works accurate for simple equation.

7.1.5 Test Case 5

Test Case No.	5
Test Type	Functional Test
Name of Test	Verify opening website and app feature
Test Case Description	The objective of this test case is to verify that the virtual voice assistant is able to open websites and apps in the user's system.
Input	Speak “open APP_NAME” OR Speak “open WEBSITE_NAME”
Expected Output	The virtual voice assistant should open websites and apps in a timely manner.
Actual Output	The virtual voice assistant opens apps and websites as expected.
Result	Pass
Comments	Working properly. Opens installed apps and websites present in websites.py file.

7.1.6 Test Case 6

Test Case No.	6
Test Type	Functional Test
Name of Test	Verify chat history feature
Test Case Description	The objective of this test case is to verify that the virtual voice assistant is able to show the previous chats with date and time.
Input	Speak “show chat history”
Expected Output	The virtual voice assistant should display the previous chats with date and time in a timely manner.
Actual Output	The virtual voice assistant displays chat history as expected.

Result	Pass
Comments	Working properly.

7.1.7 Test Case 7

Test Case No.	7
Test Type	Functional Test
Name of Test	Verify Google search feature
Test Case Description	The objective of this test case is to verify that the virtual voice assistant is able to perform Google searches as per user's command.
Input	Speak “search google for XYZ”
Expected Output	The virtual voice assistant should open a new webpage and perform a Google search for the topic specified by the user.
Actual Output	The virtual voice assistant performs Google search as expected.
Result	Pass
Comments	Working properly.

7.1.8 Test Case 8

Test Case No.	8
Test Type	Functional Test
Name of Test	Verify YouTube video play feature
Test Case Description	The objective of this test case is to verify that the virtual voice assistant is able to play specified videos on YouTube as per user's command.
Input	Speak “play XYZ video”
Expected Output	The virtual voice assistant should open a new webpage and play the specified video on YouTube.
Actual Output	The virtual voice assistant plays YouTube videos as expected.
Result	Pass
Comments	Working properly.

7.1.9 Test Case 9

Test Case No.	9
Test Type	Functional Test
Name of Test	Verify map display feature
Test Case Description	The objective of this test case is to verify that the virtual voice assistant is able to display maps of specified locations and routes as per user's command.
Input	Speak “show the map of CITY” OR Speak “show distance between CITY1 and CITY2”
Expected Output	The virtual voice assistant should open a new webpage and display the map of the specified location. OR The virtual voice assistant should open a new webpage and display the route and distance between the two specified locations.
Actual Output	The virtual voice assistant display maps and routes as expected.
Result	Pass
Comments	Working properly.

7.1.10 Test Case 10

Test Case No.	10
Test Type	Unit Test
Name of Test	Verify exit functionality
Test Case Description	The objective of this test case is to verify that the virtual voice assistant is able to terminate itself as per user's command.
Input	Speak “exit”
Expected Output	The virtual voice assistant should terminate itself with exit code 0.
Actual Output	The virtual voice assistant terminate itself as expected.
Result	Pass
Comments	Working properly.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion:

In conclusion, the voice assistant developed in this project is capable of performing various tasks such as browsing the internet, sending emails, generating images, and interacting with the user through conversation. It is able to do so by utilizing various APIs and technologies such as stability_sdk, Google Speech Recognition, and SMTP. The voice assistant is also able to perform system tasks such as opening and closing tabs, windows, and applications, as well as taking screenshots and manipulating text in the clipboard.

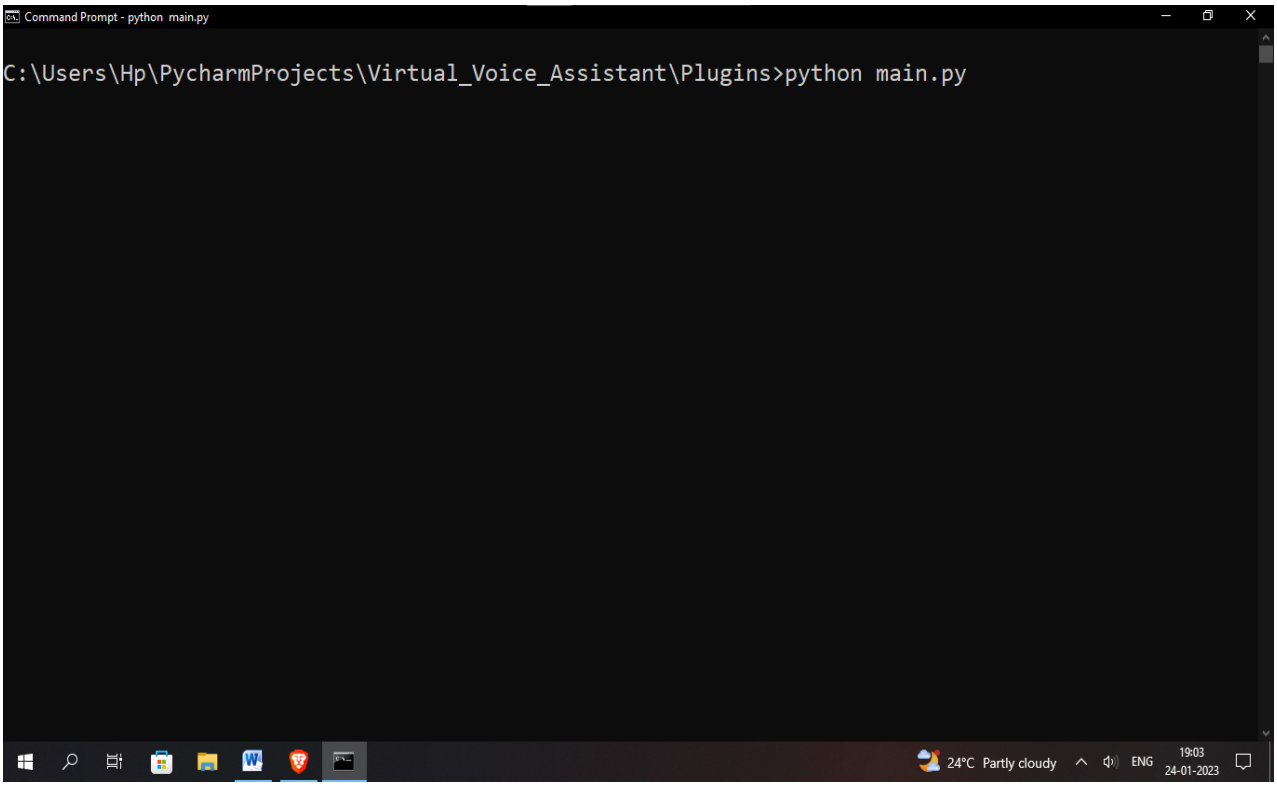
Future Enhancement:

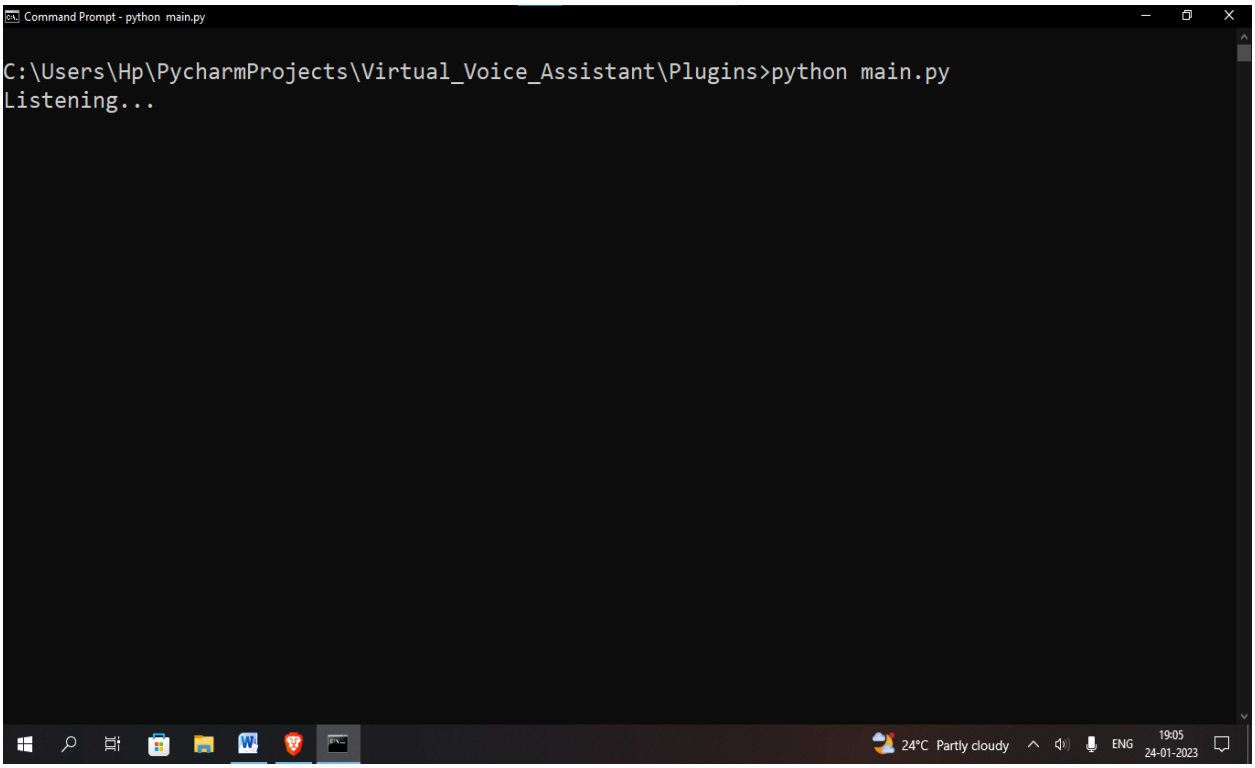
There are several potential areas for future enhancement for the voice assistant. One possibility is to improve the natural language processing capabilities of the chatbot model, in order to enable more seamless conversation with the user. Another possibility is to expand the range of tasks that the voice assistant can perform, for example by integrating with more APIs and services. Additionally, the voice assistant could be made more user-friendly by adding features such as voice prompts and visual feedback. By continuing to improve and expand upon the functionality of the voice assistant, it has the potential to become a valuable tool for users looking to streamline their daily tasks and improve their productivity.

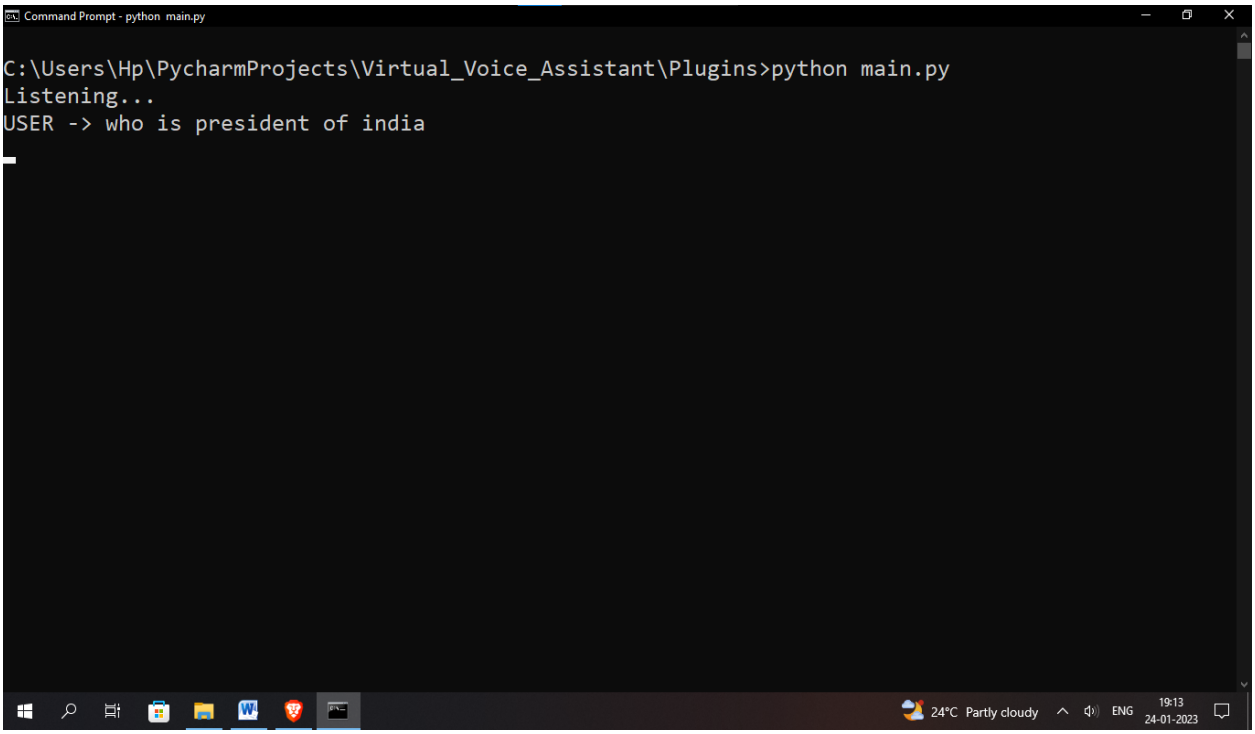
CHAPTER 9

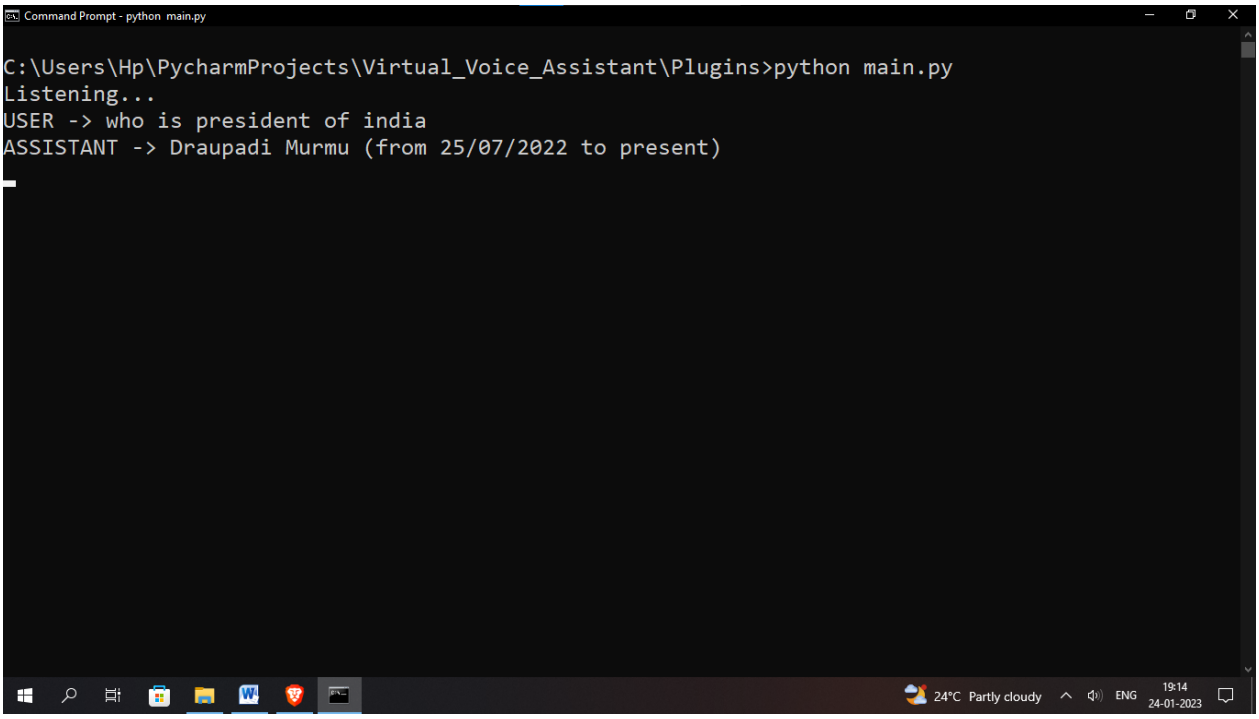
SCREENSHOTS

1. A Desktop View




Command Prompt - python main.py
C:\Users\Hp\PycharmProjects\Virtual_Voice_Assistant\Plugins>python main.py
Listening...
24°C Partly cloudy
ENG
19:05
24-01-2023


Command Prompt - python main.py
C:\Users\Hp\PycharmProjects\Virtual_Voice_Assistant\Plugins>python main.py
Listening...
USER -> who is president of india
24°C Partly cloudy
ENG
19:13
24-01-2023


Command Prompt - python main.py
C:\Users\Hp\PycharmProjects\Virtual_Voice_Assistant\Plugins>python main.py
Listening...
USER -> who is president of india
ASSISTANT -> Draupadi Murmu (from 25/07/2022 to present)
24°C Partly cloudy
ENG
19:14
24-01-2023

CHAPTER 10

REFERENCES

[1] Kumar. Personal Assistant. Github, Jan 13, 2021. (https://github.com/roshan9419/PersonalAssistantChatbot)

[2] Jamie Alexandre, David Marx, Stephan Auerhahn, John Sabath, Arijit Basu, Wes Brown, Jacob Kelley, Frankwin Faber and Chris Allen. SDK for interacting with stability.ai APIs (e.g. stable diffusion inference). Dec 01, 2022. (https://github.com/Stability-AI/stability-sdk)

[3] Anonymous. Sending Emails via Gmail with Python. Script Reference, Sep 23, 2020. (https://scriptreference.com/sending-emails-via-gmail-with-python/)

[4] Amila Viraj. How To Build Your Own Chat Bot Using Deep Learning. Towards Data Science, Nov 1, 2020. (https://towardsdatascience.com/how-to-build-your-own-chatbot-using-deep-learning-bb41f970e281)