

ABSTRACT

Personal Assistants, or conversational interfaces, or chat bots reinvent a new way for individuals to interact with computers. A Personal Virtual Assistant allows a user to simply ask questions in the same manner that they would address a human, and are even capable of doing some basic tasks like opening apps, reading out news, taking notes etc., with just a voice command. Personal Assistants like Google Assistant, Alexa, Siri work by Speech Recognition (Speech-to-text) and Text-to-Speech. Today the technological advancement is increasing day by day. Earlier only there was a computer system in which we can do only few tasks. But now machine learning, artificial intelligence, deep learning, and few more technologies have made computer systems so advanced that we can perform any type of task. In such era of advancement if people are still struggling to interact using various input devices, then it's not worth it. For this reason, we developed a voice assistant using python which allows the user to run any type of command in windows without interaction with keyboard. The main task of voice assistant is to minimize the use of input devices like keyboard, mouse etc. It will also reduce the hardware space and cost.

Table Of Content

	pgno
1.Introduction	
1.1 About the company:	13
1.2 Services provides:	13
2.About the Project	
2.1 Project Title	14
2.2 Project Introduction	14
2.3 Objectives and scope of the project	15
2.4 Project Requirements	15
2.5 Languages Used	16
3.Tecnologies Used	16
3.1 Visual studio IDE	16
4.Methodology and phases	
4.1 Requirements	20
4.2 Design Document	20
4.3 Training	20
4.4 Development	20
4.5 Integration and testing	20
4.6 Implementation	20
4.7 Literature survey	20

5.Project Work Flow

5.1 Problem Statement	21
5.2 Setup and installation of vs code	21
5.3 Design	22
5.4 Methodology	23
5.5Proposed Work	24

6.Glossary

6.1 Implementation	25
6.2 Final Code	38
6.3 Project Output	47

7.Conclusion

8. References	61
---------------	----

LIST OF FIGURES

	pgno:
6.1.1 Install the below commands in terminal	25
6.1.2 Import the commands for the program which are required	26
6.1.3 speak function	26
6.1.4 defining the user take command to get the input speech	27
6.1.5 defining wish me function	27
6.1.6 defining the required functions	28
6.1.7 main function	28
6.1.8 To get the wikipedia	29
6.1.9 open any website using webbrowser	29
6.1.10 open any windows apps using subprocess popen	29
6.1.11 control the volume pyautogui	30
6.1.12 Restart, log off, shut down the system using import os module	30
6.1.13 make a note using assistant	31
6.1.14 get the date time month day using import datetime module	31
6.1.15 ask some questions about assistant	32
6.1.16 convert currency using forex exchange	32
6.1.17 get the internet speed using speedtestcli	33
6.1.18 get live weather using open weather api key	33
6.1.19 calculate distance from one place to another place	34
6.1.20 we can calculate the numbers using api key	34
6.1.21 close any apps using os module	35
6.1.22 empty the recycle bin using winsheel	35
6.1.23 set an alarm using the assistant	35
6.1.24 get the livestock price, open amazon, open chrome, open mail inbox, search anything in google	36
6.1.25 take a screenshot of the current application	36
6.1.26 play music	36
6.1.27 exit the assistant	37

	pgno:
6.3.1 executing the code	47
6.3.2 Wikipedia Search	47
6.3.3 Open Chrome	48
6.3.4 Open YouTube	48
6.3.5 Open Presidency University	49
6.3.6 Open News	49
6.3.7 Open Calculator	50
6.3.8 Open Notepad	50
6.3.9 Control Volume Down	51
6.3.10 Open default Images	51
6.3.11 Control Volume Mute	52
6.3.12 Control Volume Up	52
6.3.13 Empty Recycle Bin	53
6.3.14 Empty Recycle Bin	53
6.3.15 Make a Note	54
6.3.16 Read Notes	54
6.3.17 Setting Alarm	55
6.3.18 Activating Alarm	55
6.3.19 Checking Live Weather	56
6.3.20 Checking Internet Speed	56
6.3.21 Opening Gmail Inbox	57
6.3.22 Calculating the Distance	57
6.3.23 Open Chrome	58
6.3.24 Close Chrome	58
6.3.25 Using Calculator	59
6.3.26 Exit the Application	59

1.INTRODUCTION

1.1 About the company:

- AiRobosoft is a Software, Robotics combined EV Manufacturing Company with HQ and development centre in Bangalore, India.
- The company Products Include development's using technologies such as Artificial Intelligence, Cloud computing Machine learning, Embedded Systems and Internet of Things.
- The company deals with Data Analytics which includes Data visualization and reporting, Data Warehousing and Big Data.

1.2 Services Provided by the company

The company provides the following services:

- Automotive
- Machine Learning
- Data science
- Embedded systems
- Internet of things
- Data Analytics and Industrial Automation

2. About the Project

2.1 Project Title

Virtual Assistant

2.2 Project Introduction

In this era of technology everything that human being can do are being replaced by machines. One of the main reasons is change in performance. In today's world we train our machine to think like humans and do their task by themselves. Therefore, there came a concept of virtual assistant. A virtual assistant is a digital assistant that uses voice recognition features and language processing algorithms to recognize voice commands of user and perform relevant tasks as requested by the user. Based on specific commands given by the user a virtual assistant is capable of filtering out the ambient noise and return relevant information. Virtual Assistant are completely software based but nowadays they are integrated in different devices and also some of the assistants are designed specifically for single devices like Alexa. Due to drastic change in technology now it's a. high time to train our machine with the help of machine learning, deep learning, neural networks. Today we can talk to our machine with the help of Voice Assistant. Today every big company is using Voice Assistant so that their user can take the help of machine through their voice. So, with the Voice Assistant we are moving to the next level advancement where we are able to talk to our machine. interact with it using their voice only. The Voice Assistant that we have developed is for Windows users. The voice assistant we have developed is a desktop-based built using python modules and libraries. This assistant is just a basic version that could perform all the basic tasks.

2.3 Objectives and scope of the project

Virtual Assistants are nothing but AI (Artificial Intelligence) based programs. These programmed applications perceive human commands and language that leads to program performing tasks. To be more precise, these smart computer programs understand human language through text or particular voice commands. Consequently, they perform tasks for the user.

Ideally, Virtual Assistants execute simple human jobs, namely, adding tasks to the calendar, rendering information that is usually searched for in web browsers, and keeping track of smart devices at homes such as cameras, lights, or thermostats.

Virtual Assistants are basically a part of the IoT (Internet of Things). IoT is usually a network/chain of "things" that collect and transmit data over a wireless network. The best part is, this process occurs without any human intervention.

Coming back to virtual assistants, the core of these assistants comprises an audio engine and speech recognition programs. Once you obtain this core, you will come across endless possibilities along with multiple features to add.

2.4 Project Requirements

Hardware:

Pentium-pro processor or later.

RAM 1gb or more

Speaker, microphone

Software:

Windows 7 or any later versions

Vs code

2.5 Languages Used

2.5.1 Python:

Python is an interpreted high-level, general-purpose language. It is dynamically typed and garbage collected, supports multiple programming paradigms. The version of python used in this project is 3.7 64-bit. A lot many packages were used too for the completion of the said project.



3.Techologies Used:

3.1 Visual studio IDE:

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.



3.2 Packages used:

To build a personal voice assistant it's necessary to install the following packages in your system using the pip command.

- 1) Speech recognition — Speech recognition is an important feature used in house automation and in artificial intelligence devices. The main function of this library is it tries to understand whatever the humans speak and converts the speech to text.

```
pip install speech Recognition
```

- 2) pytsx3 — pytsx3 is a text to speech conversion library in python. This package supports text to speech engines on Mac os x, Windows and on Linux.

```
pip install pytsx3
```

- 3) Wikipedia — Wikipedia is a multilingual online encyclopaedia used by many people from academic community ranging from freshmen to students to professors who wants to gain information over a particular topic. This package in python extracts data required from Wikipedia.

```
pip install wikipedia
```

- 4) datetime — This is an inbuilt module in python and it works on date and time

```
import datetime
```

- 5) os — This module is a standard library in python and it provides the function to interact with operating system

```
import os
```

- 6) Subprocess — This is a standard library use to process various system commands like to log off or to restart your PC.

```
import subprocess
```

- 7) request - The request module is used to send all types of HTTP request. Its accepts URL as parameters and gives access to the given URL'S.

```
import requests
```

- 8) time — The time module helps us to display time

```
import time
```

- 9) smtplib-The **smtplib** module defines an SMTP client session object that can be used to send mail to any internet machine with an SMTP or ESMTP listener daemon.

```
import smtplib
```

10) Web browser — This is an in-built package in python. It extracts data from the web

```
import webbrowser
```

11) Wolfram-alpha - Wolfram Alpha is an API which can compute expert-level answers using Wolfram's algorithms, knowledge base and AI technology. It is made possible by the Wolfram Language.

```
Pip install wolframalpha
```

12) pip install speedtest-cli- Command line interface for testing internet bandwidth using speedtest.net

```
pip install speedtest-cli
```

13) pip install pyAutoGUI- PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse & keyboard.

```
Pip install pyautogui
```

14) pip install forex-python- Forex Python is a Free Foreign exchange rates and currency conversion. Features: List all currency rates. Bit Coin price for all currencies. Converting amount to Bit Coins. Get historical rates for any day since 1999. Conversion rate for one currency (ex; USD to INR). Convert amount from one currency to other. ('USD 10\$' to INR). Currency symbols. Currency names.

```
Pip install forex-python
```

15) pip install winshell- It includes convenience functions for accessing special folders, for using the shell's file copy, rename & delete functionality, and a certain amount of support for structured storage.

```
pip install winshell
```

16) pip install geopy- geopy makes it easy for Python developers to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources.

```
pip install geopy
```

17) pip install geocoder- from geopy. Geocoders import Nominatim and then create an instance of Nominatim and provide your *App name (as per your choice)

```
pip install geocoder
```

3.3 API'S USED:

Api: API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

Eg: weather api, covid api, twitter api, google maps api.

Weather: 8ef61edcf1c576d65d836254e11ea420

[“<https://api.openweathermap.org/data/2.5/weather?>”](https://api.openweathermap.org/data/2.5/weather?)

Calculate: "JUGV8R-RXJ4RP7HAG"

4.METHODOLOGY AND PHASES

4.1 Planning /Requirements: Understanding the requirements that are necessary for virtual assistant

4.2 Design Document: Designing of the requirement modules.

4.3 Training: Train the model to be suitable in different scenarios.

4.4 Development: Developing the application its complete functionality

4.5 Integration and testing: Testing of various modules of the software.

4.6 Implementation: Implementation of the software application.

4.7 Literature survey: Personal Assistants, or Virtual assistants have become an integral part of our lives these days. Every organisation, or individual is switching to this kind of technologies, as they help them get their things done in an easier way. This system is based on a Desktop Application. This system includes a Virtual Assistant, that is capable of accepting input from the user, understanding it, analysing it, and performing tasks accordingly. This helps users save a lot of time.

5.Project Work Flow

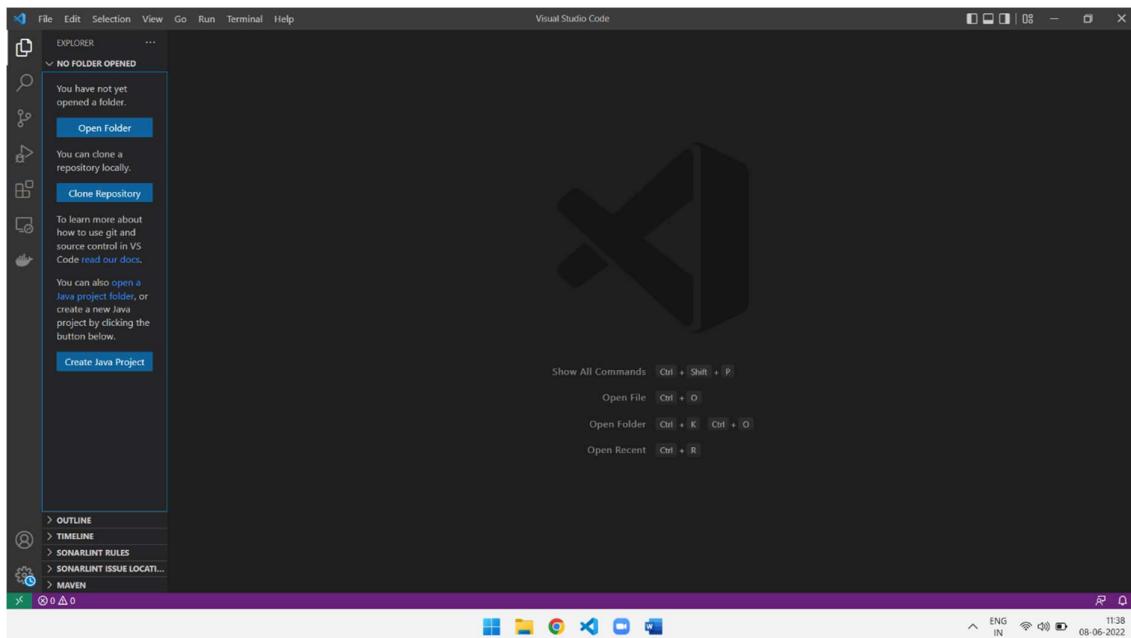
5.1 Problem Statement:

Build Your Own Sci-fi Ai Personal Assistant. The implemented voice assistant should perform the following task it can open YouTube, Gmail, Google chrome and stack overflow. Predict current time, take a photo, search Wikipedia to abstract required data, predict weather in different cities, get top headline news from Times of India and can answer computational and geographical questions too

5.2 Setup and installation of vs code

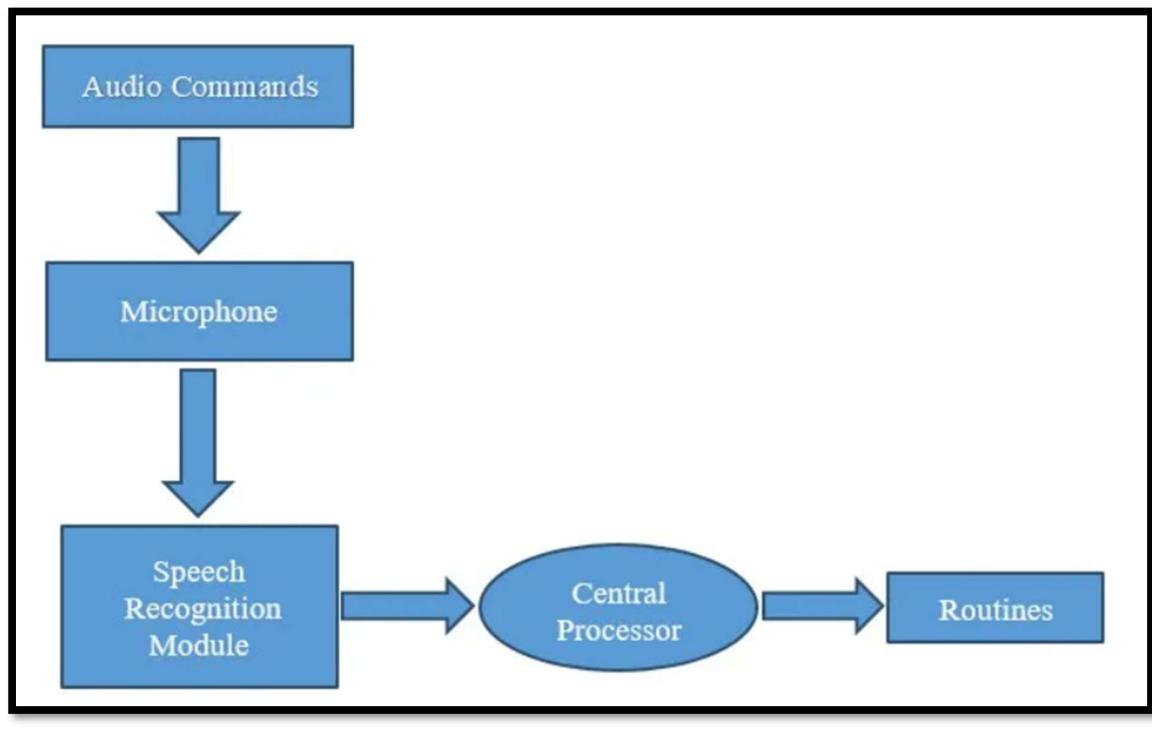
Download the [Visual Studio Code installer](#) for Windows.

Once it is downloaded, run the installer (VSCodeUserSetup-{version}.exe). This will only take a minute. Select the path or keep the default path And start the installation



5.3 Design

The work started with analysing the audio commands given by the user through the microphone. This can be anything like getting any information, operating a computer's internal files, etc. This is an empirical qualitative study, based on reading above mentioned literature and testing their examples. Tests are made by programming according to books and online resources, with the explicit goal to find best practices and a more advanced understanding of Voice Assistant.



5.4 Methodology:

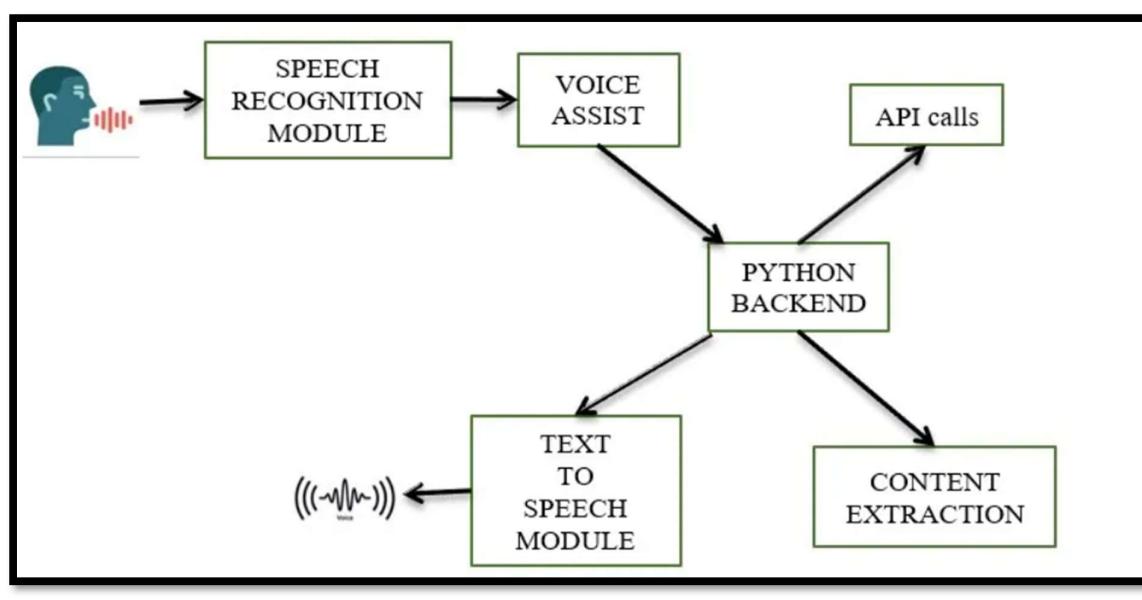
The system uses Google's online speech recognition system for converting speech input to text. The speech input Users can obtain texts from the special corpora organized on the computer network server at the information centre from the microphone is temporarily stored in the system which is then sent to Google cloud for speech recognition. The equivalent text is then received and fed to the central processor.

The python backend gets the output from the speech recognition module and then identifies whether the command or the speech output is an API Call and Context Extraction. The output is then sent back to the python backend to give the required output to the user.

API stands for Application Programming Interface. An API is a software intermediary that allows two applications to talk to each other. In other words, an API is a messenger that delivers your request to the provider that you're requesting it from and then delivers the response back to you.

Context extraction (CE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. In most cases, this activity concerns processing human language texts using natural language processing (NLP). Recent activities in multimedia document processing like automatic annotation and content extraction out of images/audio/video could be seen as context extraction TEST RESULTS.

Text-to-Speech (TTS) refers to the ability of computers to read text aloud. A TTS Engine converts written text to a phonemic representation, then converts the phonemic representation to waveforms that can be output as sound. TTS engines with different languages, dialects and specialized vocabularies are available through third-party publishers.



5.5Proposed Work

In this proposed concept effective way of implementing a Personal voice assistant, Speech Recognition library has many in-built functions, that will let the assistant understand the command given by user and the response will be sent back to user in voice, with Text to Speech functions. When assistant captures the voice command given by used, the under lying algorithms will convert the voice into text. And according to the keywords present in the text (command given by user), respective action will be performed by the assistant. This is made possible with the functions present in different libraries. Also, the assistant was able to achieve all the functionalities with help of some API's. We had used these APIs for functionalities like performing calculations, extracting news from web sources, and for some other things. We will be sending a request, and through the API, we're getting the respective output. API's like WOLFRAMALPHA, are very helpful in performing things like calculations, making small web searches. And for getting the data from web, not every API will have the capability to convert the raw JSON data into text. So, we used a library called JSON, and it will help in parsing the JSON Data coming from websites, to string format. In this way, we are able to extract news from the web sources, and send them as input to a function for further purposes. Also, we have libraries like Random and many other libraries, each corresponding to a different technology. We used the library OS to implement Operating System related functionalities like Shutting down a system.

6.Implementation

6.1.1 Install the below commands in terminal

pip install pytsxs3

pip install Speech Recognition

pip install pipwin

pipwin install pyaudio

pip install Wikipedia

pip install speedtest-cli

pip install pyAutoGUI

pip install forex-python

pip install requests

pip install winshell

pip install geopy

pip install geocoder

pip install wolframalpha

```
from time import time
from more_itertools import take
import pytsxs3
import datetime
import speech_recognition as sr
import wikipedia
import webbrowser
import os
import requests
import subprocess as sp
import wolframalpha
import json
import random
from PIL import Image
import pyautogui
import MyAlarm
import winshell as winshell
from forex_python.converter import CurrencyRates
from speedtest import Speedtest
import pywhatkit
from geopy.geocoders import Nominatim
from geopy import distance
```

6.1.2 Import the commands for the program which are required:

```
from time import time
import pyttsx3
import datetime
import speech_recognition as sr
import wikipedia
import webbrowser
import os
import requests
import subprocess as sp
import wolframalpha
import json
import random
from PIL import Image
import pyautogui
import MyAlarm
import winshell as winshell
from forex_python.converter import CurrencyRates
from speedtest import Speedtest
import pywhatkit
from geopy.geocoders import Nominatim
from geopy import distance
import pywhatkit
```

6.1.3 speak function :

```
engine = pyttsx3.init('sapi5')
voices= engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
```

What is sapi5:

Microsoft developed speech api , helps in synthesis and recognition of voice

What is voice Id:

Voiceid helps us to select different voices

voice[0]: Male voice

voice[0]: Female voice

6.1.4 defining the user take command to get the input speech :

```
def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)

    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language='en-in')
        print(f"User said: {query}\n")

    except Exception as e:
        print("Say that again please...")
        return "None"
    return query
```

6.1.5 defining wish me function :

```
def wishMe():
    hour = int(datetime.datetime.now().hour)
    if hour>=0 and hour<12:
        speak("Good Morning!")

    elif hour>=12 and hour<18:
        speak("Good Afternoon!")

    else:
        speak("Good Evening!")

    print("Loading your AI personal assistant Sam")
    speak("I am Sam Sir. Please tell me how may I help you")
```

6.1.6 defining the required functions :

```
def speak(audio):
    engine.say(audio)
    engine.runAndWait()

def open_cmd():
    os.system('start cmd')

def open_calculator():
    sp.Popen(paths['calculator'])

def open_camera():
    sp.run('start microsoft.windows.camera:', shell=True)

def open_notepad():
    os.startfile(paths['notepad'])

def wishMe():
    hour = int(datetime.datetime.now().hour)
    if hour>=0 and hour<12:
        speak("Good Morning!")

    elif hour>=12 and hour<18:
        speak("Good Afternoon!")

    else:
        speak("Good Evening!")

    print("Loading your AI personal assistant Sam")
    speak("I am Sam Sir. Please tell me how may I help you")
```

6.1.7 main function :

```
if __name__=="__main__":
    wishMe()
    while True:
        query = takeCommand().lower()
```

6.1.8 To get the wikipedia :

```
if 'wikipedia' in query:  
    speak('Searching Wikipedia...')  
    query = query.replace("wikipedia", "")  
    results = wikipedia.summary(query, sentences=3)  
    speak("According to Wikipedia")  
    print(results)  
    speak(results)
```

Here we can change the sentences to our needs.

6.1.9 open any website using webbrowser:

```
elif 'open youtube' in query:  
    webbrowser.open("youtube.com")  
  
elif 'open google' in query:  
    webbrowser.open("google.com")  
  
elif 'open stackoverflow' in query:  
    webbrowser.open("stackoverflow.com")  
  
elif 'news' in query:  
    news = webbrowser.open_new_tab("https://timesofindia.indiatimes.com/home/headlines")  
    speak('Here are some headlines from the Times of India - Happy reading')  
  
elif 'presidency university' in query:  
    news = webbrowser.open_new_tab("https://presidencyuniversity.in/")  
    speak('Here is the presidency university home page')
```

6.1.10 open any windows apps using subprocess popen:

```
elif 'open calculator' in query:  
    sp.Popen('C:\\Windows\\System32\\calc.exe')  
  
elif 'open command prompt' in query or 'open cmd' in query:  
    sp.Popen('C:\\Windows\\System32\\cmd.exe')  
  
elif 'open notepad' in query:  
    sp.Popen('C:\\Windows\\notepad.exe')  
  
elif 'open camera' in query:  
    open_camera()
```

6.1.11 control the volume pyautogui:

```
elif 'volume up' in query:  
    pyautogui.press("volumeup")  
  
elif 'volume down' in query:  
    pyautogui.press("volumedown")  
  
elif 'mute volume' in query:  
    pyautogui.press("volumemute")
```

6.1.12 Restart , log off , shut down the system using import os module :

```
elif 'shut down' in query:  
    print("Do you want to shutdown your system?")  
    speak("Do you want to shutdown your system?")  
    cmd = takeCommand()  
    if 'no' in cmd:  
        continue  
    else:  
  
        os.system("shutdown /s /t 1")  
  
elif 'restart' in query:  
    print("Do you want to restart your system?")  
    speak("Do you want to restart your system?")  
    cmd = takeCommand()  
    if 'no' in cmd:  
        continue  
    else:  
  
        os.system("shutdown /r /t 1")  
  
elif 'log out' in query:  
    print("Do you want to logout from your system?")  
    speak("Do you want to logout from your system?")  
    cmd = takeCommand()  
    if 'no' in cmd:  
        continue  
    else:  
        os.system("shutdown -l")
```

6.1.13 make a note using assistant :

```
elif 'write a note' in query or 'make a note' in query:
    speak("What should I write, sir??")
    note = takeCommand()
    file = open('Notes.txt', 'a')
    speak("Should I include the date and time??")
    n_conf = takeCommand()
    if 'yes' in n_conf:
        str_time = datetime.datetime.now().strftime("%H:%M:%S")
        file.write(str_time)
        file.write(" --> ")
        file.write(note)
        speak("Point noted successfully.")
    else:
        file.write("\n")
        file.write(note)
        speak("Point noted successfully.")

elif 'show me the notes' in query or 'read notes' in query:
    speak("Reading Notes")
    file = open("Notes.txt", "r")
    data_note = file.readlines()
    # for points in data_note:
    print(data_note)
    speak(data_note)
```

6.1.14 get the date time month day using import datetime module

```
elif 'date' in query:
    strDate = datetime.datetime.today().strftime('%Y-%m-%d')
    print(strDate)
    speak(f"The date is {strDate}")

elif 'month' in query or 'month is going' in query:
    def tell_month():
        month = datetime.datetime.now().strftime("%B")
        speak(month)
    tell_month()

elif 'day' in query or 'day today' in query:
    def tell_day():
        day = datetime.datetime.now().strftime("%A")
        speak(day)
    tell_day()
```

6.1.15 ask some questions about assistant :

```
elif 'who are you' in query:  
    speak("I am P.A. (Python Assistant), developed by chaitanya as a project in their company.")  
  
elif 'what you do' in query:  
    speak("I want to help people to do certain tasks on their single voice commands.")  
  
elif 'what language you use' in query:  
    speak("I am written in Python and I generally speak english.")
```

6.1.16 convert currency using forex exchange :

```
elif 'convert currency' in query:  
    try:  
        curr_list = {  
            'dollar': 'USD',  
            'rupee': 'INR'  
        }  
  
        cur = CurrencyRates()  
        # print(cur.get_rate('USD', 'INR'))  
        speak('From which currency u want to convert?')  
        from_cur = takeCommand()  
        src_cur = curr_list[from_cur.lower()]  
        speak('To which currency u want to convert?')  
        to_cur = takeCommand()  
        dest_cur = curr_list[to_cur.lower()]  
        speak('Tell me the value of currency u want to convert.')  
        val_cur = float(takeCommand())  
        # print(val_cur)  
        print(cur.convert(src_cur, dest_cur, val_cur))  
        speak(cur.convert)  
  
    except Exception as e:  
        print("Couldn't get what you have said, Can you say it again??")
```

6.1.17 get the internet speed using speedtestcli :

```
elif 'internet speed' in query:
    st = Speedtest()
    print("Wait!! I am checking your Internet Speed...")
    speak("Wait!! I am checking your Internet Speed...")
    dw_speed = st.download()
    up_speed = st.upload()
    dw_speed = dw_speed / 1000000
    up_speed = up_speed / 1000000
    print('Your download speed is', round(dw_speed, 3), 'Mbps')
    print('Your upload speed is', round(up_speed, 3), 'Mbps')
    speak(f'Your download speed is {round(dw_speed, 3)} Mbps')
    speak(f'Your upload speed is {round(up_speed, 3)} Mbps')
```

6.1.18 get live weather using open weather api key :

```
elif "weather" in query:
    api_key="8ef61edcf1c576d65d836254e11ea420"
    base_url="https://api.openweathermap.org/data/2.5/weather?"
    speak("whats the city name")
    city_name=takeCommand()
    complete_url=base_url+"appid="+api_key+"&q="+city_name
    response = requests.get(complete_url)
    x=response.json()
    if x["cod"]!="404":
        y=x["main"]
        current_temperature = y["temp"]
        current_humidiy = y["humidity"]
        z = x["weather"]
        weather_description = z[0]["description"]
        speak(" Temperature in kelvin unit is " +
              str(current_temperature) +
              "\n humidity in percentage is " +
              str(current_humidiy) +
              "\n description " +
              str(weather_description))
        print(" Temperature in kelvin unit = " +
              str(current_temperature) +
              "\n humidity (in percentage) = " +
              str(current_humidiy) +
              "\n description = " +
              str(weather_description))
```

6.1.19 calculate distance from one place to another place :

```
elif 'distance' in query:
    geocoder = Nominatim(user_agent="Singh")
    speak("Tell me the first city name??")
    location1 = takeCommand()
    speak("Tell me the second city name??")
    location2 = takeCommand()

    coordinates1 = geocoder.geocode(location1)
    coordinates2 = geocoder.geocode(location2)

    lat1, long1 = coordinates1.latitude, coordinates1.longitude
    lat2, long2 = coordinates2.latitude, coordinates2.longitude

    place1 = (lat1, long1)
    place2 = (lat2, long2)

    distance_places = distance.distance(place1, place2)

    print(f"The distance between {location1} and {location2} is {distance_places}.")
    speak(f"The distance between {location1} and {location2} is {distance_places}")
```

6.1.20 we can calculate the numbers using api key :

```
elif "calculate" in query:
    try:
        app_id = "JUGV8R-RXJ4RP7HAG"
        client = wolframalpha.Client(app_id)
        indx = query.lower().split().index('calculate')
        query = query.split()[indx + 1:]
        res = client.query(' '.join(query))
        answer = next(res.results).text
        print("The answer is " + answer)
        speak("The answer is " + answer)

    except Exception as e:
        print("Couldn't get what you have said, Can you say it again??")
```

6.1.21 close any apps using os module :

```
elif 'close chrome' in query:  
    os.system("TASKKILL /F /IM chrome.exe")  
  
elif 'close notepad' in query:  
    os.system("TASKKILL /F /IM notepad.exe")
```

6.1.22 empty the recycle bin using winsheel :

```
elif 'empty recycle bin' in query or 'clear recycle bin' in query:  
    try:  
        winshell.recycle_bin().empty(confirm=False, show_progress=False, sound=True)  
        print("Recycle Bin is cleaned successfully.")  
        speak("Recycle Bin is cleaned successfully.")  
  
    except Exception as e:  
        print("Recycle bin is already Empty.")  
        speak("Recycle bin is already Empty.")
```

6.1.23 set an alram using the assistant :

```
elif 'set alarm' in query:  
    speak("Tell me the time to set an Alarm. For example, set an alarm for 11:21 AM")  
    a_info = takeCommand()  
    a_info = a_info.replace('set an alarm for', '')  
    a_info = a_info.replace('.', '')  
    a_info = a_info.upper()  
    MyAlarm.alarm(a_info)
```

6.1.24 get the live stock prive , open amazon , open chrome , open mail inbox , search anything in google :

```
elif 'stock price' in query:  
    search_term=query.split("for")[-1]  
    url="https://google.com/search?q="+ search_term  
    webbrowser.get().open(url)  
    speak("Here is what i found for"+ search_term)  
  
elif 'search for' in query:  
    search_term=query.split("for")[-1]  
    url="https://google.com/search?q="+ search_term  
    webbrowser.get().open(url)  
    speak("Here is what i found for "+ search_term)  
  
elif 'email' in query:  
    search_term = query.split("for")[-1]  
    url="https://mail.google.com/mail/u/0/#inbox"  
    webbrowser.get().open(url)  
    speak("here you can check your gmail")  
  
elif 'open chrome' in query:  
    os.startfile("C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe")
```

6.1.25 take a screenshot of the current application :

```
elif 'screenshot' in query:  
    myScreenshot=pyautogui.screenshot()  
    myScreenshot.save(r'D:\\wipro\\jarvis\\screenshots\\1.png')  
    speak("Your screenshot is saved")  
    print("Your screenshot is saved")
```

6.1.26 play music:

```
elif 'play music ' in query or ' play songs ' in query:  
    music_dir = 'D:\\wipro\\jarvis\\songs'  
    songs = os.listdir(music_dir)  
    print(songs)  
    os.startfile(os.path.join(music_dir, songs[0]))
```

6.1.27 Exit the application:

```
if "exit" in query or "shut of" in query:  
    speak("Ok ,thank you for using this application")  
    print("Ok ,thank you for using this application")  
    exit()
```

6.2 Final Code

```
from time import time
from more_itertools import take
import pytsx3
import datetime
import speech_recognition as sr
import wikipedia
import webbrowser
import os
import requests
import subprocess as sp
import wolframalpha
import json
import random
from PIL import Image
import pyautogui
import MyAlarm
import winshell as winshell
from forex_python.converter import CurrencyRates
from speedtest import Speedtest
import pywhatkit
from geopy.geocoders import Nominatim
from geopy import distance
import pywhatkit

engine = pytsx3.init('sapi5')
voices= engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)

paths = {
    'notepad': "C:\\Windows\\notepad.exe",
    'calculator':"C:\\Windows\\System32\\calc.exe"
}

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

def open_cmd():
    os.system('start cmd')

def open_calculator():
    sp.Popen(paths['calculator'])

def open_camera():
    sp.run('start microsoft.windows.camera:', shell=True)
```

```

def open_notepad():
    os.startfile(paths['notepad'])

def wishMe():
    hour = int(datetime.datetime.now().hour)
    if hour>=0 and hour<12:
        speak("Good Morning!")

    elif hour>=12 and hour<18:
        speak("Good Afternoon!")

    else:
        speak("Good Evening!")

    print("Loading your AI personal assistant Sam")
    speak("I am Sam Sir. Please tell me how may I help you")

def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)
    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language='en-in')
        print(f"User said: {query}\n")

    except Exception as e:
        print("Say that again please...")
        return "None"
    return query

if __name__=="__main__":
    wishMe()
    while True:
        query = takeCommand().lower()

        if 'open youtube' in query:
            webbrowser.open("youtube.com")

        elif 'open google' in query:
            webbrowser.open("google.com")

        elif 'open github' in query:

```

```

        webbrowser.open("github.com")

    elif 'news' in query:
        news = webbrowser.open_new_tab("https://timesofindia.indiatimes.com/home/headlines")
        speak('Here are some headlines from the Times of India - Happy reading')

    elif 'presidency university' in query:
        news = webbrowser.open_new_tab("https://presidencyuniversity.in/")
        speak('Here is the presidency university home page')

    elif 'play music ' in query or ' play songs ' in query:
        music_dir = 'D:\wipro\jarvis\songs'
        songs = os.listdir(music_dir)
        print(songs)
        os.startfile(os.path.join(music_dir, songs[0]))

    elif 'the time' in query:
        strTime = datetime.datetime.now().strftime("%H:%M:%S")
        speak(f"Sir, the time is {strTime}")
        print(time)

    elif 'open calculator' in query:
        sp.Popen('C:\\Windows\\System32\\calc.exe')

    elif 'open command prompt' in query or 'open cmd' in query:
        sp.Popen('C:\\Windows\\System32\\cmd.exe')

    elif 'open notepad' in query:
        sp.Popen('C:\\Windows\\notepad.exe')

    elif 'open camera' in query:
        open_camera()

    elif "who made you" in query or "who created you" in query or "who discovered you" in query:
        speak("I was built by chaitanya")
        print("I was built by chaitanya")

    elif 'empty recycle bin' in query or 'clear recycle bin' in query:
        try:
            winshell.recycle_bin().empty(confirm=False, show_progress=False,
sound=True)
            print("Recycle Bin is cleaned successfully.")
            speak("Recycle Bin is cleaned successfully.")

        except Exception as e:

```

```
        print("Recycle bin is already Empty.")
        speak("Recycle bin is already Empty.")

    elif 'cricket table' in query:
        im = Image.open(r"C:\Users\chaitu\Downloads\1.jpeg")
        im.show()

    elif 'coin game' in query:
        moves=["head", "tails"]
        cmove=random.choice(moves)
        speak("The computer chose " + cmove)

    elif 'volume up' in query:
        pyautogui.press("volumeup")

    elif 'volume down' in query:
        pyautogui.press("volumedown")

    elif 'mute volume' in query:
        pyautogui.press("volumemute")

    elif 'shut down' in query:
        print("Do you want to shutdown your system?")
        speak("Do you want to shutdown your system?")
        cmd = takeCommand()
        if 'no' in cmd:
            continue
        else:

            os.system("shutdown /s /t 1")

    elif 'restart' in query:
        print("Do you want to restart your system?")
        speak("Do you want to restart your system?")
        cmd = takeCommand()
        if 'no' in cmd:
            continue
        else:

            os.system("shutdown /r /t 1")

    elif 'log out' in query:
        print("Do you want to logout from your system?")
        speak("Do you want to logout from your system?")
        cmd = takeCommand()
        if 'no' in cmd:
            continue
        else:
```

```

os.system("shutdown -l")

elif 'write a note' in query or 'make a note' in query:
    speak("What should I write, sir??")
    note = takeCommand()
    file = open('Notes.txt', 'a')
    speak("Should I include the date and time??")
    n_conf = takeCommand()
    if 'yes' in n_conf:
        str_time = datetime.datetime.now().strftime("%H:%M:%S")
        file.write(str_time)
        file.write(" --> ")
        file.write(note)
        speak("Point noted successfully.")
    else:
        file.write("\n")
        file.write(note)
        speak("Point noted successfully.")

elif 'show me the notes' in query or 'read notes' in query:
    speak("Reading Notes")
    file = open("Notes.txt", "r")
    data_note = file.readlines()
    # for points in data_note:
    print(data_note)
    speak(data_note)

elif 'screenshot' in query:
    myScreenshot=pyautogui.screenshot()
    myScreenshot.save(r'D:\wipro\jarvis\screenshots\1.png')
    speak("Your screenshot is saved")
    print("Your screenshot is saved")

elif 'date' in query:
    strDate = datetime.datetime.today().strftime('%Y-%m-%d')
    print(strDate)
    speak(f"The date is {strDate}")

elif 'month' in query or 'month is going' in query:
    def tell_month():
        month = datetime.datetime.now().strftime("%B")
        speak(month)
    tell_month()

elif 'day' in query or 'day today' in query:
    def tell_day():
        day = datetime.datetime.now().strftime("%A")
        speak(day)

```

```

        tell_day()

    elif 'who are you' in query:
        speak("I am P.A. (Python Assistant), developed by chaitanya as a
project in their company.")

    elif 'what you do' in query:
        speak("I want to help people to do certain tasks on their single
voice commands.")

    elif 'what language you use' in query:
        speak("I am written in Python and I generally speak english.")

    elif 'set alarm' in query:
        speak("Tell me the time to set an Alarm. For example, set an alarm
for 11:21 AM")
        a_info = takeCommand()
        a_info = a_info.replace('set an alarm for', '')
        a_info = a_info.replace('.', '')
        a_info = a_info.upper()
        MyAlarm.alarm(a_info)

    elif 'convert currency' in query:
        try:
            curr_list = {
                'dollar': 'USD',
                'rupee': 'INR'
            }

            cur = CurrencyRates()
            # print(cur.get_rate('USD', 'INR'))
            speak('From which currency u want to convert?')
            from_cur = takeCommand()
            src_cur = curr_list[from_cur.lower()]
            speak('To which currency u want to convert?')
            to_cur = takeCommand()
            dest_cur = curr_list[to_cur.lower()]
            speak('Tell me the value of currency u want to convert.')
            val_cur = float(takeCommand())
            # print(val_cur)
            print(cur.convert(src_cur, dest_cur, val_cur))
            speak(cur.convert)

        except Exception as e:
            print("Couldn't get what you have said, Can you say it again??")

    elif 'internet speed' in query:
        st = Speedtest()

```

```

print("Wait!! I am checking your Internet Speed...")
speak("Wait!! I am checking your Internet Speed...")
dw_speed = st.download()
up_speed = st.upload()
dw_speed = dw_speed / 1000000
up_speed = up_speed / 1000000
print('Your download speed is', round(dw_speed, 3), 'Mbps')
print('Your upload speed is', round(up_speed, 3), 'Mbps')
speak(f'Your download speed is {round(dw_speed, 3)} Mbps')
speak(f'Your upload speed is {round(up_speed, 3)} Mbps')

elif "weather" in query:
    api_key="8ef61edc1c576d65d836254e11ea420"
    base_url="https://api.openweathermap.org/data/2.5/weather?"
    speak("whats the city name")
    city_name=takeCommand()
    complete_url=base_url+"appid="+api_key+"&q="+city_name
    response = requests.get(complete_url)
    x=response.json()
    if x[ "cod"]!="404":
        y=x[ "main"]
        current_temperature = y[ "temp"]
        current_humidiy = y[ "humidity"]
        z = x[ "weather"]
        weather_description = z[0][ "description"]
        speak(" Temperature in kelvin unit is " +
              str(current_temperature) +
              "\n humidity in percentage is " +
              str(current_humidiy) +
              "\n description " +
              str(weather_description))
        print(" Temperature in kelvin unit = " +
              str(current_temperature) +
              "\n humidity (in percentage) = " +
              str(current_humidiy) +
              "\n description = " +
              str(weather_description))

    elif 'distance' in query:
        geocoder = Nominatim(user_agent="Singh")
        speak("Tell me the first city name??")
        location1 = takeCommand()
        speak("Tell me the second city name??")
        location2 = takeCommand()

        coordinates1 = geocoder.geocode(location1)
        coordinates2 = geocoder.geocode(location2)

```

```

        lat1, long1 = coordinates1.latitude, coordinates1.longitude
        lat2, long2 = coordinates2.latitude, coordinates2.longitude

        place1 = (lat1, long1)
        place2 = (lat2, long2)

        distance_places = distance.distance(place1, place2)

        print(f"The distance between {location1} and {location2} is
{distance_places}.")
        speak(f"The distance between {location1} and {location2} is
{distance_places}")

    elif 'stock price' in query:
        search_term=query.split("for")[-1]
        url="https://google.com/search?q="+ search_term
        webbrowser.get().open(url)
        speak("Here is what i found for"+ search_term)

    elif 'search for' in query:
        search_term=query.split("for")[-1]
        url="https://google.com/search?q="+ search_term
        webbrowser.get().open(url)
        speak("Here is what i found for "+ search_term)

    elif 'email' in query:
        search_term = query.split("for")[-1]
        url="https://mail.google.com/mail/u/0/#inbox"
        webbrowser.get().open(url)
        speak("here you can check your gmail")

    elif 'open chrome' in query:
        os.startfile("C:\\Program
Files\\Google\\Chrome\\Application\\chrome.exe")

    elif 'open whatsapp' in query:
        webbrowser.open("https://web.whatsapp.com/")

    elif 'open amazon' in query:
        webbrowser.open("https://www.amazon.in/")

    elif 'close chrome' in query:
        os.system("TASKKILL /F /IM chrome.exe")

    elif 'close notepad' in query:
        os.system("TASKKILL /F /IM notepad.exe")

    elif "calculate" in query:

```

```
try:
    app_id = "JUGV8R-RXJ4RP7HAG"
    client = wolframalpha.Client(app_id)
    indx = query.lower().split().index('calculate')
    query = query.split()[indx + 1:]
    res = client.query(' '.join(query))
    answer = next(res.results).text
    print("The answer is " + answer)
    speak("The answer is " + answer)

except Exception as e:
    print("Couldn't get what you have said, Can you say it again??")

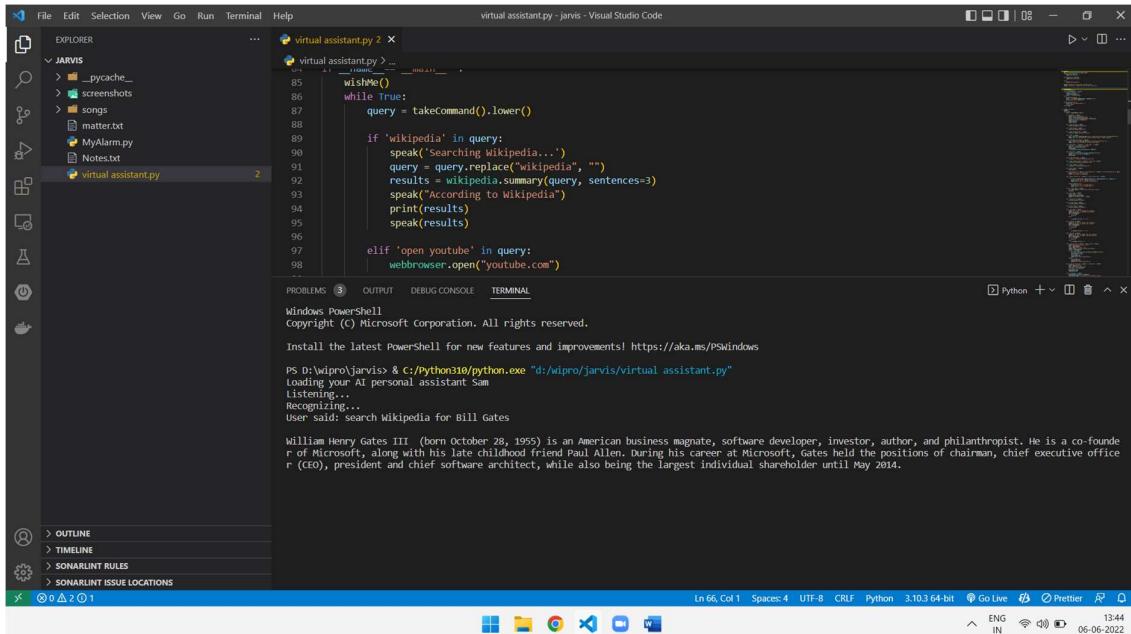
if "exit" in query or "shut of" in query:
    speak("Ok ,thank you for using this application")
    print("Ok ,thank you for using this application")
    exit()
```

6.3 Working:

6.3.1 executing the code:

```
PS D:\wipro\jarvis> & C:/Python310/python.exe "d:/wipro/jarvis/virtual assistant.py"
Loading your AI personal assistant Sam
Listening...
```

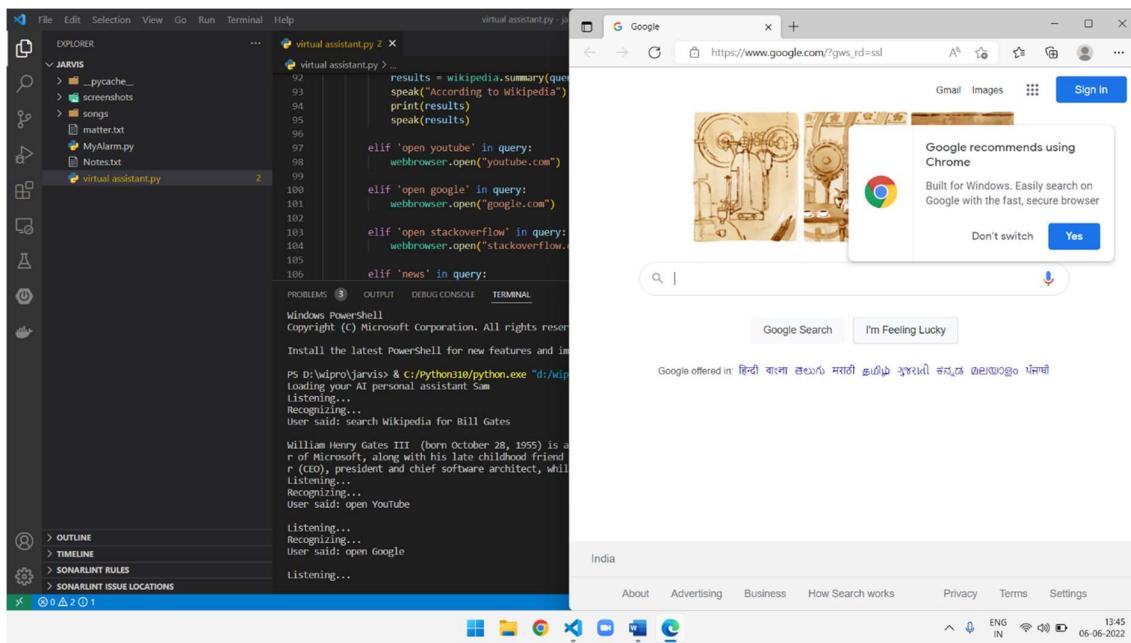
6.3.2 Wikipedia Search:



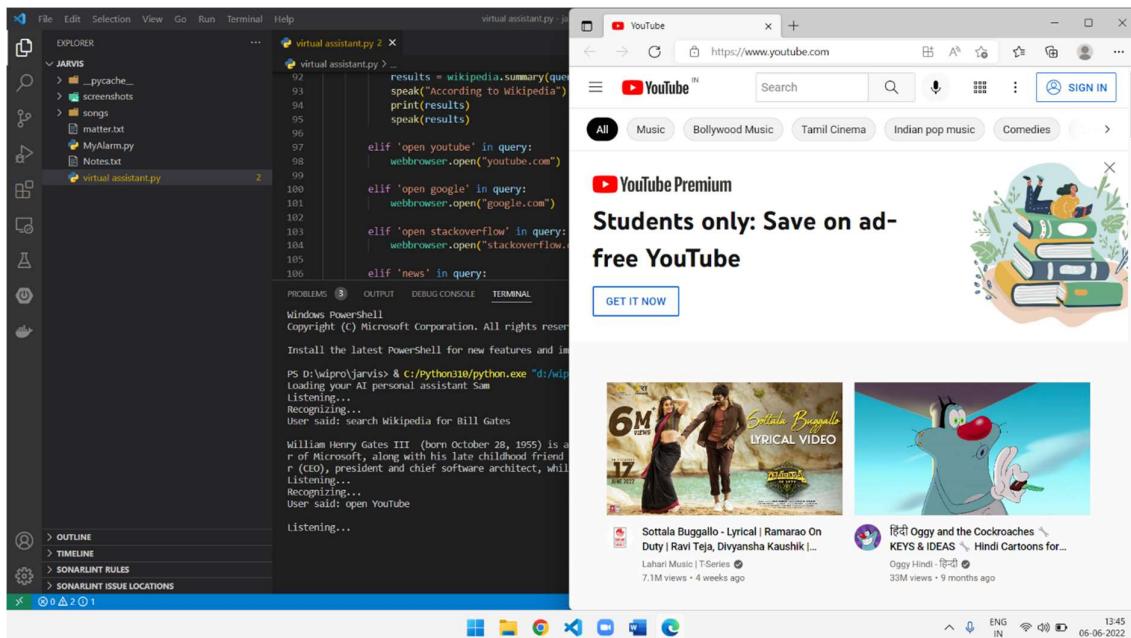
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "JARVIS" containing files like ".pycache", "screenshots", "songs", "matter.txt", "MyAlarm.py", "Notes.txt", and "virtual assistant.py".
- Code Editor:** Displays the "virtual assistant.py" file with Python code. The code includes functions for wishing, taking commands, searching Wikipedia, and opening YouTube.
- Terminal:** Shows the command "PS D:\wipro\jarvis> & C:/Python310/python.exe "d:/wipro/jarvis/virtual assistant.py"" being run, followed by the AI's response: "Loading your AI personal assistant Sam", "Listening...", "Recognizing...", and "User said: search Wikipedia for Bill Gates". Below this, it displays the Wikipedia summary for Bill Gates.
- Status Bar:** Shows the file path "D:\wipro\jarvis\virtual assistant.py", line 66, column 1, and other system information like battery level, network status, and date/time.

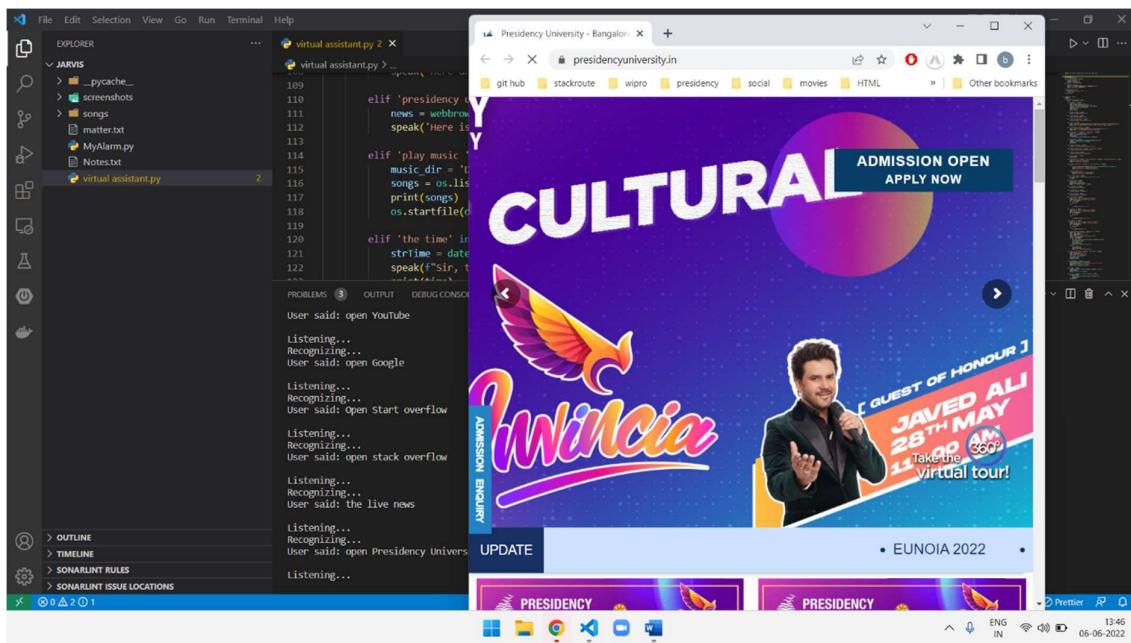
6.3.3 Open Chrome



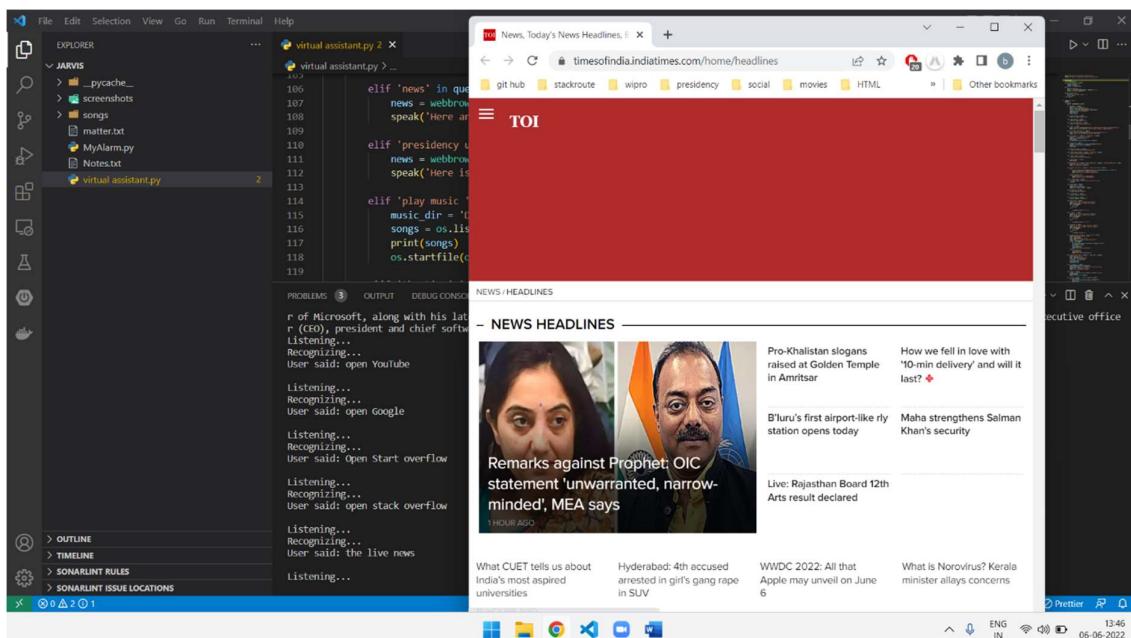
6.3.4 Open YouTube:



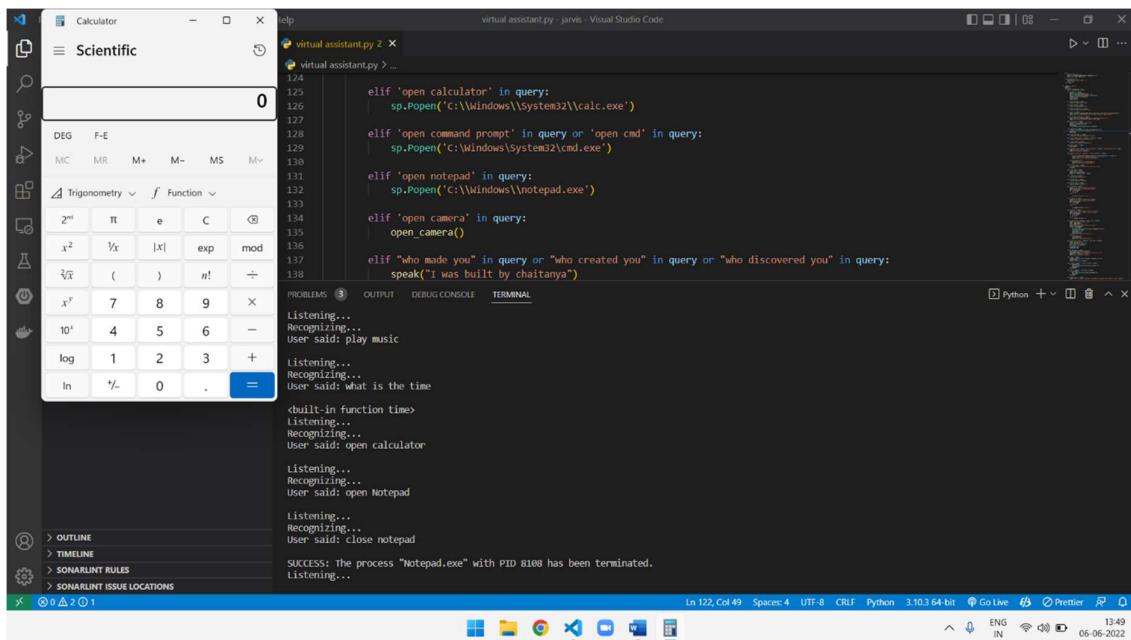
6.3.5 Open Presidency University:



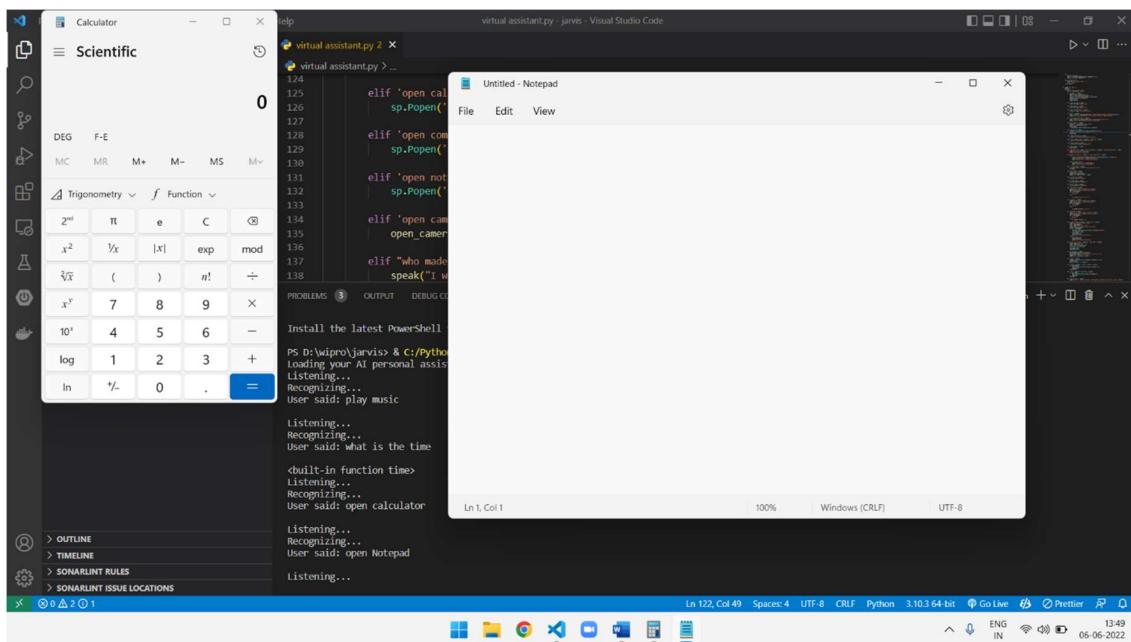
6.3.6 Open News:



6.3.7 Open Calculator:



6.3.8 Open Notepad:



6.3.9 Control Volume Down

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "JARVIS" containing files like ".pycache", "screenshots", "songs", "matter.txt", "Myalarm.py", and "Notes.txt". The file "virtual assistant.py" is currently open.
- Code Editor:** Displays the "virtual assistant.py" code, which includes logic for volume control, shutdown, and note-taking commands using PyAutoGUI.
- Terminal:** Shows the AI assistant's responses to user commands like "open Notepad" and "close notepad".
- Status Bar:** Shows the current file path as "virtual assistant.py - Jarvis - Visual Studio Code", and the status bar indicates "L 129, Col 49" and "Python 3.10.3 64-bit".

6.3.10 Open default Images:

The screenshot shows a Visual Studio Code window with the following details:

- Title Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help, virtual assistant.py - jars - Visual Studio Code
- Code Editor:** The Python script for the virtual assistant. It includes functions for clearing the recycle bin, checking if it's empty, and handling user input for cricket tables and coin games.
- Terminal:** Shows the execution history of the script, including commands like `python virtual assistant.py` and the output of the `coin game` function.
- Sidebar:** A sidebar titled "TATA Indian Premier League 2022 Consolidated Official Schedule of the Tournament" displays the match schedule from March 26 to April 9, 2022, across various venues.
- Status Bar:** Shows file statistics (Line 122, Col 49), spaces used (Spaces: 4), encoding (UTF-8), and terminal type (Python).

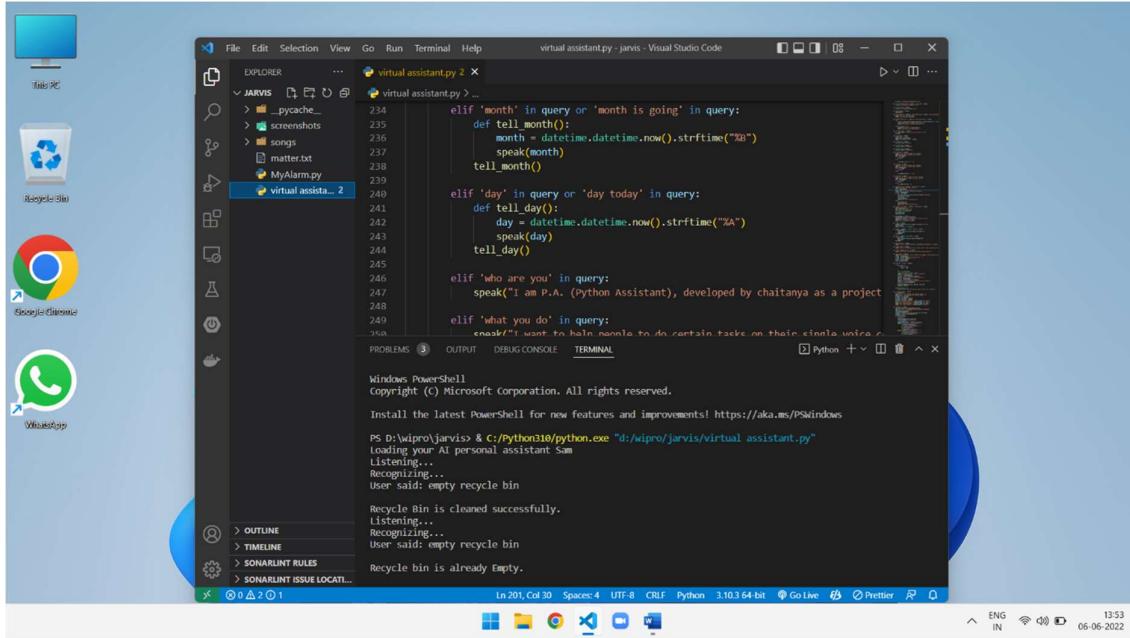
6.3.11 Control Volume Mute:

6.3.12 Control Volume Up:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "JARVIS" containing files like __pycache__, screenshots, songs, matter.txt, Myalarm.py, Notes.txt, and virtual assistant.py (which is currently open).
- Code Editor:** The "virtual assistant.py" file is open, displaying Python code for a voice-controlled assistant. It includes logic for volume control ("volume up", "volume down", "mute volume"), shutdown ("shut down"), and opening applications ("close notepad", "open cricket table").
- Terminal:** The terminal window shows the execution of the script. It prints "Listening...", "Recognizing...", and user commands like "User said: close notepad" and "User said: open cricket table". It also displays a success message: "SUCCESS: The process "Notepad.exe" with PID 8108 has been terminated.".
- Status Bar:** Shows the current file is "virtual assistant.py - Jarvis - Visual Studio Code", the line number is 122, column 49, and the file type is Python 3.10.64-bit.

6.3.12 Empty Recycle Bin:



The screenshot shows a Windows desktop environment with a blue theme. A Visual Studio Code window is open, displaying Python code for a virtual assistant. The terminal tab shows the following interaction:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

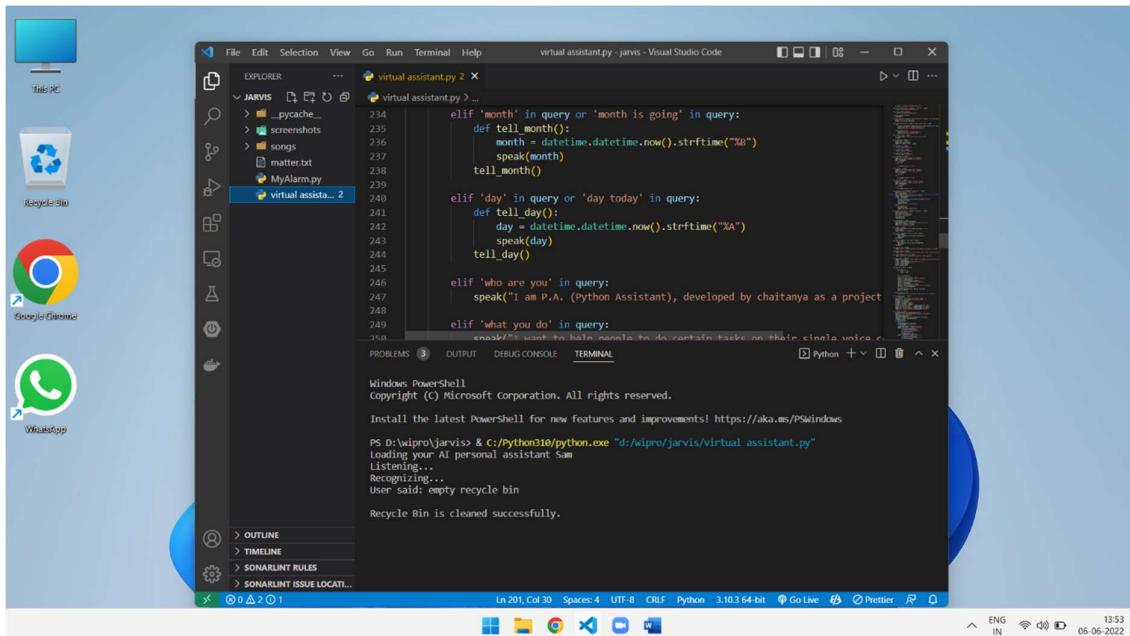
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\wipro\jarvis> & C:\Python310\python.exe "d:/wipro/jarvis/virtual assistant.py"
Loading your AI personal assistant Sam
Listening...
Recognizing...
User said: empty recycle bin

Recycle Bin is cleaned successfully.
```

The desktop icons include a recycle bin, Google Chrome, WhatsApp, and File Explorer.

6.3.13 Empty Recycle Bin:



The screenshot shows a Windows desktop environment with a blue theme. A Visual Studio Code window is open, displaying Python code for a virtual assistant. The terminal tab shows the following interaction:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\wipro\jarvis> & C:\Python310\python.exe "d:/wipro/jarvis/virtual assistant.py"
Loading your AI personal assistant Sam
Listening...
Recognizing...
User said: empty recycle bin

Recycle Bin is cleaned successfully.
```

The desktop icons include a recycle bin, Google Chrome, WhatsApp, and File Explorer.

6.3.14 Make a Note:

The screenshot shows the Visual Studio Code interface with the 'virtual assistant.py' file open. The code handles user input for creating a note. It includes logic to speak the note, save it to a file, and provide feedback. The terminal window shows the program's interaction with the user, including speech recognition and synthesis.

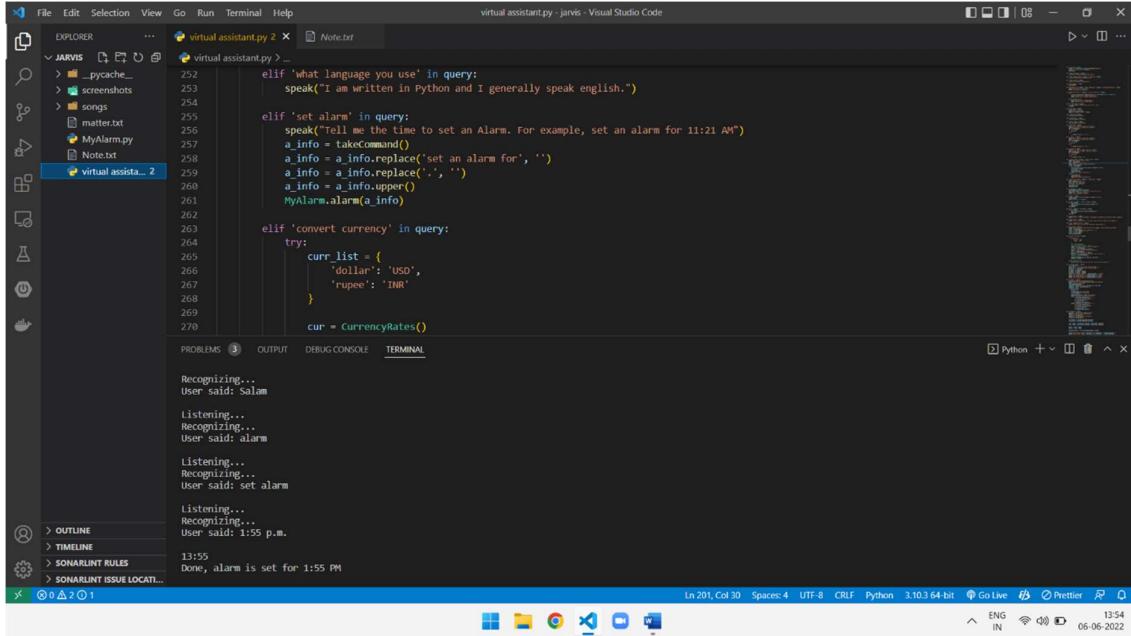
```
virtual assistant.py 2
...
214     elif 'show me the notes' in query or 'read notes' in query:
215         speak("Reading Notes")
216         file = open("Note.txt", "r")
217         data_note = file.readlines()
218         # for points in data_note:
219         print(data_note)
220         speak(data_note)
221
222
223     elif 'screenshot' in query:
224         myScreenshot=pyautogui.screenshot()
225         myScreenshot.save('D:\wipro\jarvis\screenshots\1.png')
226         speak("Your screenshot is saved")
227         print("Your screenshot is saved")
228
229     elif 'date' in query:
230         strDate = datetime.datetime.today().strftime('%Y-%m-%d')
231         print(strDate)
232         speak(f"The date is {strDate}")
...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Recycle bin is already Empty.
Listening...
Recognizing...
User said: make a note
Listening...
Recognizing...
User said: this is Chaitanya Kumar I am from Vijayawada
Listening...
Recognizing...
Say that again please...
Listening...
Recognizing...
User said: show me the notes
User said: this is Chaitanya Kumar I am from Vijayawada
[ '\n', 'this is Chaitanya Kumar I am from Vijayawada' ]
Ls 201, Col 30  Spaces: 4  UTF-8  CRLF  Python  3.10.3 64-bit  Go Live  Prettier  13:53  ENG IN  06-06-2022
```

6.3.15 Read Notes:

The screenshot shows the Visual Studio Code interface with the 'virtual assistant.py' file open. The code handles user input for reading notes. It includes logic to speak the note, write it to a file, and provide feedback. The terminal window shows the program's interaction with the user, including speech recognition and synthesis.

```
virtual assistant.py 2
...
212     file.write(note)
213     speak("Point noted successfully.")
214
215     elif 'show me the notes' in query or 'read notes' in query:
216         speak("Reading Notes")
217         file = open("Note.txt", "r")
218         data_note = file.readlines()
219         # for points in data_note:
220         print(data_note)
221         speak(data_note)
222
223
224     elif 'screenshot' in query:
225         myScreenshot=pyautogui.screenshot()
226         myScreenshot.save('D:\wipro\jarvis\screenshots\1.png')
227         speak("Your screenshot is saved")
228         print("Your screenshot is saved")
229
230     elif 'date' in query:
231         strDate = datetime.datetime.today().strftime('%Y-%m-%d')
...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Recycle bin is already Empty.
Listening...
Recognizing...
User said: empty recycle bin
Listening...
Recognizing...
User said: make a note
Listening...
Recognizing...
User said: this is Chaitanya Kumar I am from Vijayawada
Listening...
Recognizing...
Say that again please...
Listening...
Ls 201, Col 30  Spaces: 4  UTF-8  CRLF  Python  3.10.3 64-bit  Go Live  Prettier  13:53  ENG IN  06-06-2022
```

6.3.16 Setting Alarm:



The screenshot shows the Visual Studio Code interface with the file `virtual assistant.py` open. The code handles user queries related to language, alarms, and currency conversion. In the terminal, the user says "alarm" and the program responds by setting an alarm for 1:55 PM.

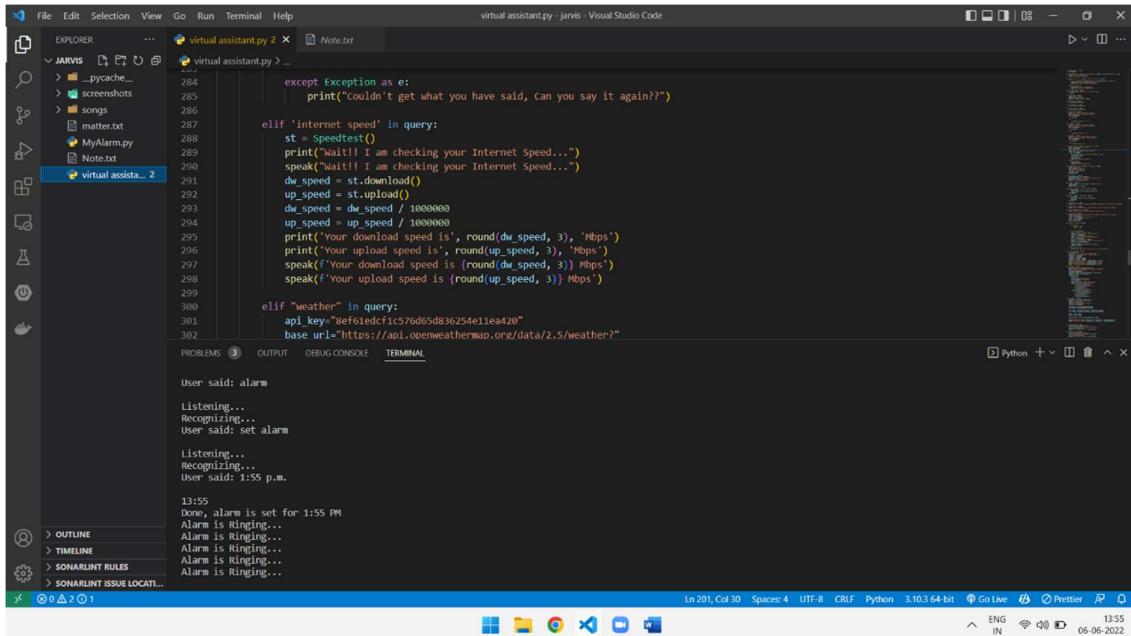
```
elif 'what language you use' in query:
    speak("I am written in Python and I generally speak english.")

elif 'set alarm' in query:
    speak("tell me the time to set an Alarm. For example, set an alarm for 11:21 AM")
    a_info = takeCommand()
    a_info = a_info.replace('set an alarm for', '')
    a_info = a_info.replace('.', '')
    a_info = a_info.upper()
    myAlarm.alarm(a_info)

elif 'convert currency' in query:
    try:
        curr_list = {
            'dollar': 'USD',
            'rupee': 'INR'
        }
        cur = CurrencyRates()
```

User said: Salam
Listening...
Recognizing...
User said: alarm
Listening...
Recognizing...
User said: set alarm
Listening...
Recognizing...
User said: 1:55 p.m.
13:55
Done, alarm is set for 1:55 PM

6.3.17 Activating Alarm:



The screenshot shows the Visual Studio Code interface with the file `virtual assistant.py` open. The code handles user queries related to internet speed and weather. In the terminal, the user says "alarm" and the program responds by activating the alarm for 1:55 PM.

```
except Exception as e:
    print("Couldn't get what you have said, Can you say it again??")

elif 'internet speed' in query:
    st = Speedtest()
    print("Wait!! I am checking your Internet Speed...")
    speak("Wait!! I am checking your Internet Speed...")
    dw_speed = st.download()
    up_speed = st.upload()
    dw_speed = dw_speed / 1000000
    up_speed = up_speed / 1000000
    print("Your download speed is", round(dw_speed, 3), 'Mbps')
    print("Your upload speed is", round(up_speed, 3), 'Mbps')
    speak("Your download speed is", round(dw_speed, 3), 'Mbps')
    speak("Your upload speed is", round(up_speed, 3), 'Mbps')

elif "weather" in query:
    api_key="8ef01edcf1c576dd5d836254e11ea420"
    base_url="https://api.openweathermap.org/data/2.5/weather?"
```

User said: alarm
Listening...
Recognizing...
User said: set alarm
Listening...
Recognizing...
User said: 1:55 p.m.
13:55
Done, alarm is set for 1:55 PM
Alarm is Ringing...
Alarm is Ringing...

6.3.18 Checking Live Weather:

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "JARVIS" containing files like ".pycache_0", "screenshots", "songs", "matter.txt", "Myalarm.py", "Notet.txt", and "virtual assistant.py".
- Code Editor:** The active file is "virtual assistant.py" (line 299). The code handles user input for weather and internet speed.
- Terminal:** Displays the program's output:

```
User said: what is my internet speed
Wait!! I am checking your Internet Speed...
Your download speed is 43.239 Mbps
Your upload speed is 41.418 Mbps
Listening...
Recognizing...
User said: what is the live weather

Listening...
Recognizing...
User said: Bangalore

Temperature in kelvin unit = 300.94
humidity (in percentage) = 69
description = few clouds
Listening...
```
- Status Bar:** Shows file statistics (Line 201, Col 30), encoding (UTF-8), and other details (Python 3.10.3 64-bit, Go Live, Prettier).

6.3.19 Checking Internet Speed:

The screenshot shows a Visual Studio Code interface with the following details:

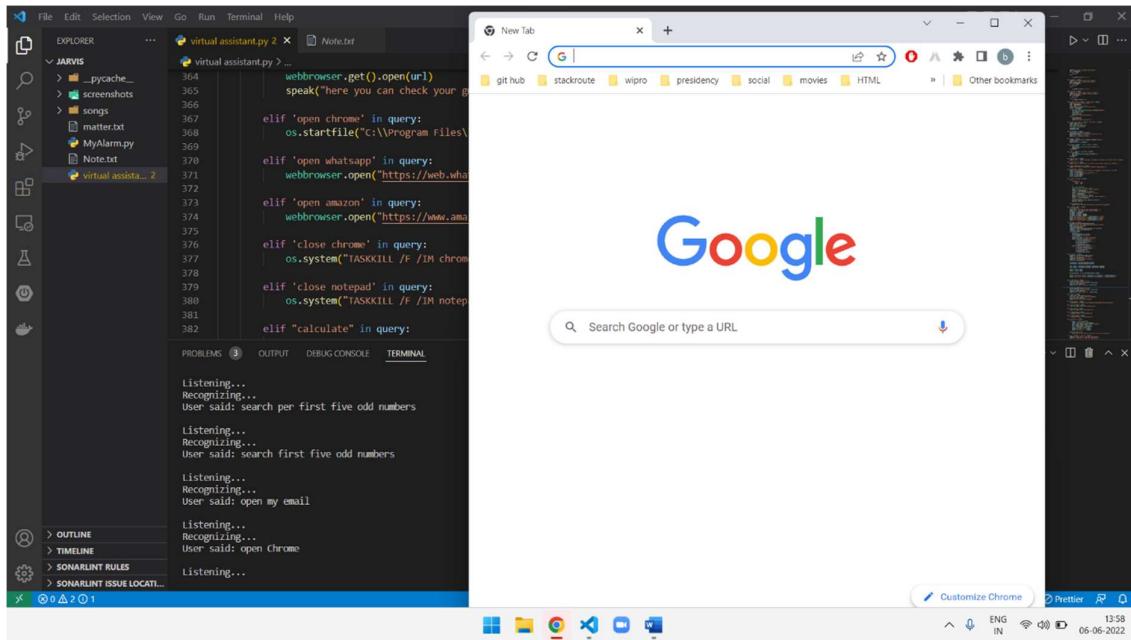
- File Explorer:** Shows a tree view of files and folders. The current file is `virtual assistant.py`. Other visible files include `matter.txt`, `Myalarm.py`, and `Notet.txt`.
- Code Editor:** Displays the `virtual assistant.py` script. The code includes logic for checking internet speed and weather. It uses the `Speedtest` library for speed testing and the `openweathermap.org` API for weather data.
- Terminal:** Shows the command line output of running the script. It includes the PowerShell copyright notice, a message to install the latest PowerShell, and the execution of the Python script. The script initializes the AI assistant, listens for commands, recognizes user input, and provides responses about internet speed.
- Status Bar:** Shows the current file is `virtual assistant.py - Jarvis - Visual Studio Code`. It also displays the line number (Line 201), column number (Col 30), spaces used (Spaces: 4), encoding (UTF-8), file type (Python), version (3.10.3 64-bit), and other system information like Go Live, Prettier, and file icons.

6.3.20 Opening Gmail Inbox:

The screenshot shows a Visual Studio Code interface with a Python script named `virtual assistant.py` open in the editor. The code contains logic to handle various user queries, including opening the Gmail inbox. To the right of the code editor, a web browser window is displayed, showing the Google Workspace login page at `mail.google.com/mail/u/0/#inbox`. The browser's address bar also shows the URL. The system tray at the bottom right indicates the date as 06-06-2022 and the time as 13:58.

```
virtual assistant.py 2
virtual assistant.py > ...
virtual assistant.py:
    358     webbrowser.get().open(url)
    359     speak("Here is what i found for")
    360
    361     elif 'email' in query:
    362         search_term = query.split("for")
    363         url="https://mail.google.com/mail/u/0/#inbox"
    364         webbrowser.get().open(url)
    365         speak("here you can check your g")
    366
    367     elif 'open chrome' in query:
    368         os.startfile('c:\\Program Files\\')
    369
    370     elif 'open whatsapp' in query:
    371         webbrowser.open('https://web.whatsapp.com')
    372
    373     elif 'open amazon' in query:
    374         webbrowser.open('https://www.amazon.in')
    375
    376     elif 'close chrome' in query:
    377         os.system("taskkill /F /IM chrome.exe")
    378
    379
    380
    381
    382
    383
    384
    385
    386
    387
    388
    389
    390
    391
    392
    393
    394
    395
    396
    397
    398
    399
    400
    401
    402
    403
    404
    405
    406
    407
    408
    409
    410
    411
    412
    413
    414
    415
    416
    417
    418
    419
    420
    421
    422
    423
    424
    425
    426
    427
    428
    429
    430
    431
    432
    433
    434
    435
    436
    437
    438
    439
    440
    441
    442
    443
    444
    445
    446
    447
    448
    449
    450
    451
    452
    453
    454
    455
    456
    457
    458
    459
    460
    461
    462
    463
    464
    465
    466
    467
    468
    469
    470
    471
    472
    473
    474
    475
    476
    477
    478
    479
    480
    481
    482
    483
    484
    485
    486
    487
    488
    489
    490
    491
    492
    493
    494
    495
    496
    497
    498
    499
    500
    501
    502
    503
    504
    505
    506
    507
    508
    509
    510
    511
    512
    513
    514
    515
    516
    517
    518
    519
    520
    521
    522
    523
    524
    525
    526
    527
    528
    529
    530
    531
    532
    533
    534
    535
    536
    537
    538
    539
    540
    541
    542
    543
    544
    545
    546
    547
    548
    549
    550
    551
    552
    553
    554
    555
    556
    557
    558
    559
    560
    561
    562
    563
    564
    565
    566
    567
    568
    569
    570
    571
    572
    573
    574
    575
    576
    577
    578
    579
    580
    581
    582
    583
    584
    585
    586
    587
    588
    589
    590
    591
    592
    593
    594
    595
    596
    597
    598
    599
    600
    601
    602
    603
    604
    605
    606
    607
    608
    609
    610
    611
    612
    613
    614
    615
    616
    617
    618
    619
    620
    621
    622
    623
    624
    625
    626
    627
    628
    629
    630
    631
    632
    633
    634
    635
    636
    637
    638
    639
    640
    641
    642
    643
    644
    645
    646
    647
    648
    649
    650
    651
    652
    653
    654
    655
    656
    657
    658
    659
    660
    661
    662
    663
    664
    665
    666
    667
    668
    669
    670
    671
    672
    673
    674
    675
    676
    677
    678
    679
    680
    681
    682
    683
    684
    685
    686
    687
    688
    689
    690
    691
    692
    693
    694
    695
    696
    697
    698
    699
    700
    701
    702
    703
    704
    705
    706
    707
    708
    709
    710
    711
    712
    713
    714
    715
    716
    717
    718
    719
    720
    721
    722
    723
    724
    725
    726
    727
    728
    729
    730
    731
    732
    733
    734
    735
    736
    737
    738
    739
    740
    741
    742
    743
    744
    745
    746
    747
    748
    749
    750
    751
    752
    753
    754
    755
    756
    757
    758
    759
    760
    761
    762
    763
    764
    765
    766
    767
    768
    769
    770
    771
    772
    773
    774
    775
    776
    777
    778
    779
    780
    781
    782
    783
    784
    785
    786
    787
    788
    789
    790
    791
    792
    793
    794
    795
    796
    797
    798
    799
    800
    801
    802
    803
    804
    805
    806
    807
    808
    809
    810
    811
    812
    813
    814
    815
    816
    817
    818
    819
    820
    821
    822
    823
    824
    825
    826
    827
    828
    829
    830
    831
    832
    833
    834
    835
    836
    837
    838
    839
    840
    841
    842
    843
    844
    845
    846
    847
    848
    849
    850
    851
    852
    853
    854
    855
    856
    857
    858
    859
    860
    861
    862
    863
    864
    865
    866
    867
    868
    869
    870
    871
    872
    873
    874
    875
    876
    877
    878
    879
    880
    881
    882
    883
    884
    885
    886
    887
    888
    889
    890
    891
    892
    893
    894
    895
    896
    897
    898
    899
    900
    901
    902
    903
    904
    905
    906
    907
    908
    909
    910
    911
    912
    913
    914
    915
    916
    917
    918
    919
    920
    921
    922
    923
    924
    925
    926
    927
    928
    929
    930
    931
    932
    933
    934
    935
    936
    937
    938
    939
    940
    941
    942
    943
    944
    945
    946
    947
    948
    949
    950
    951
    952
    953
    954
    955
    956
    957
    958
    959
    960
    961
    962
    963
    964
    965
    966
    967
    968
    969
    970
    971
    972
    973
    974
    975
    976
    977
    978
    979
    980
    981
    982
    983
    984
    985
    986
    987
    988
    989
    990
    991
    992
    993
    994
    995
    996
    997
    998
    999
    1000
    1001
    1002
    1003
    1004
    1005
    1006
    1007
    1008
    1009
    1010
    1011
    1012
    1013
    1014
    1015
    1016
    1017
    1018
    1019
    1020
    1021
    1022
    1023
    1024
    1025
    1026
    1027
    1028
    1029
    1030
    1031
    1032
    1033
    1034
    1035
    1036
    1037
    1038
    1039
    1040
    1041
    1042
    1043
    1044
    1045
    1046
    1047
    1048
    1049
    1050
    1051
    1052
    1053
    1054
    1055
    1056
    1057
    1058
    1059
    1060
    1061
    1062
    1063
    1064
    1065
    1066
    1067
    1068
    1069
    1070
    1071
    1072
    1073
    1074
    1075
    1076
    1077
    1078
    1079
    1080
    1081
    1082
    1083
    1084
    1085
    1086
    1087
    1088
    1089
    1090
    1091
    1092
    1093
    1094
    1095
    1096
    1097
    1098
    1099
    1100
    1101
    1102
    1103
    1104
    1105
    1106
    1107
    1108
    1109
    1110
    1111
    1112
    1113
    1114
    1115
    1116
    1117
    1118
    1119
    1120
    1121
    1122
    1123
    1124
    1125
    1126
    1127
    1128
    1129
    1130
    1131
    1132
    1133
    1134
    1135
    1136
    1137
    1138
    1139
    1140
    1141
    1142
    1143
    1144
    1145
    1146
    1147
    1148
    1149
    1150
    1151
    1152
    1153
    1154
    1155
    1156
    1157
    1158
    1159
    1160
    1161
    1162
    1163
    1164
    1165
    1166
    1167
    1168
    1169
    1170
    1171
    1172
    1173
    1174
    1175
    1176
    1177
    1178
    1179
    1180
    1181
    1182
    1183
    1184
    1185
    1186
    1187
    1188
    1189
    1190
    1191
    1192
    1193
    1194
    1195
    1196
    1197
    1198
    1199
    1200
    1201
    1202
    1203
    1204
    1205
    1206
    1207
    1208
    1209
    1210
    1211
    1212
    1213
    1214
    1215
    1216
    1217
    1218
    1219
    1220
    1221
    1222
    1223
    1224
    1225
    1226
    1227
    1228
    1229
    1230
    1231
    1232
    1233
    1234
    1235
    1236
    1237
    1238
    1239
    1240
    1241
    1242
    1243
    1244
    1245
    1246
    1247
    1248
    1249
    1250
    1251
    1252
    1253
    1254
    1255
    1256
    1257
    1258
    1259
    1260
    1261
    1262
    1263
    1264
    1265
    1266
    1267
    1268
    1269
    1270
    1271
    1272
    1273
    1274
    1275
    1276
    1277
    1278
    1279
    1280
    1281
    1282
    1283
    1284
    1285
    1286
    1287
    1288
    1289
    1290
    1291
    1292
    1293
    1294
    1295
    1296
    1297
    1298
    1299
    1300
    1301
    1302
    1303
    1304
    1305
    1306
    1307
    1308
    1309
    1310
    1311
    1312
    1313
    1314
    1315
    1316
    1317
    1318
    1319
    1320
    1321
    1322
    1323
    1324
    1325
    1326
    1327
    1328
    1329
    1330
    1331
    1332
    1333
    1334
    1335
    1336
    1337
    1338
    1339
    1340
    1341
    1342
    1343
    1344
    1345
    1346
    1347
    1348
    1349
    1350
    1351
    1352
    1353
    1354
    1355
    1356
    1357
    1358
    1359
    1360
    1361
    1362
    1363
    1364
    1365
    1366
    1367
    1368
    1369
    1370
    1371
    1372
    1373
    1374
    1375
    1376
    1377
    1378
    1379
    1380
    1381
    1382
    1383
    1384
    1385
    1386
    1387
    1388
    1389
    1390
    1391
    1392
    1393
    1394
    1395
    1396
    1397
    1398
    1399
    1400
    1401
    1402
    1403
    1404
    1405
    1406
    1407
    1408
    1409
    1410
    1411
    1412
    1413
    1414
    1415
    1416
    1417
    1418
    1419
    1420
    1421
    1422
    1423
    1424
    1425
    1426
    1427
    1428
    1429
    1430
    1431
    1432
    1433
    1434
    1435
    1436
    1437
    1438
    1439
    1440
    1441
    1442
    1443
    1444
    1445
    1446
    1447
    1448
    1449
    1450
    1451
    1452
    1453
    1454
    1455
    1456
    1457
    1458
    1459
    1460
    1461
    1462
    1463
    1464
    1465
    1466
    1467
    1468
    1469
    1470
    1471
    1472
    1473
    1474
    1475
    1476
    1477
    1478
    1479
    1480
    1481
    1482
    1483
    1484
    1485
    1486
    1487
    1488
    1489
    1490
    1491
    1492
    1493
    1494
    1495
    1496
    1497
    1498
    1499
    1500
    1501
    1502
    1503
    1504
    1505
    1506
    1507
    1508
    1509
    1510
    1511
    1512
    1513
    1514
    1515
    1516
    1517
    1518
    1519
    1520
    1521
    1522
    1523
    1524
    1525
    1526
    1527
    1528
    1529
    1530
    1531
    1532
    1533
    1534
    1535
    1536
    1537
    1538
    1539
    1540
    1541
    1542
    1543
    1544
    1545
    1546
    1547
    1548
    1549
    1550
    1551
    1552
    1553
    1554
    1555
    1556
    1557
    1558
    1559
    1560
    1561
    1562
    1563
    1564
    1565
    1566
    1567
    1568
    1569
    1570
    1571
    1572
    1573
    1574
    1575
    1576
    1577
    1578
    1579
    1580
    1581
    1582
    1583
    1584
    1585
    1586
    1587
    1588
    1589
    1590
    1591
    1592
    1593
    1594
    1595
    1596
    1597
    1598
    1599
    1600
    1601
    1602
    1603
    1604
    1605
    1606
    1607
    1608
    1609
    1610
    1611
    1612
    1613
    1614
    1615
    1616
    1617
    1618
    1619
    1620
    1621
    1622
    1623
    1624
    1625
    1626
    1627
    1628
    1629
    1630
    1631
    1632
    1633
    1634
    1635
    1636
    1637
    1638
    1639
    1640
    1641
    1642
    1643
    1644
    1645
    1646
    1647
    1648
    1649
    1650
    1651
    1652
    1653
    1654
    1655
    1656
    1657
    1658
    1659
    1660
    1661
    1662
    1663
    1664
    1665
    1666
    1667
    1668
    1669
    1670
    1671
    1672
    1673
    1674
    1675
    1676
    1677
    1678
    1679
    1680
    1681
    1682
    1683
    1684
    1685
    1686
    1687
    1688
    1689
    1690
    1691
    1692
    1693
    1694
    1695
    1696
    1697
    1698
    1699
    1700
    1701
    1702
    1703
    1704
    1705
    1706
    1707
    1708
    1709
    1710
    1711
    1712
    1713
    1714
    1715
    1716
    1717
    1718
    1719
    1720
    1721
    1722
    1723
    1724
    1725
    1726
    1727
    1728
    1729
    1730
    1731
    1732
    1733
    1734
    1735
    1736
    1737
    1738
    1739
    1740
    1741
    1742
    1743
    1744
    1745
    1746
    1747
    1748
    1749
    1750
    1751
    1752
    1753
    1754
    1755
    1756
    1757
    1758
    1759
    1760
    1761
    1762
    1763
    1764
    1765
    1766
    1767
    1768
    1769
    1770
    1771
    1772
    1773
    1774
    1775
    1776
    1777
    1778
    1779
    1780
    1781
    1782
    1783
    1784
    1785
    1786
    1787
    1788
    1789
    1790
    1791
    1792
    1793
    1794
    1795
    1796
    1797
    1798
    1799
    1800
    1801
    1802
    1803
    1804
    1805
    1806
    1807
    1808
    1809
    1810
    1811
    1812
    1813
    1814
    1815
    1816
    1817
    1818
    1819
    1820
    1821
    1822
    1823
    1824
    1825
    1826
    1827
    1828
    1829
    1830
    1831
    1832
    1833
    1834
    1835
    1836
    1837
    1838
    1839
    1840
    1841
    1842
    1843
    1844
    1845
    1846
    1847
    1848
    1849
    1850
    1851
    1852
    1853
    1854
    1855
    1856
    1857
    1858
    1859
    1860
    1861
    1862
    1863
    1864
    1865
    1866
    1867
    1868
    1869
    1870
    1871
    1872
    1873
    1874
    1875
    1876
    1877
    1878
    1879
    1880
    1881
    1882
    1883
    1884
    1885
    1886
    1887
    1888
    1889
    1890
    1891
    1892
    1893
    1894
    1895
    1896
    1897
    1898
    1899
    1900
    1901
    1902
    1903
    1904
    1905
    1906
    1907
    1908
    1909
    1910
    1911
    1912
    1913
    1914
    1915
    1916
    1917
    1918
    1919
    1920
    1921
    1922
    1923
    1924
    1925
    1926
    1927
    1928
    1929
    1930
    1931
    1932
    1933
    1934
    1935
    1936
    1937
    1938
    1939
    1940
    1941
    1942
    1943
    1944
    1945
    1946
    1947
    1948
    1949
    1950
    1951
    1952
    1953
    1954
    1955
    1956
    1957
    1958
    1959
    1960
    1961
    1962
    1963
    1964
    1965
    1966
    1967
    1968
    1969
    1970
    1971
    1972
    1973
    1974
    1975
    1976
    1977
    1978
    1979
    1980
    1981
    1982
    1983
    1984
    1985
    1986
    1987
    1988
    1989
    1990
    1991
    1992
    1993
    1994
    1995
    1996
    1997
    1998
    1999
    2000
    2001
    2002
    2003
    2004
    2005
    2006
    2007
    2008
    2009
    2010
    2011
    2012
    2013
    2014
    2015
    2016
    2017
    2018
    2019
    2020
    2021
    2022
```

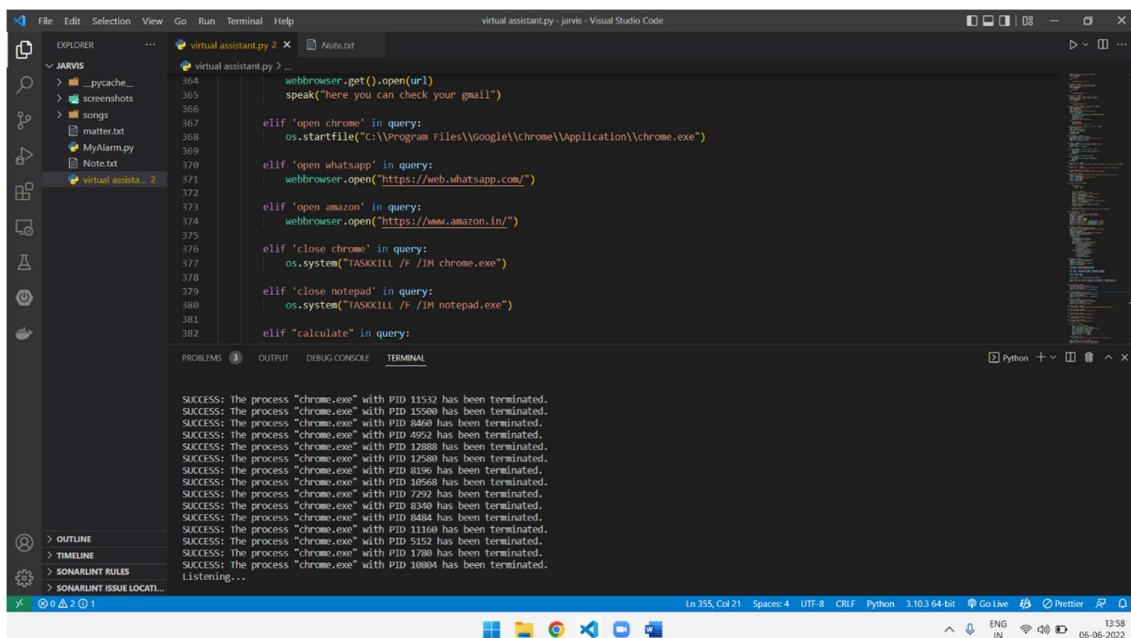
6.3.22 Open Chrome:



The screenshot shows the Visual Studio Code interface with a Python file named `virtual assistant.py` open in the editor. The code contains logic for opening various websites based on user queries. To the right of the code editor is a Microsoft Edge browser window showing the Google search results page.

```
virtual assistant.py
...
364     webbrowser.get().open(url)
365     speak("here you can check your gmail")
366
367 elif 'open chrome' in query:
368     os.startfile("C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe")
369
370 elif 'open whatsapp' in query:
371     webbrowser.open("https://web.whatsapp.com/")
372
373 elif 'open amazon' in query:
374     webbrowser.open("https://www.amazon.in/")
375
376 elif 'close chrome' in query:
377     os.system("TASKKILL /F /IM chrome.exe")
378
379 elif 'close notepad' in query:
380     os.system("TASKKILL /F /IM notepad.exe")
381
382 elif "calculate" in query:
```

6.3.23 Close Chrome:



The screenshot shows the Visual Studio Code interface with the same Python file open. This time, the terminal output at the bottom of the screen shows multiple entries of the message "SUCCESS: The process 'chrome.exe' with PID 11532 has been terminated.", indicating that the script has successfully closed several instances of the Chrome browser.

```
SUCCESS: The process "chrome.exe" with PID 11532 has been terminated.
SUCCESS: The process "chrome.exe" with PID 15508 has been terminated.
SUCCESS: The process "chrome.exe" with PID 15508 has been terminated.
SUCCESS: The process "chrome.exe" with PID 4952 has been terminated.
SUCCESS: The process "chrome.exe" with PID 12888 has been terminated.
SUCCESS: The process "chrome.exe" with PID 12580 has been terminated.
SUCCESS: The process "chrome.exe" with PID 8196 has been terminated.
SUCCESS: The process "chrome.exe" with PID 10568 has been terminated.
SUCCESS: The process "chrome.exe" with PID 7292 has been terminated.
SUCCESS: The process "chrome.exe" with PID 7292 has been terminated.
SUCCESS: The process "chrome.exe" with PID 8084 has been terminated.
SUCCESS: The process "chrome.exe" with PID 11160 has been terminated.
SUCCESS: The process "chrome.exe" with PID 5152 has been terminated.
SUCCESS: The process "chrome.exe" with PID 1788 has been terminated.
SUCCESS: The process "chrome.exe" with PID 10884 has been terminated.
Listening...
```

6.3.24 Using Calculator:

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder structure for "JARVIS" containing files like "pycache_._", "screenshots", "songs", "matter.txt", "Myalarm.py", "Note.txt", and "virtual assistant.py".
- Code Editor:** Displays the content of "virtual assistant.py". The code handles various commands such as closing chrome, notepad, and performing calculations using Wolfram Alpha.
- Terminal:** Shows the output of running the script, which includes terminating multiple instances of "chrome.exe" and responding to user input like "calculate 10 + 10".
- Status Bar:** Shows the current file is "virtual assistant.py - Jarvis - Visual Studio Code", along with other status indicators like "Python 3.10.3 64-bit" and "Go Live".

6.3.25 Exit the Application:

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a project structure for "JARVIS" containing files like `__pycache__.py`, `screenshots`, `songs`, `matter.txt`, `Myalarm.py`, `Note.txt`, and `virtual assistant.py`.
- Code Editor (Top Center):** Displays the content of `virtual assistant.py`. The code includes logic to handle exit commands and speak responses.
- Terminal (Bottom):** Shows the application's log output:
 - SUCCESS: The process "chrome.exe" with PID 5152 has been terminated.
 - SUCCESS: The process "chrome.exe" with PID 1780 has been terminated.
 - SUCCESS: The process "chrome.exe" with PID 10804 has been terminated.
 - Listening...
 - Recognizing...
 - User said: paik corn Plus plan
 - Listening...
 - Recognizing...
 - User said: calculate 10 + 10
 - The answer is 20
 - Listening...
 - Recognizing...
 - User said: exit
 - PS D:\wipro\jarvis>
- Bottom Status Bar:** Shows the current file is `virtual assistant.py - Jarvis - Visual Studio Code`, with status indicators for ENG, IN, 13:58, 06-05-2022, and various developer tools like Go Live, Prettier, and Linting.

7.Conclusion:

In this we have discussed a Voice Assistant developed using python. This assistant currently works as an application based and performs basic tasks like weather updates, stream music, search Wikipedia, open desktop applications, etc. The functionality of the current system is limited to working on application based only. The upcoming updates of this assistant will have machine learning incorporated in the system which will result in better suggestions with IoT to control the nearby devices similar to what Amazon's Alexa does. The modular nature of this project makes it more flexible and easier to add additional features without disturbing current system functionalities. It not only works on human commands but also give responses to the user based on the query being asked or the words spoken by the user such as opening tasks and operations. It is greeting the user the way the user feels more comfortable and feels free to interact with the voice assistant. The application should also eliminate any kind of unnecessary manual work required in the user life of performing every task.

8. References:

- [1] R. Belvin, R. Burns, and C. Hein, “*Development of the HRL route navigation dialogue system*,” in Proceedings of ACL-HLT, 2001
- [2] V. Zue, S. Seneff, J. R. Glass, J. Polifroni, C. Pao, T.J.Hazen, and L.Hetherington, “*JUPITER: A Telephone Based Conversational Interface for Weather Information*,” IEEE Transactions on Speech and Audio Processing, vol. 8, no. 1, pp. 85–96, 2000.
- [3] M. Kolss, D. Bernreuther, M. Paulik, S. St̄ucker, S. Vogel, and A. Waibel, “*Open Domain Speech Recognition & Translation: Lectures and Speeches*,” in Proceedings of ICASSP, 2006.
- [4] D. R. S. Caon, T. Simonnet, P. Sendorek, J. Boudy, and G. Chollet, “*vAssist: The Virtual Interactive Assistant for Daily Homer-Care*,” in Proceedings of pHealth, 2011.
- [5] Crevier, D. (1993). *AI: The Tumultuous Search for Artificial Intelligence*. New York, NY: Basic Books, ISBN 0-465-02997-3.
- [6] Sadun, E., & Sande, S. (2014). *Talking to Siri: Mastering the Language of Apple's Intelligent Assistant*.