ABSTRACT

Personal Assistants, or conversational interfaces, or chat bots reinvent a new way for individuals to interact with computes. 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 works 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 advance 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

5.Project Work Flow

6.Glossary

LIST OF FIGURES

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.Technologies Used:

3.1Visual 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) pyttsx3 — pyttxs3 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 pyttsx3
```

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
```

3.3API'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?"

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

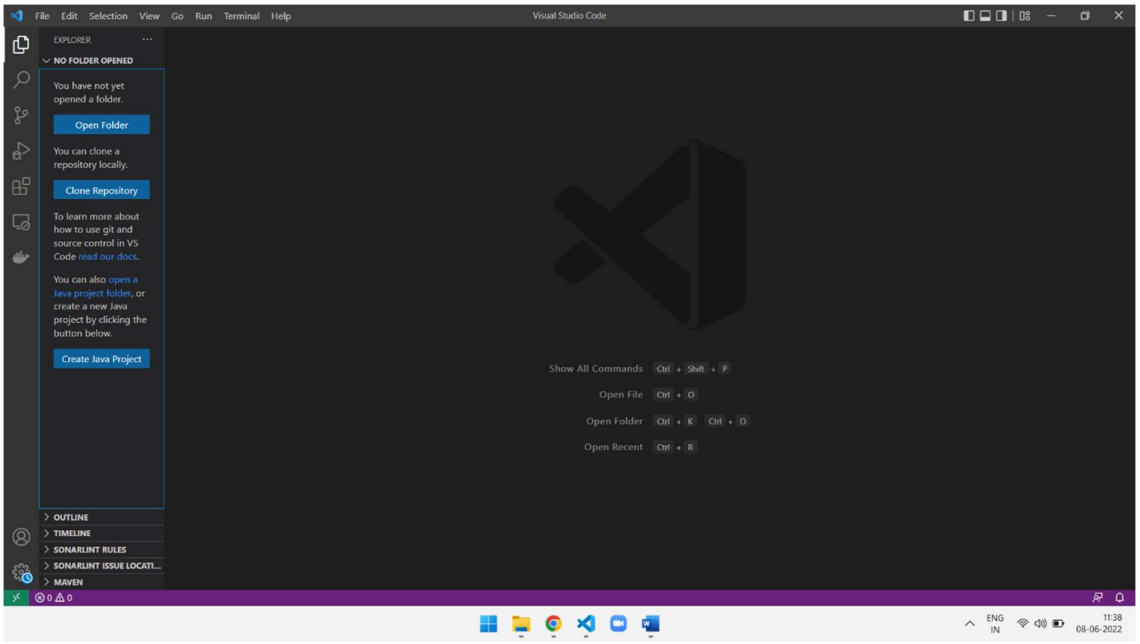
5.1 Problem Statement:

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

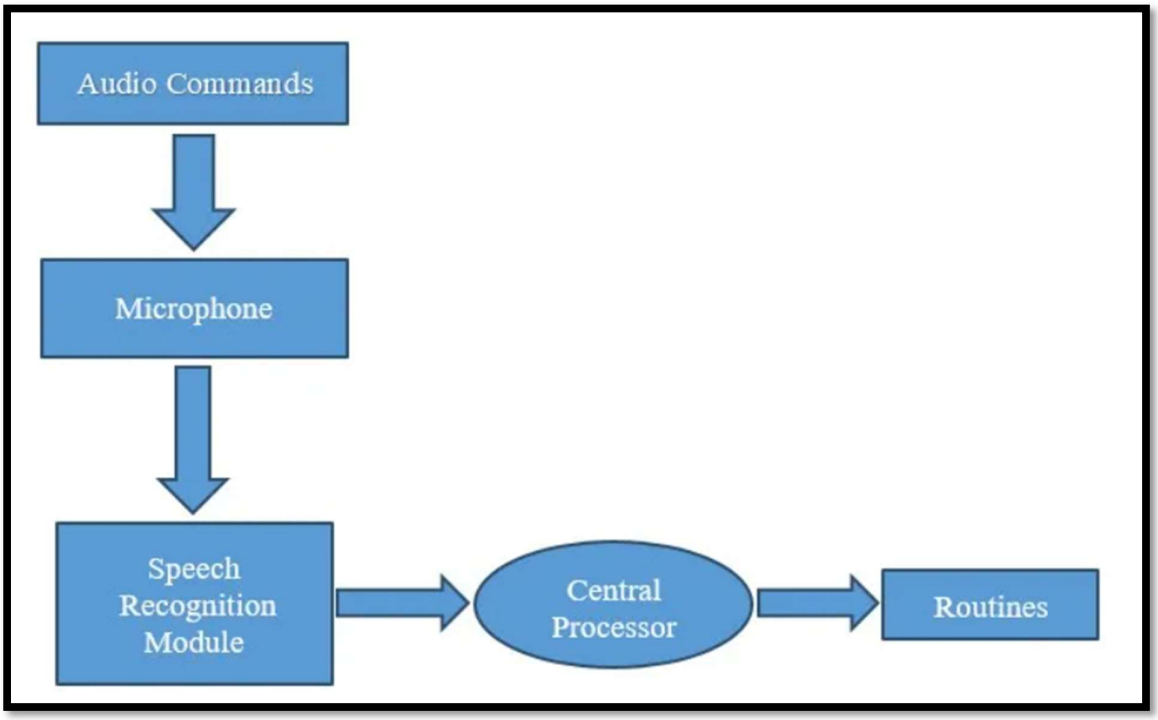
Download the Visual Studio Code installer for Windows.

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:

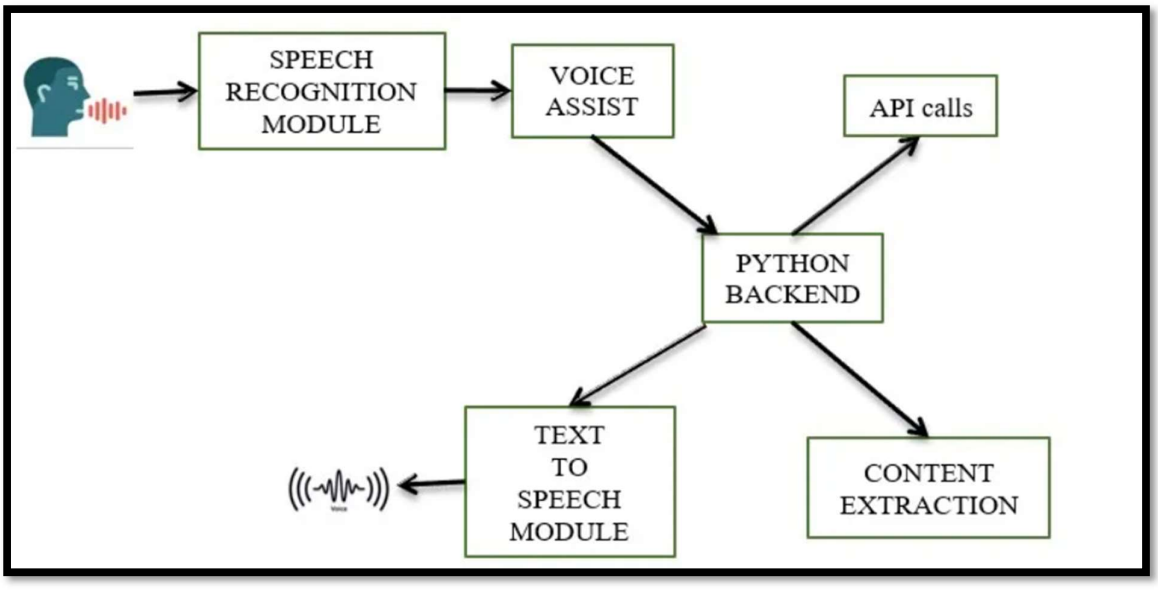
The system uses Google's online speech recognition system for converting speech input to text. The speech input Users can obtain texts from the special corpora organized on the computer network server at the information centre from the microphone is temporarily stored in the system which is then sent 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 pyttsx3

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 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
```

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 devloped 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 you system?")
    speak("Do you want to shutdown you 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 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


engine = pyttsx3.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 you system?")
            speak("Do you want to shutdown you 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="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))

        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:
```

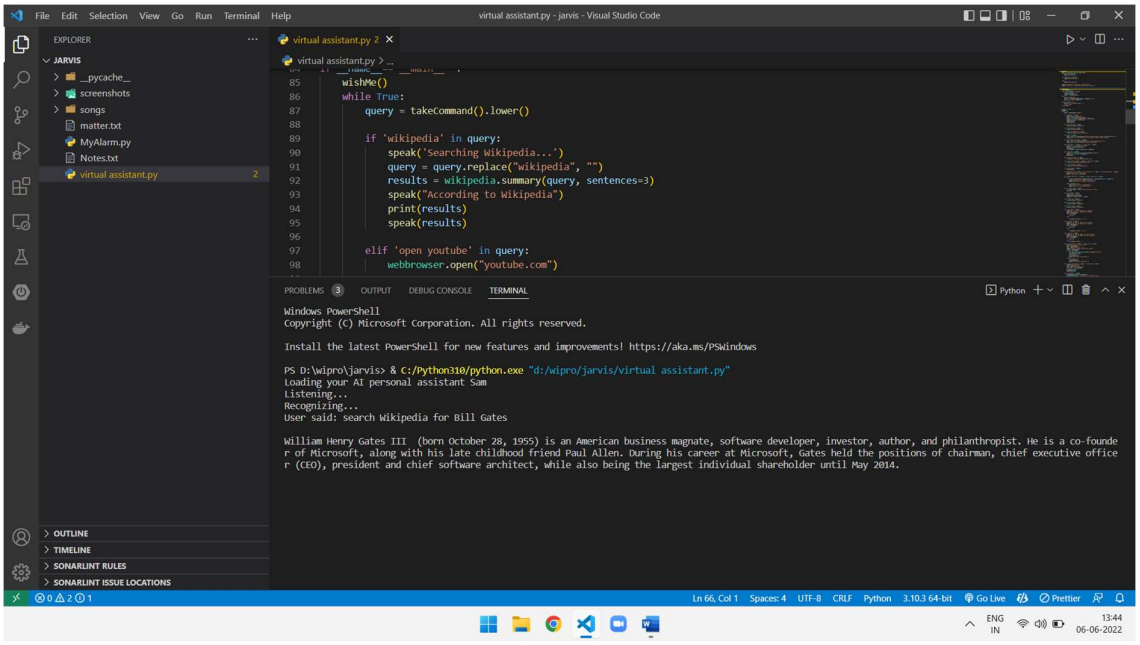
```
            try:
                app_id = "JUGV8R-RXJ4RP7HAG"
                client = wolframalpha.Client(app_id)
                indx = query.lower().split().index('calculate')
                query = query.split()[indx + 1:]
                res = client.query(' '.join(query))
                answer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

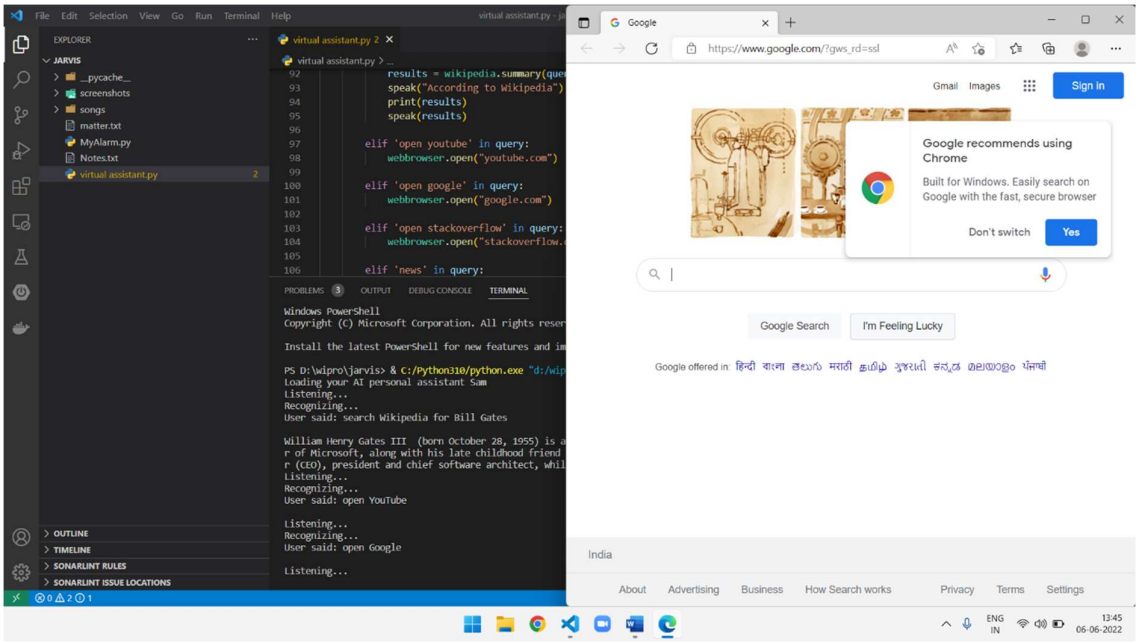
6.3.1 executing the code:

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

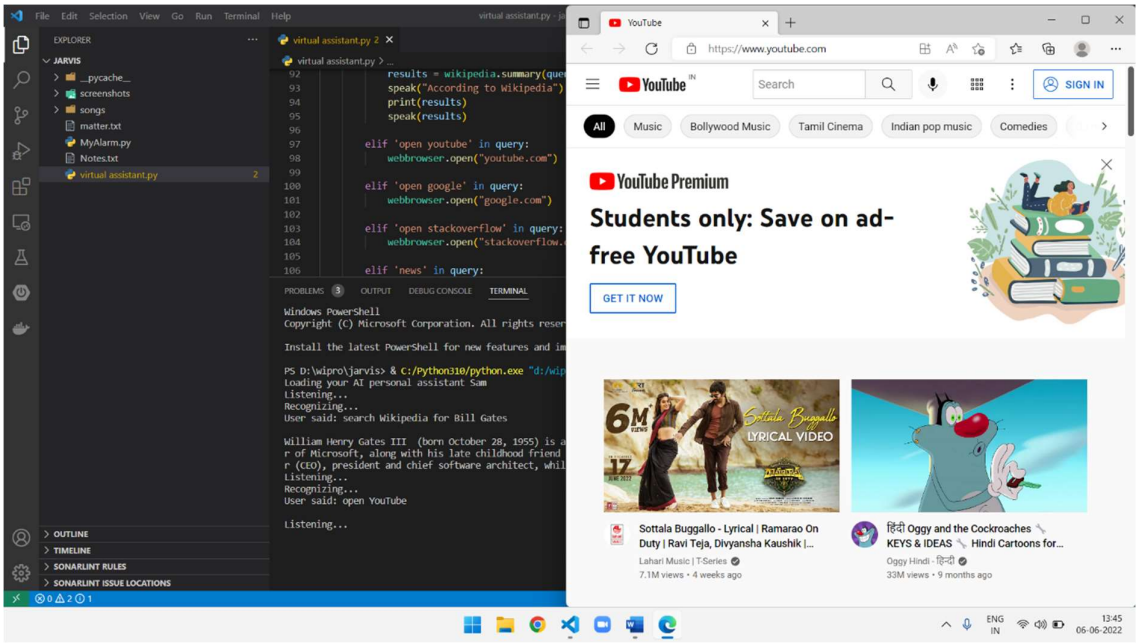
6.3.2 Wikipedia Search:



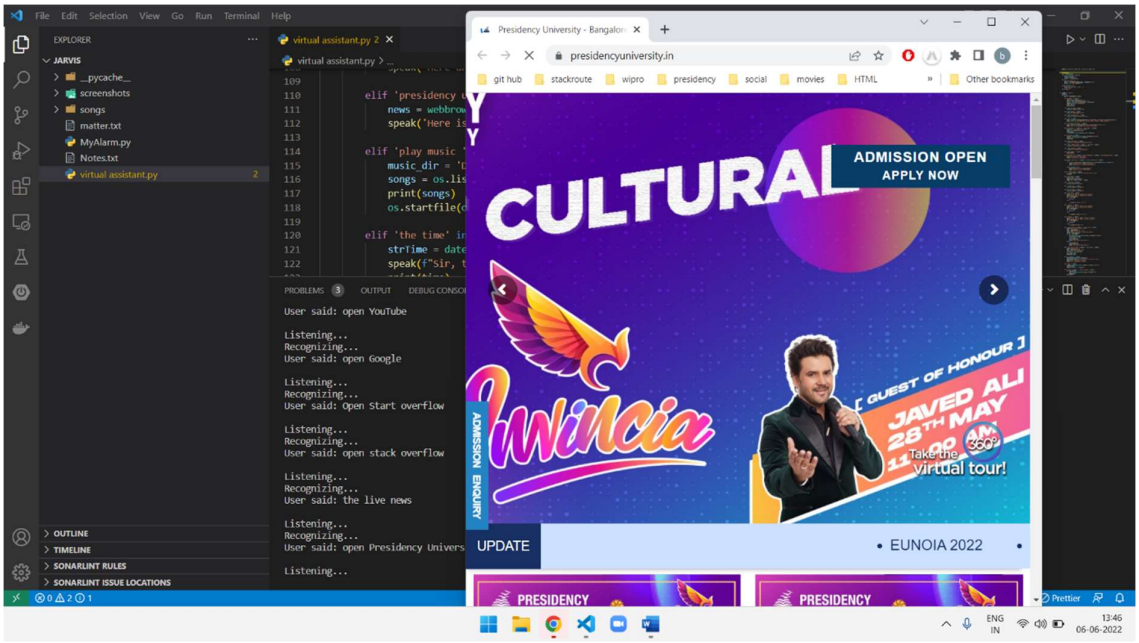
6.3.3 Open Chrome



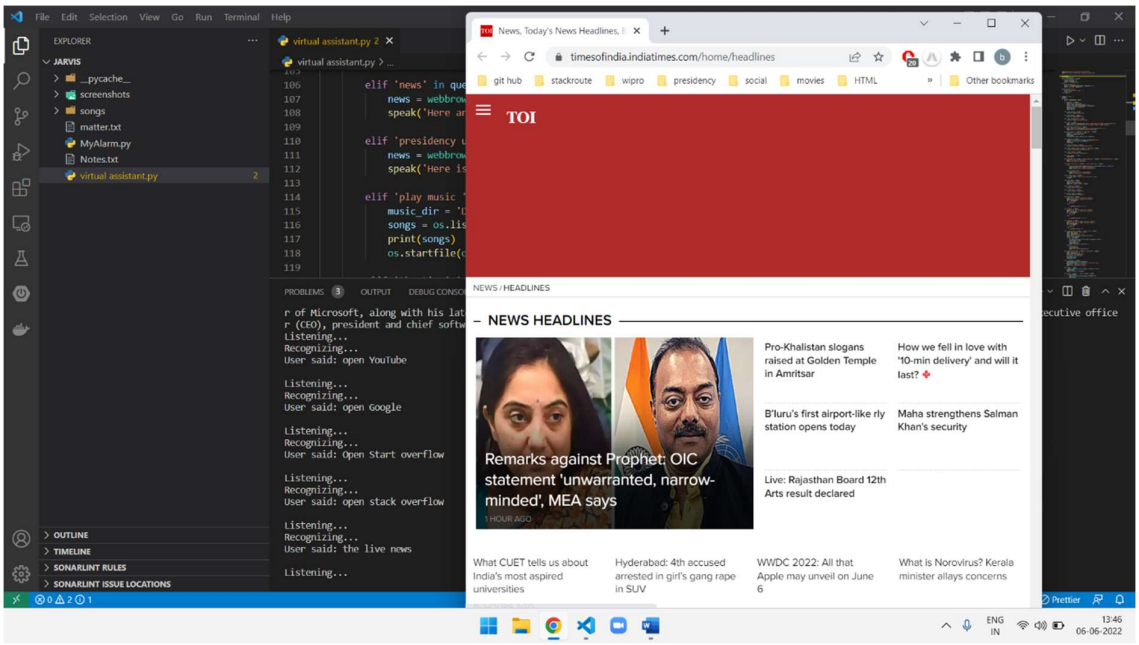
6.3.4 Open YouTube:



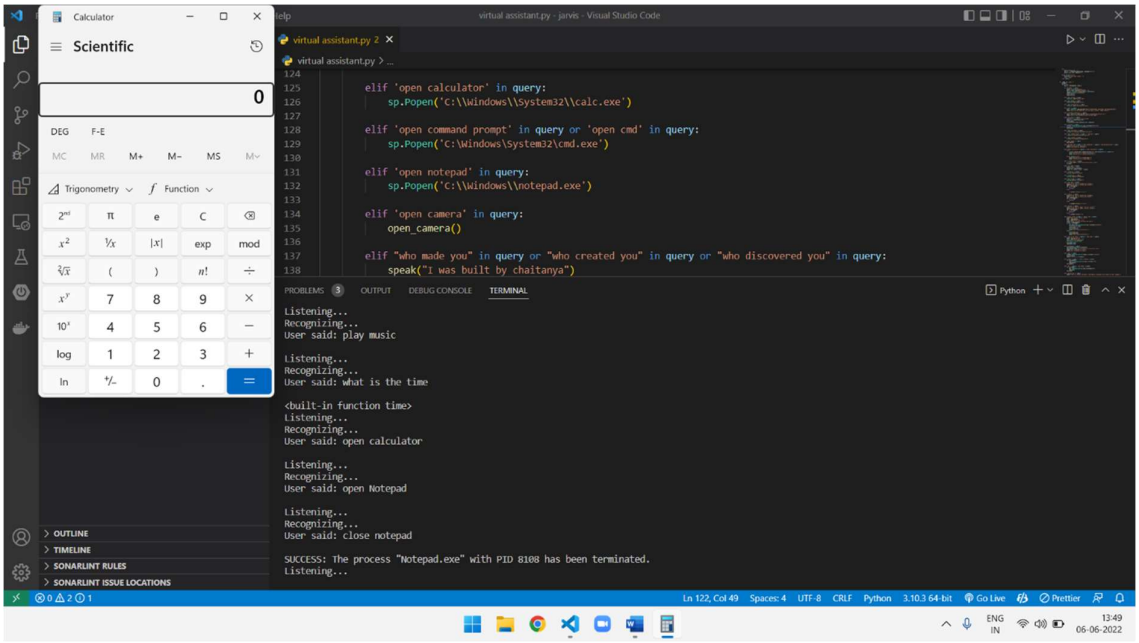
6.3.5 Open Presidency University:



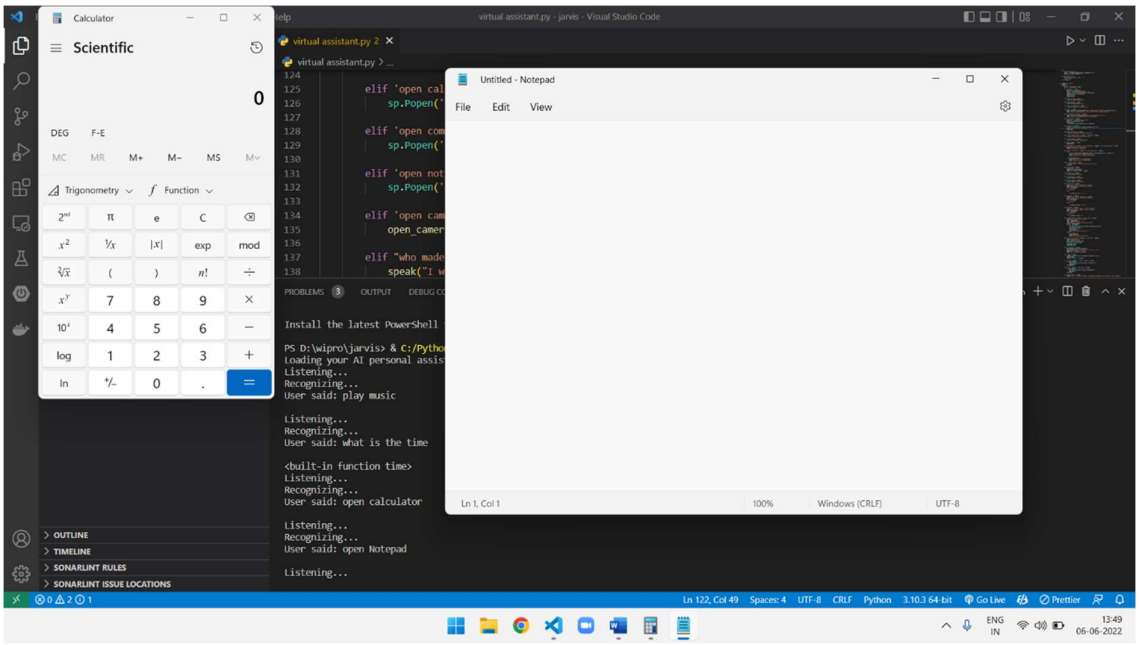
6.3.6 Open News:



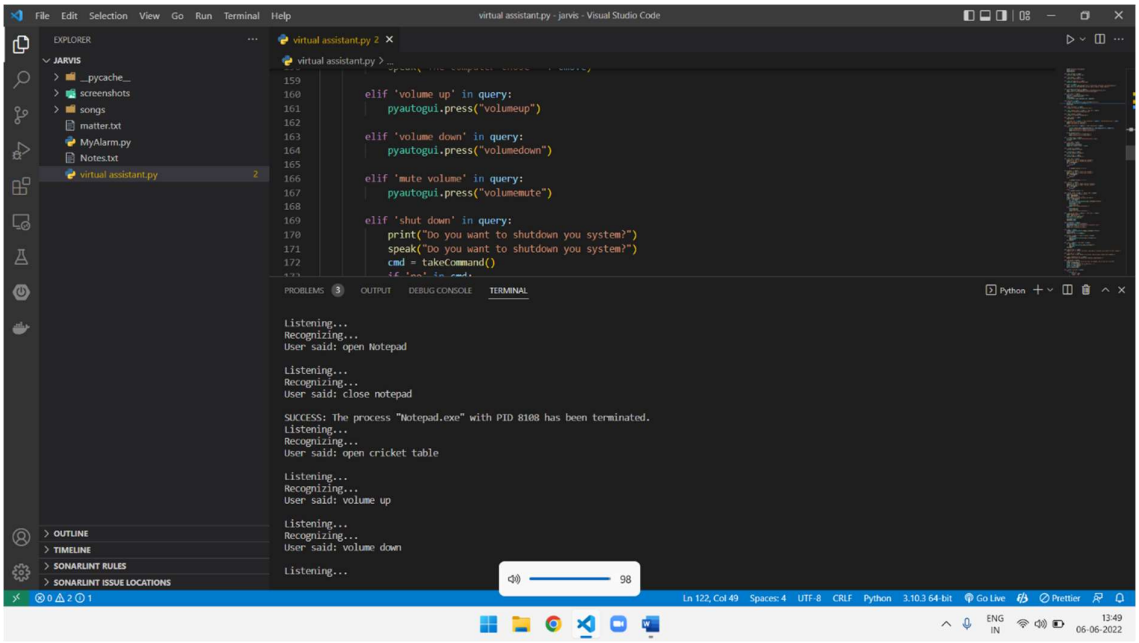
6.3.7 Open Calculator:



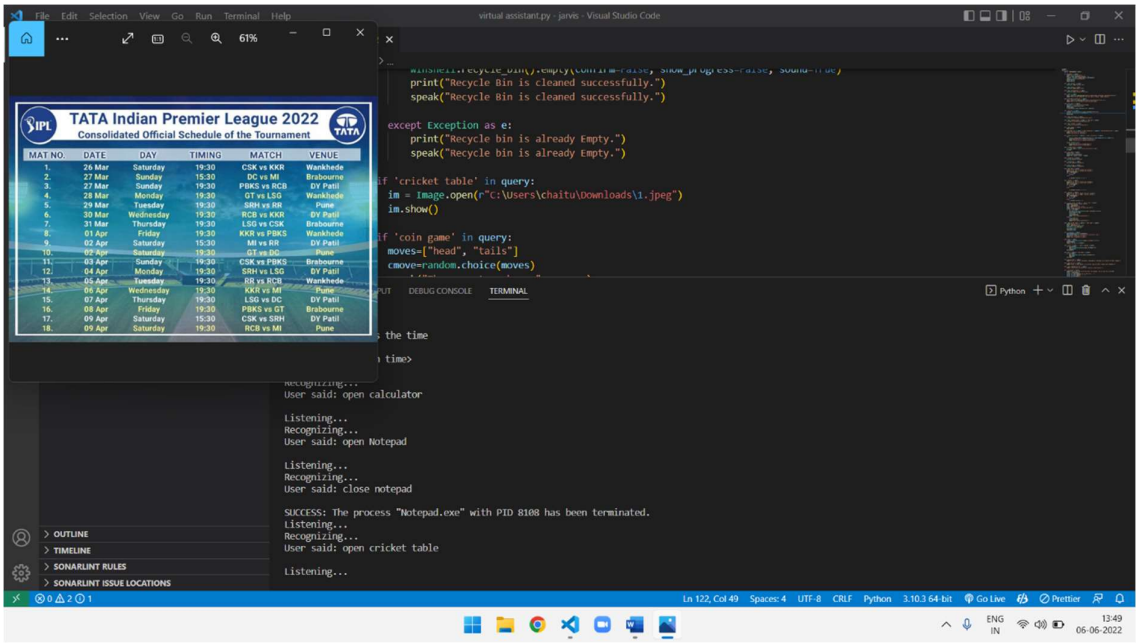
6.3.8 Open Notepad:



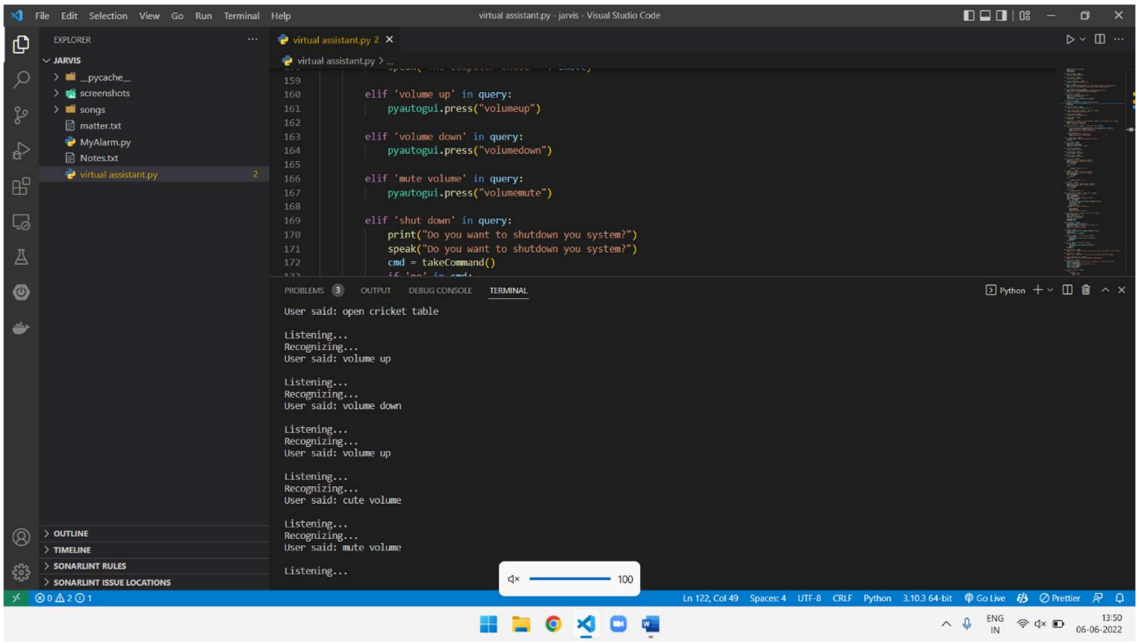
6.3.9 Control Volume Down



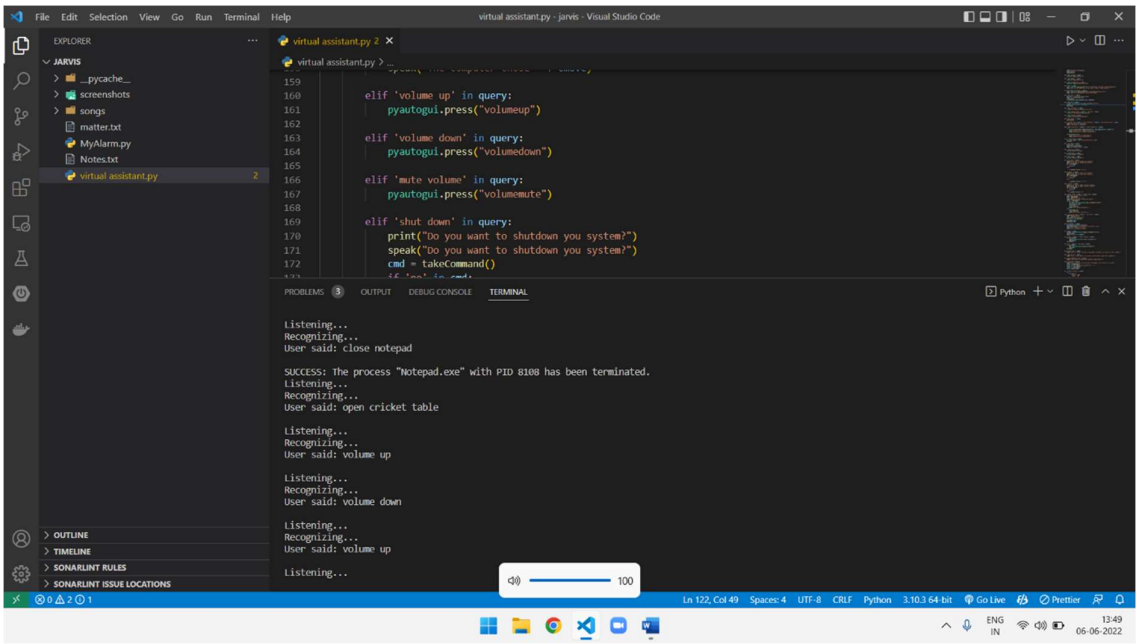
6.3.10 Open default Images:



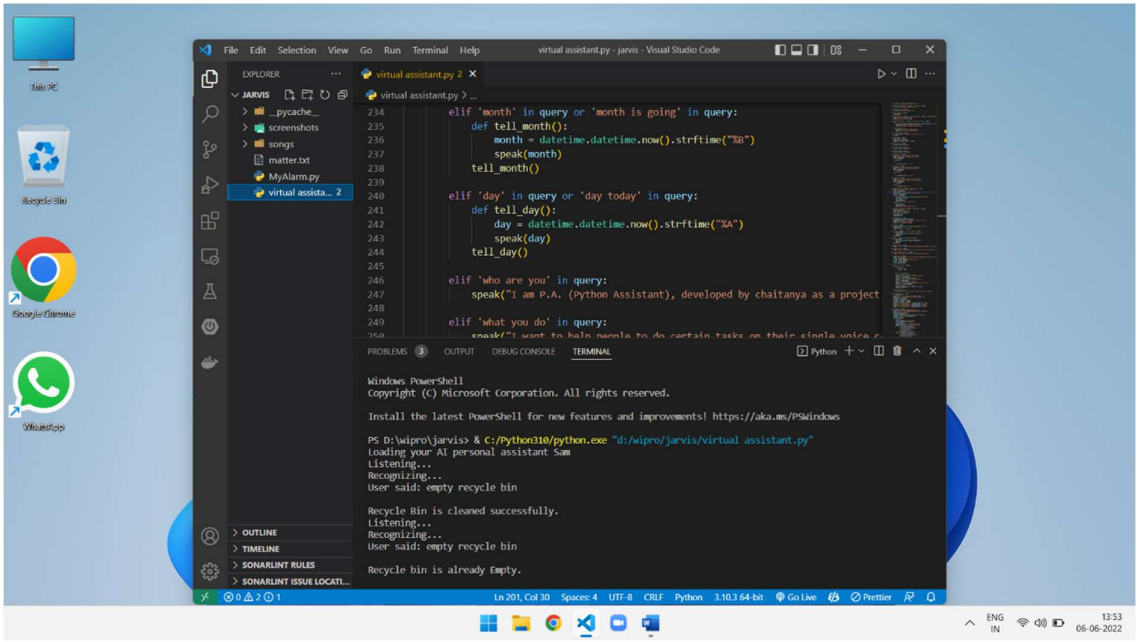
6.3.11 Control Volume Mute:



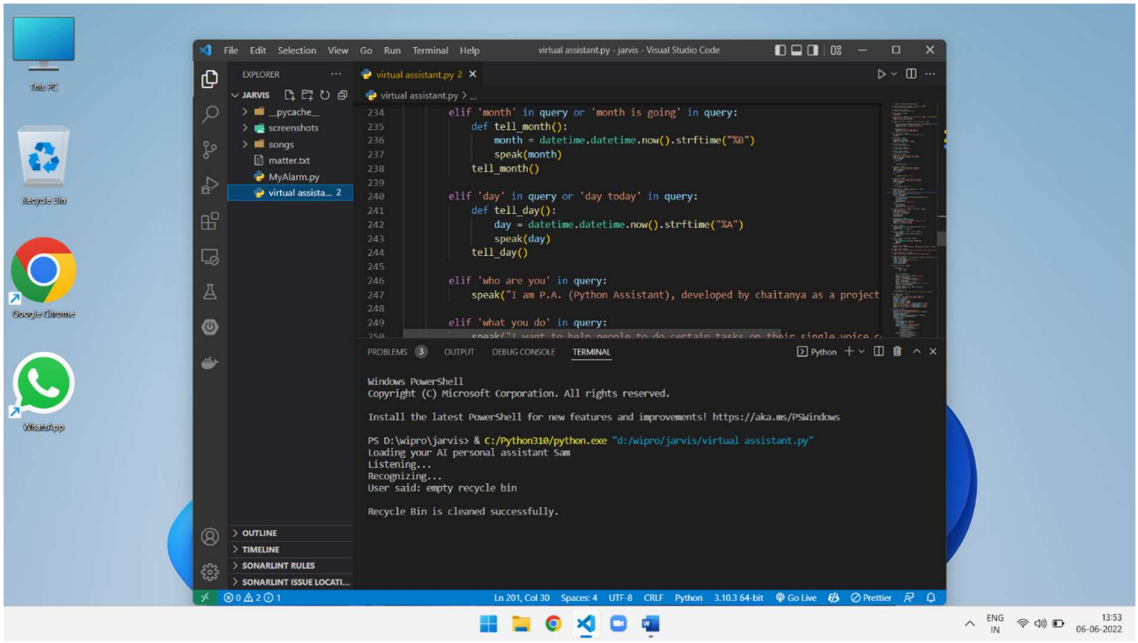
6.3.12 Control Volume Up:



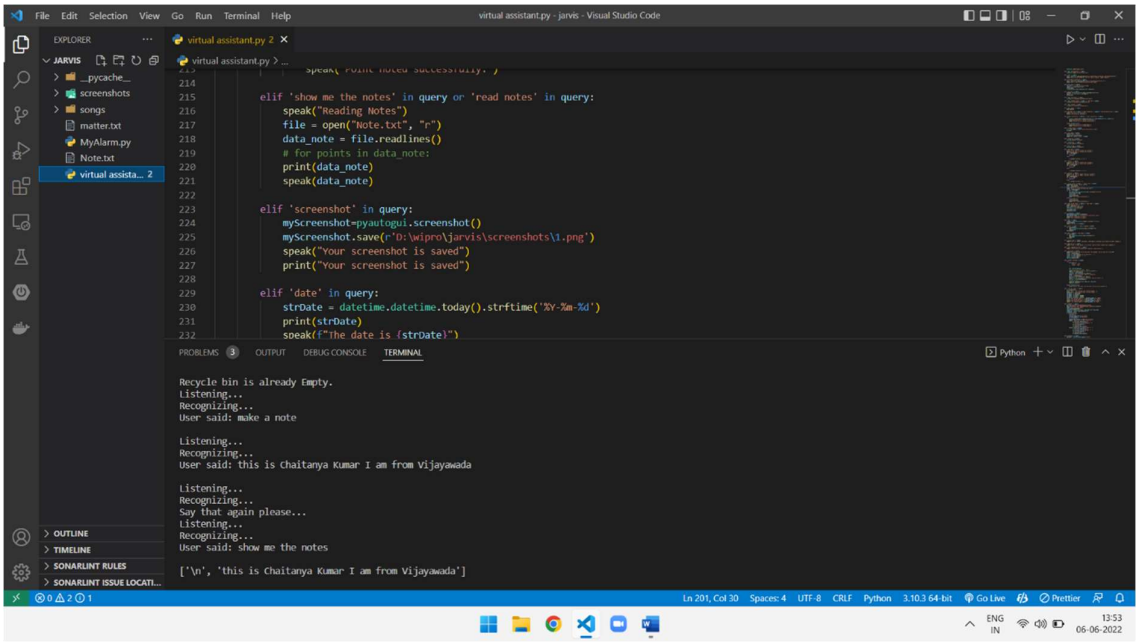
6.3.12 Empty Recycle Bin:



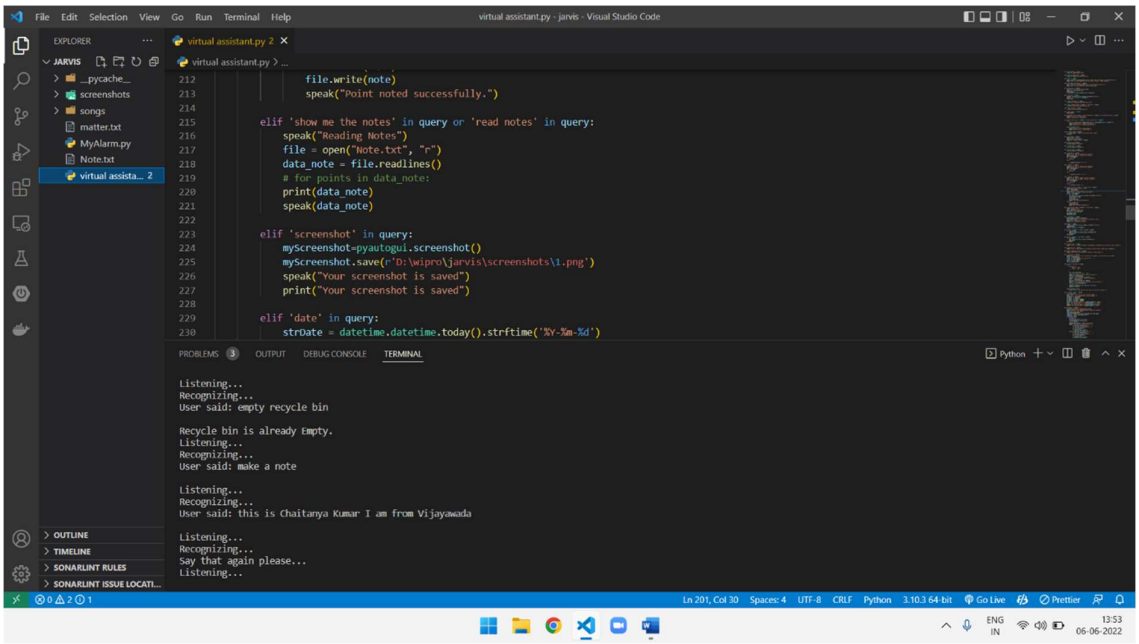
6.3.13 Empty Recycle Bin:



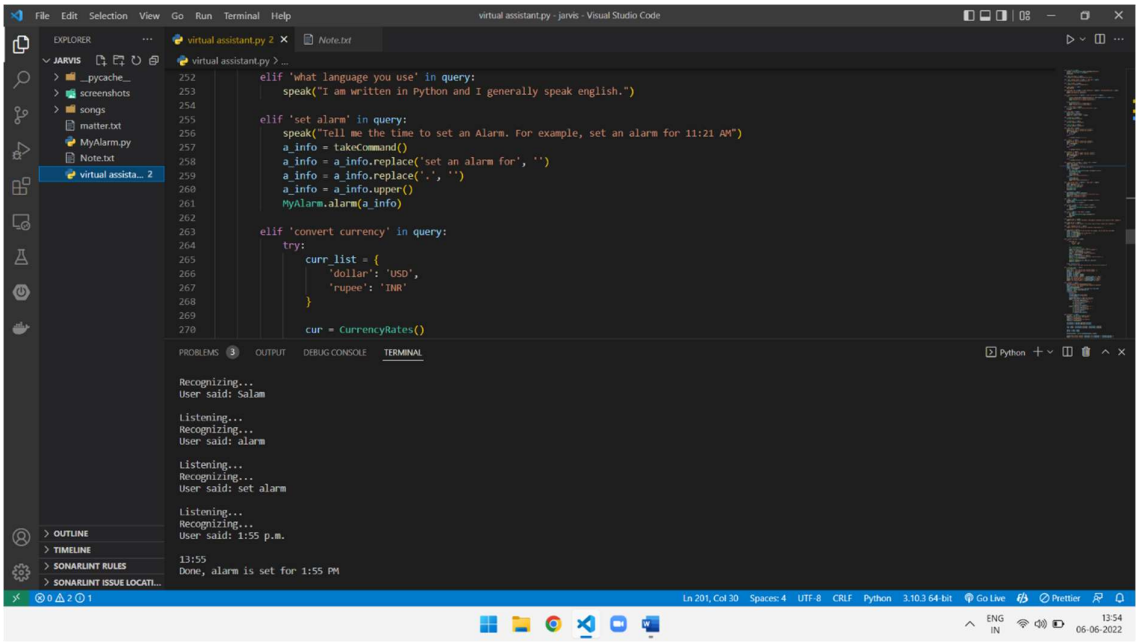
6.3.14 Make a Note:



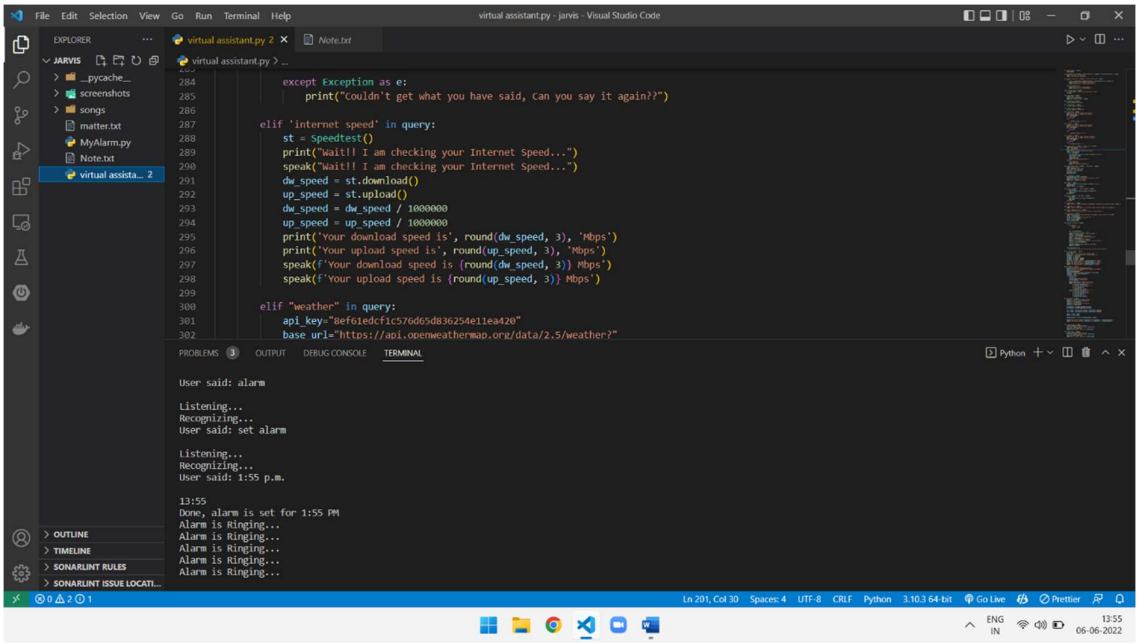
6.3.15 Read Notes:



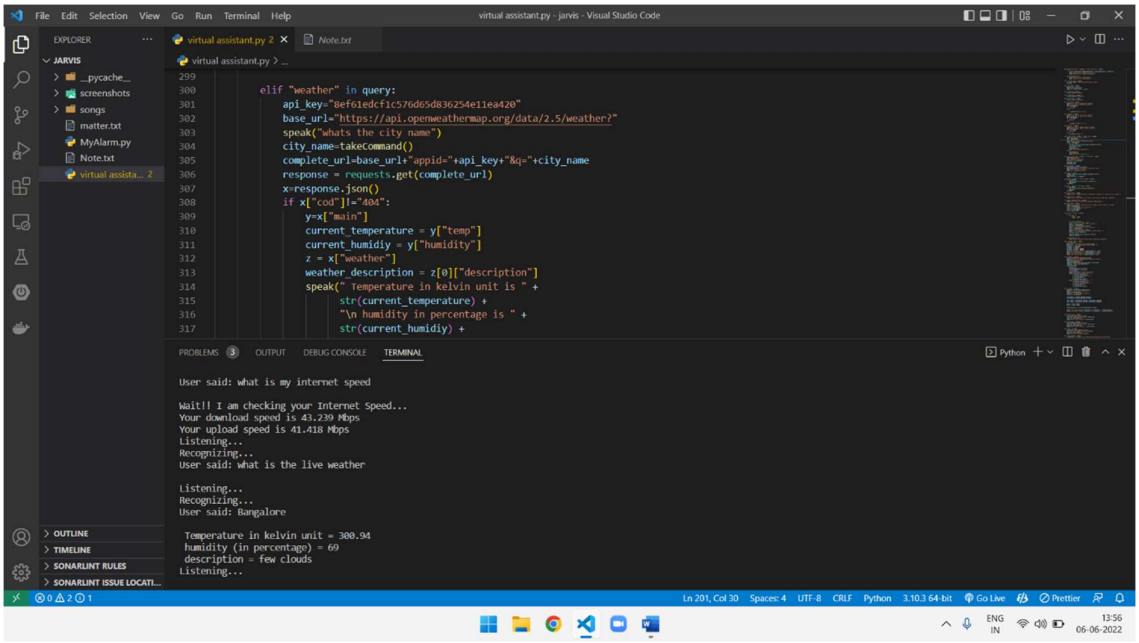
6.3.16 Setting Alarm:



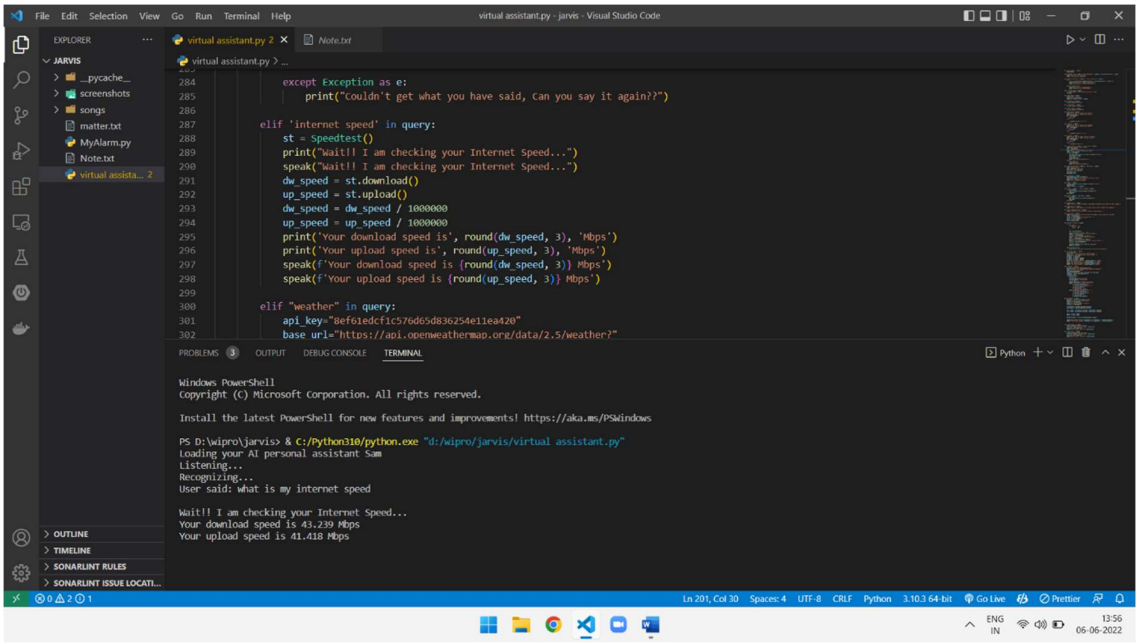
6.3.17 Activating Alarm:



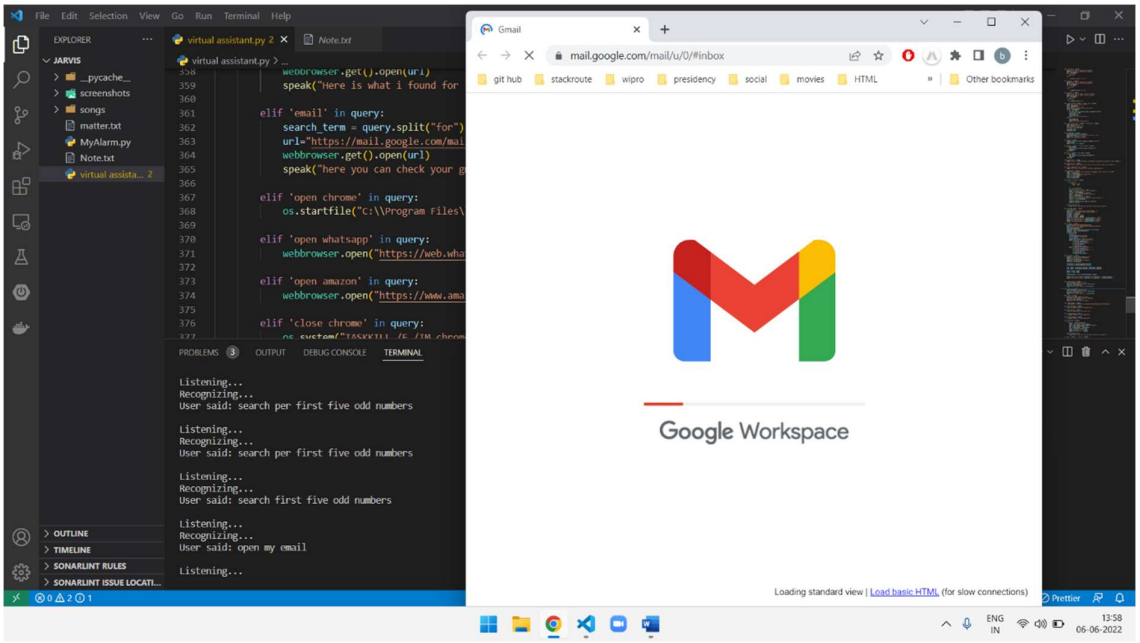
6.3.18 Checking Live Weather:



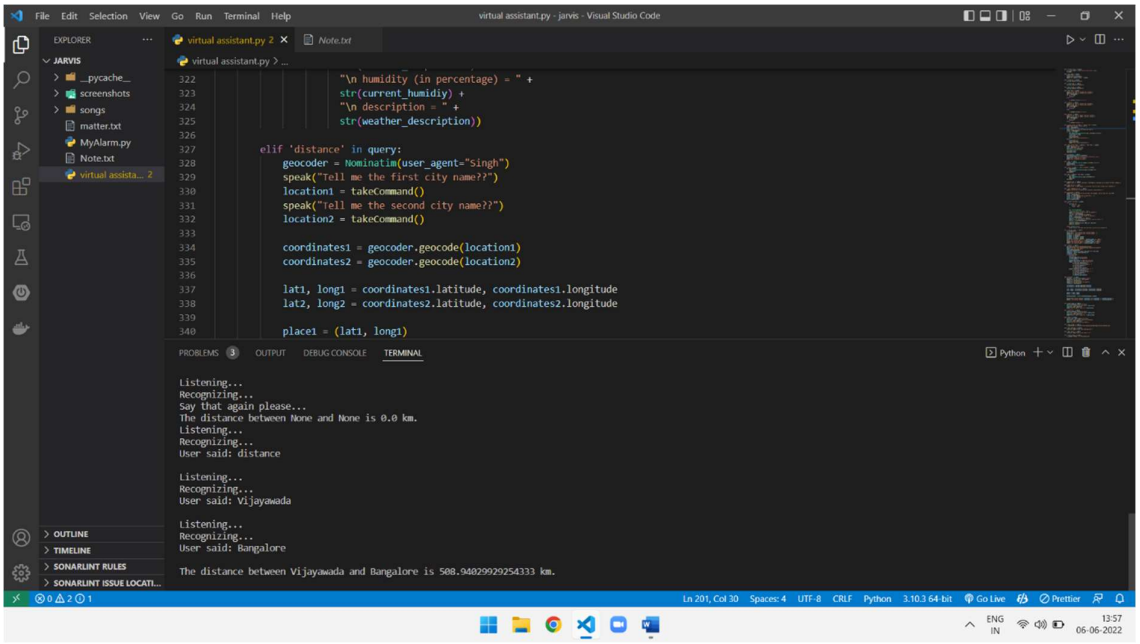
6.3.19 Checking Internet Speed:



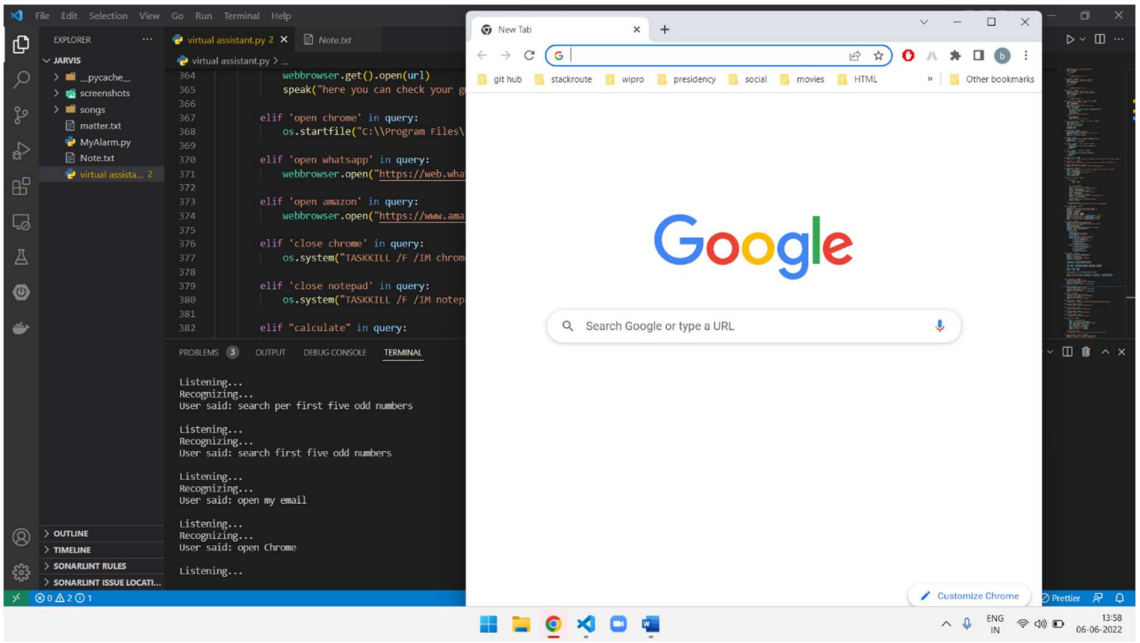
6.3.20 Opening Gmail Inbox:



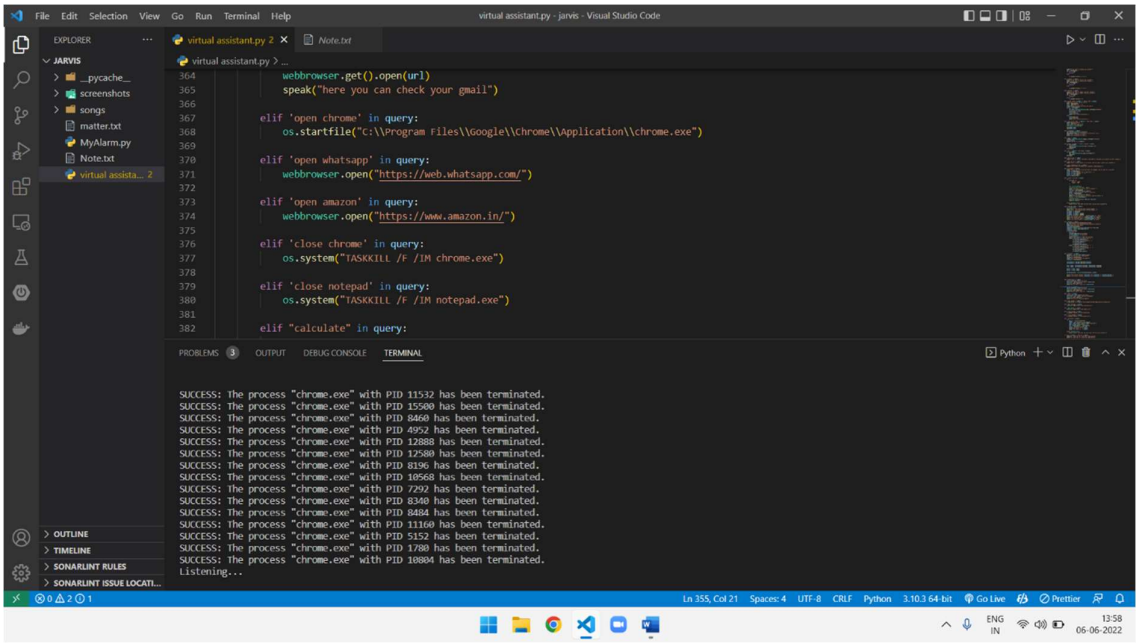
6.3.21 Calculating the Distance:



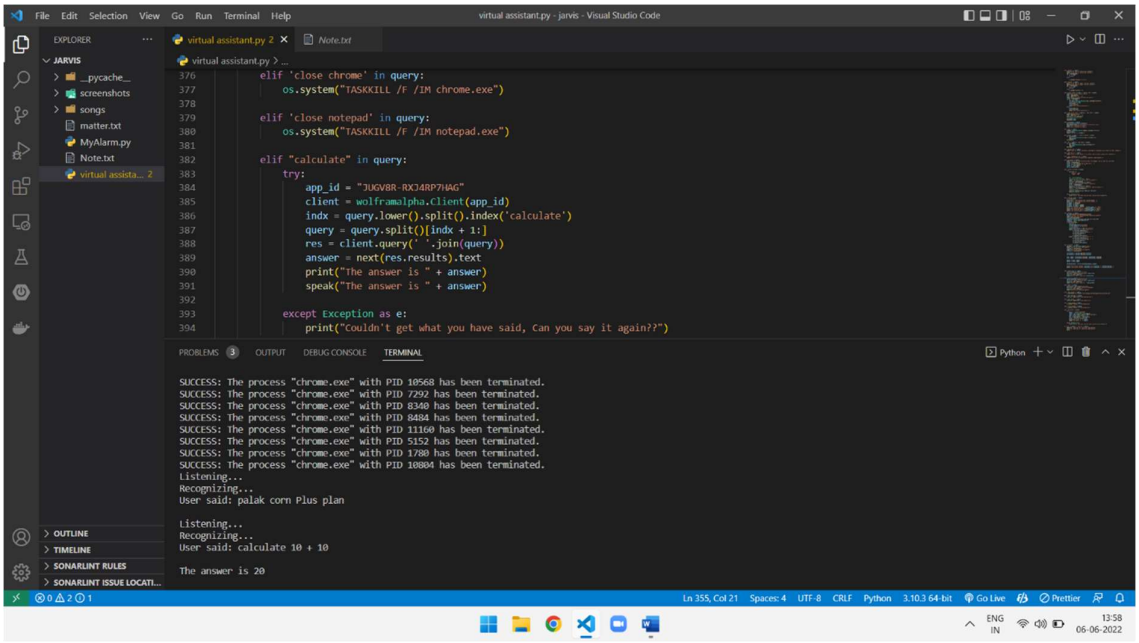
6.3.22 Open Chrome:



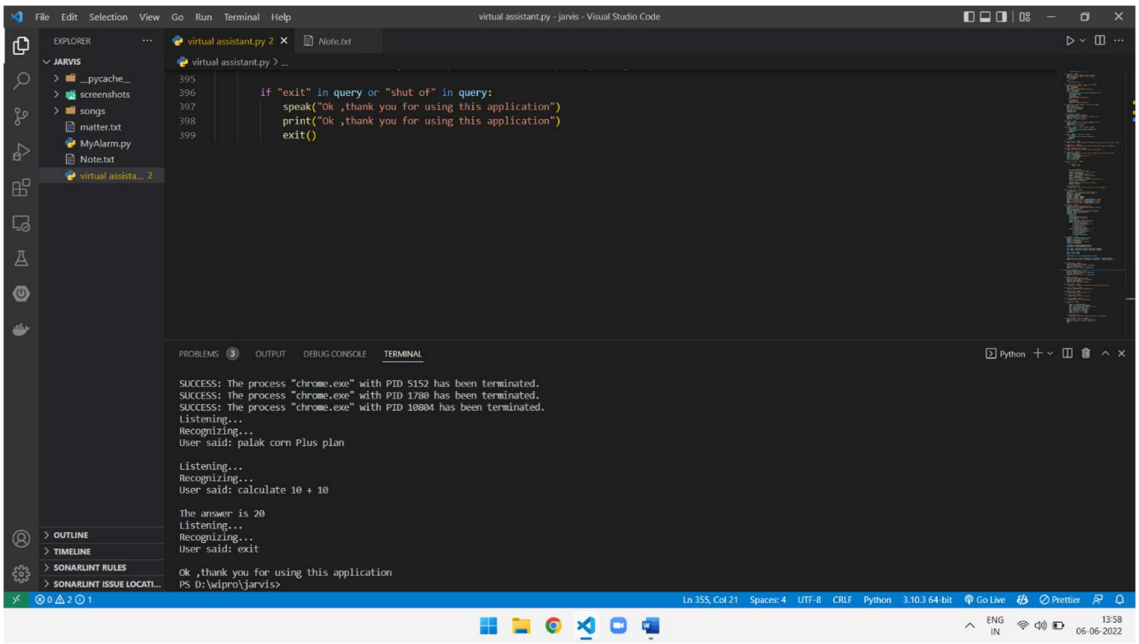
6.3.23 Close Chrome:



6.3.24 Using Calculator:



6.3.25 Exit the Application:



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. St¨ucker, 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*.