

ABSTRACT

As we know Python is an emerging language so it becomes easy to write a script for Voice Assistant in Python. The instructions for the assistant can be handled as per the requirement of the user. Speech recognition is the process of converting speech into text. This is commonly used in voice assistants like Alexa, Siri, etc. In Python there is an API called **SpeechRecognition** which allows us to convert speech into text. It was an interesting task to make my own assistant. It became easier to send emails without typing any word, Searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favorite IDE with the help of a single voice command. In the current scenario, advancement in technologies is such that they can perform any task with the same effectiveness or can speak more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time.

Functionalities of these project include:

1. It can send emails.
2. It can read PDFs.
3. It can send text on WhatsApp.
4. It can open command prompt, your favorite IDE, notepad etc.
5. It can play music.
6. It can do Wikipedia searches for you.
7. It can open websites like Google, YouTube, etc., in a web browser.
8. It can give a weather forecast.
9. It can give desktop reminders of your choice.
10. It can do basic calculations.
11. It can have some basic conversation.

Now the basic question arises in mind that how it is an AI? The virtual assistant that I have created is like if it is not an A.I., but it is the output of a bundle of the statement. But fundamentally, the main purpose of A.I machines is that it can perform human tasks with the same efficiency or even more efficiently than humans. It is a fact that my virtual assistant is not a very good example of A.I., but it is an A.I.

Chapter 1: Introduction

1. INTRODUCTION

Artificial Intelligence, when used with machines, shows us the capability of thinking like humans. In this, a computer system is designed in such a way that typically requires interaction from humans. As we know Python is an emerging language so it becomes easy to write a script for Voice Assistant in Python. The instructions for the assistant can be handled as per the requirement of the user. Speech recognition is Alexa, Siri, etc. In Python there is an API called Speech Recognition which allows us to convert speech into text. It was an interesting task to make my own assistant. It became easier to send emails without typing any word, Searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favorite IDE with the help of a single voice command. In the current scenario, advancement in technologies is such that they can perform any task with the same effectiveness or can speak more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time.

As the voice assistant is using Artificial Intelligence, hence the results that it is providing are highly accurate and efficient. The assistant can help to reduce human effort and time while executing any task. They removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform tasks. The assistant is no less than a human assistant but we can say that this is more effective and efficient to perform any task. The libraries and packages used to make this assistant focuses on the time complexities and reduces time.

The functionalities include, It can send emails, it can read PDF, it can send text on WhatsApp, it can open command prompt, your favorite IDE, notepad etc.. It can play music, it can do Wikipedia searches for you, it can open websites like Google, YouTube, etc., in a web browser, it can give a weather forecast. Also it can give desktop reminders of your choice. It can have some basic conversation too.

Tools and technologies used are PyCharm IDE for making this project, and I created all py files in PyCharm. Along with this I used the following modules and libraries in my project. **pyttsx3**, **SpeechRecognition**, **Datetime**, **Wikipedia**, **Smtplib**, **pywhatkit**, **pyjokes**, **pyPDF2** etc.

1.1 PRESENT SYSTEM

We are familiar with many existing voice assistants like Alexa, Siri, Google Assistant, Cortana which use the concept of language processing, and voice recognition. They listen to the command given by the user as per their requirements and perform that specific function in a very efficient and effective manner.

As these voice assistants are using Artificial Intelligence hence the results that they are providing are highly accurate and efficient. These assistants can help to reduce human effort and time while performing any task. They remove the concept of typing completely and behave as another individual to whom we are talking and asking to perform tasks. These assistants are no less than a human assistant but we can say that they are more effective and efficient to perform any task. The algorithm used to make these assistants focuses on the time complexities and reduces time.

But for using these assistants one should have an account (like Google account for Google assistant, Microsoft account for Cortana) and can use it with internet connection only because these assistants are going to work with internet connectivity. They are integrated with many devices like phones, laptops, and speakers etc.

1.2 PROPOSED SYSTEM

It was an interesting task to make my own assistant. It became easier to send emails without typing any word, Searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favorite IDE with the help of a single voice command. ZARA is different from other traditional voice assistants in terms that it is specific to the desktop and the user does not need to make an account to use this, it does not require any internet connection while getting the instructions to perform any specific task.

The IDE used in this project is PyCharm. All the python files were created in PyCharm and all the necessary packages were easily installable in this IDE. For this project following modules and libraries were used i.e. pyttsx3, SpeechRecognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyPDF2 etc.

With the advancement, ZARA can perform any task with the same effectiveness or can say more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time. Functionalities of this project include, It can **send emails**, It can **read PDF**, It can **send text on WhatsApp**, It can **open command prompt**, your favorite IDE, notepad etc., It can **play music**, It can do **Wikipedia searches** for you, It can **open websites like Google, YouTube**, etc., in a web browser, It can give **weather forecasts**, It can give **desktop reminders** of your choice. It can do basic **calculations**, It can have some basic conversation, etc.

Chapter 2: System Design

2.1 DATA FLOW

The data flow for ZARA as follows:

- **Start:** Press the run button.
- **Input:** It will take input through voice commands related to the task which is required to be done.
- **Perform:** It will perform the required task for the user like opening notepad, searching on browser, sending email, playing song etc.
- **Exit:** It keeps on asking for the command from the user until the user says "goodbye". Once the user says "goodbye", it exits.

Data Flow for ZARA

The system is designed using the concept of Artificial Intelligence and with the help of necessary packages of Python. Python provides many libraries and packages to perform the tasks, for example pyPDF2 can be used to read PDF. The details of these packages are mentioned in Chapter 3 of this report.

The data in this project is nothing but user input. Whatever the user says, the assistant performs the task accordingly. The user input is nothing specific but the list of tasks which a user wants to get performed in human language i.e. English.

Chapter 3: Software Details

The IDE used in this project is PyCharm. All the python files were created in PyCharm and all the necessary packages were easily installable in this IDE. For this project following modules and libraries were used i.e. pyttsx3, SpeechRecognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyPDF2 etc.

3.1 PYCHARM

It is an IDE i.e. Integrated Development Environment which has many features like it supports scientific tools (like matplotlib, numpy, scipy etc) web frameworks (example Django, web2py and Flask) refactoring in Python, integrated python debugger, code completion, code and project navigation etc. It also provides Data Science when used with Anaconda.

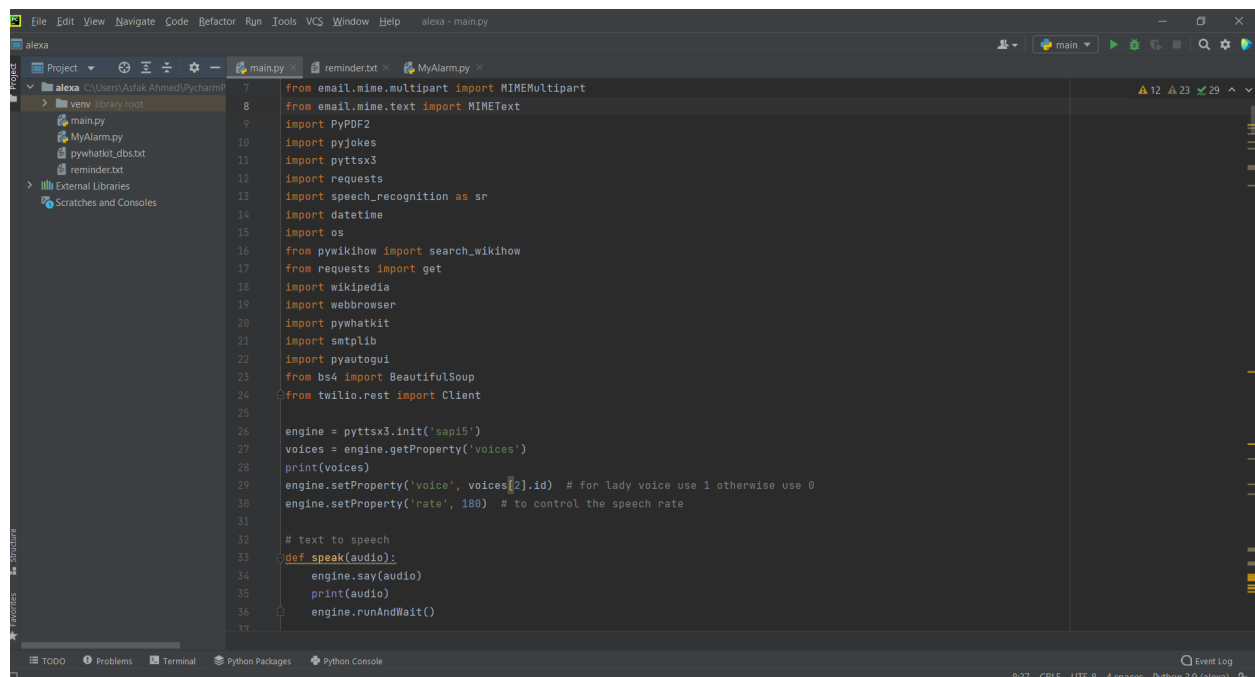


Figure 3.1 Pycharm IDE

3.2 PYTHON LIBRARIES

In ZARA following python libraries were used:

3.3.1. **pyttsx3**: It is a python library which converts text to speech.

3.3.2. **SpeechRecognition**: It is a python module which converts speech to text.

3.3.3. **pywhatkit**: It is a python library to send WhatsApp messages at a particular time with some additional features.

3.3.4. **Datetime**: This library provides us the actual date and time.

3.3.5. **Wikipedia**: It is a python module for searching anything on Wikipedia.

3.3.6. **Smtplib**: Simple mail transfer protocol that allows us to send mail and to route mail between mail servers.

3.3.7. **pyPDF2**: It is a python module which can read, split, merge any PDF.

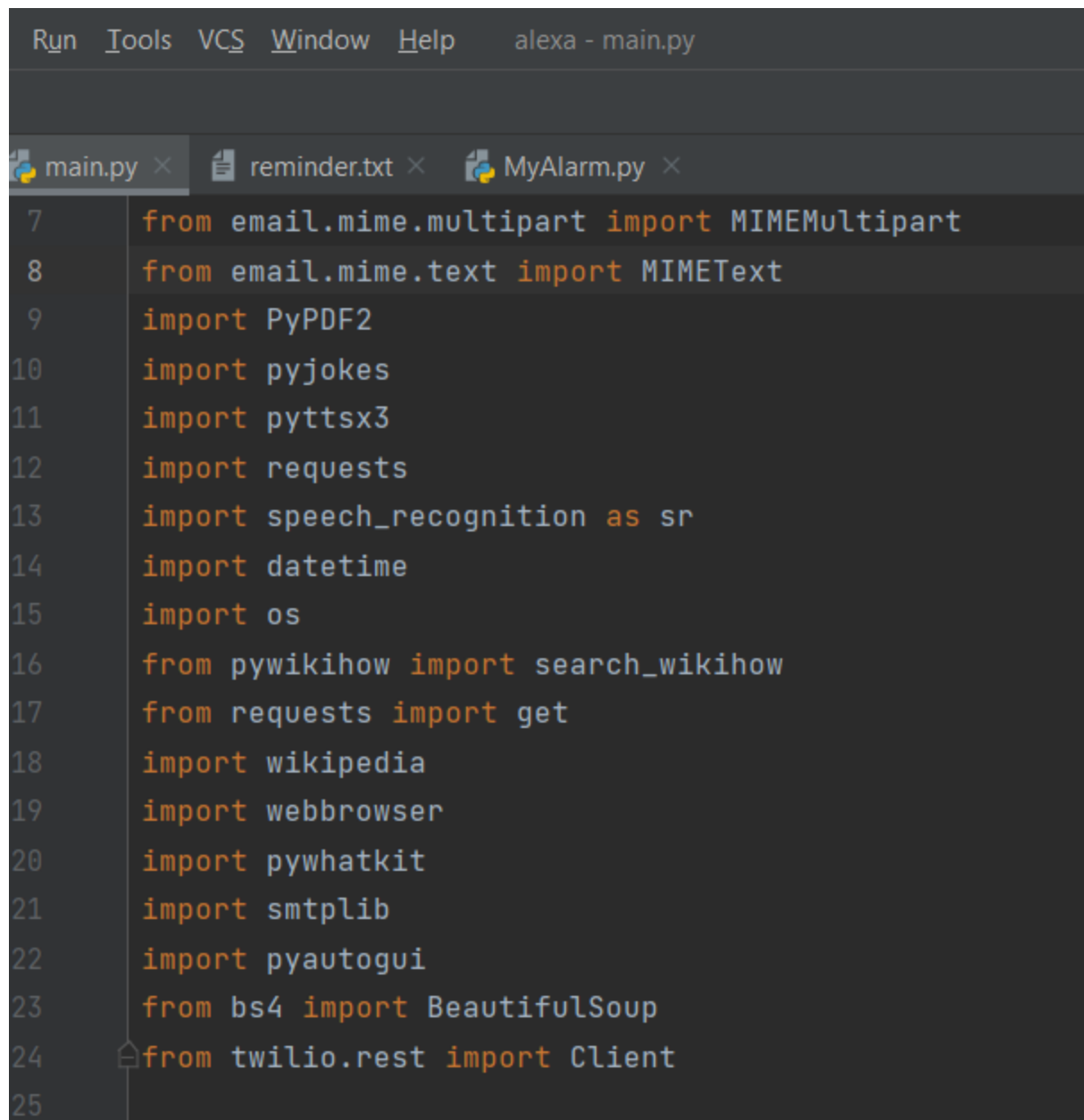
3.3.8. **Pyjokes**: It is a python library which contains lots of interesting jokes in it.

3.3.9. **Webbrowser**: It provides an interface for displaying web-based documents to users.

3.3.10. **os**: It represents Operating System related functionality.

3.3.11. **sys**: It allows operating on the interpreter as it provides access to the variables and functions that usually interact strongly with the interpreter.

3.3.12 **twilio**: It allows us to call and send messages on our mobile phone virtually.



The image shows a screenshot of a Python IDE window titled 'alexa - main.py'. The window has three tabs: 'main.py', 'reminder.txt', and 'MyAlarm.py'. The 'main.py' tab is active, displaying a list of imported modules. The code is as follows:

```
7 from email.mime.multipart import MIMEMultipart
8 from email.mime.text import MIMEText
9 import PyPDF2
10 import pyjokes
11 import pyttsx3
12 import requests
13 import speech_recognition as sr
14 import datetime
15 import os
16 from pywikihow import search_wikihow
17 from requests import get
18 import wikipedia
19 import webbrowser
20 import pywhatkit
21 import smtplib
22 import pyautogui
23 from bs4 import BeautifulSoup
24 from twilio.rest import Client
25
```

Figure 3.2 Imported Modules

Chapter 4: Implementation Work Details

ZARA, a desktop assistant, is a voice assistant that can perform many daily tasks on the desktop like playing music, opening your favorite IDE with the help of a single voice command. Zara is different from other traditional voice assistants in terms that it is specific to the desktop and the user does not need to make an account to use this, it does not require any internet connection while getting the instructions to perform any specific task.

4.1. REAL LIFE APPLICATION

4.1.1. **Saves time:** ZARA is a desktop voice assistant which works on the voice command offered to it, it can do voice searching, voice-activated device control and can let us complete a set of tasks.

4.1.2. **Conversational interaction:** It makes it easier to complete any task as it automatically does it by using the essential module or libraries of Python, in a conversational interaction way. Hence any user when instructing any task to it, they feel like giving task to a human assistant because of the conversational interaction for giving input and getting the desired output in the form of task done.

4.1.3. **Reactive nature:** The desktop assistant is reactive which means it knows human language very well and understands the context that is provided by the user and gives response in the same way, i.e. human understandable language, English. So the user finds its reaction in an informed and smart way.

4.1.4. **Multitasking:** The main application of it can be its multitasking ability. It can ask for continuous instruction one after another until the user "QUIT" it.

4.1.5. **No Trigger phase:** It asks for the instruction and listens to the response that is given by the user without needing any trigger phase and then only executes the task.

4.2. DATA IMPLEMENTATION AND PROGRAM EXECUTION

As the first step, install all the necessary packages and libraries. The command used to install the libraries is “**pip install**” and then import it. The necessary packages included are as follows:

4.2.1. LIBRARIES AND PACKAGES

4.2.1.1. **pyttsx3**: It is a python library which converts text to speech.

4.2.1.2. **SpeechRecognition**: It is a python module which converts speech to text.

4.2.1.3. **pymhatkit**: It is a python library to send WhatsApp messages at a particular time with some additional features.

4.2.1.4. **Datetime**: This library provides us the actual date and time.

4.2.1.5. **Wikipedia**: It is a python module for searching anything on Wikipedia.

4.2.1.6. **Smtplib**: Simple mail transfer protocol that allows us to send mail and to route mail between mail servers.

4.2.1.7. **pyPDF2**: It is a python module which can read, split, merge any PDF.

4.2.1.8. **Pyjokes**: It is a python library which contains lots of interesting jokes in it.

4.2.1.9. **Webbrowser**: It provides an interface for displaying web-based documents to users.

4.2.1.10. **Pyautogui**: It is a python library for graphical user interface.

4.2.1.11. **os**: It represents Operating System related functionality.

4.2.1.12. **sys**: It allows operating on the interpreter as it provides access to the variables and functions that usually interact strongly with the interpreter.

4.2.2. FUNCTIONS

4.2.2.1. **takeCommand()**: The function is used to take the command as input through the microphone of the user and returns the output as string.

4.2.2.2. **wishMe()**: This function greets the user according to the time like Good Morning, Good Afternoon and Good Evening.

4.2.2.3. **taskExecution()**: This is the function which contains all the necessary task execution definition like `sendEmail()`, `pdf_reader()`, `news()` and many conditions in if condition like “open google”, “open notepad”, “search on Wikipedia” ,”play music” and “open command prompt” etc.

Chapter 5: Source Code and Command

Main.py

```
import operator

import random

import sys

import time

from email import encoders

from email.mime.base import MIMEBase

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

import PyPDF2

import pyjokes

import pyttsx3

import requests

import speech_recognition as sr

import datetime

import os

from pywikihow import search_wikihow

from requests import get

import wikipedia

import webbrowser

import pywhatkit

import smtplib
```

```

import pyautogui

from bs4 import BeautifulSoup

from twilio.rest import Client

engine = pyttsx3.init('sapi5')

voices = engine.getProperty('voices')

print(voices)

engine.setProperty('voice', voices[2].id) # for lady voice use 1 otherwise use 0

engine.setProperty('rate', 180) # to control the speech rate

# text to speech

def speak(audio):

    engine.say(audio)

    print(audio)

    engine.runAndWait()

# to convert voice into text

def takecommand():

    r = sr.Recognizer()

    with sr.Microphone() as source:

        print('listening...')

        r.pause_threshold = 1

        audio = r.listen(source, timeout=100, phrase_time_limit=100)

    try:

        print("Recognizing...")

        query = r.recognize_google(audio, language='en-in')

        print(f"user said: {query}")

```

```

except Exception as e:

    # speak("say that again please sir...")

    return "none"

query = query.lower()

return query

# to read pdf

def pdf_reader():

    speak("sir please enter the location of the file")

    location = input("Enter the location here: ")

    book = open(location, 'rb')

    pdfReader = PyPDF2.PdfFileReader(book)

    pages = pdfReader.numPages

    speak(f"Total number of pages of this book {pages}")

    speak("sir please enter the page number i have to read")

    pg = int(input("Please enter the page number: "))

    page = pdfReader.getPage(pg)

    text = page.extractText()

    speak(text)

# to wish

def wish():

    user_name = 'name of the user'    # enter the name of the user

    hour = int(datetime.datetime.now().hour)

    tt = time.strftime("%I:%M %p")

```

```

if hour >= 6 and hour <= 12:

    speak(f"good morning {user_name}, its {tt}")

elif hour >= 12 and hour <= 15:

    speak(f"good noon {user_name}, its {tt}")

elif hour > 15 and hour <= 18:

    speak(f"good afternoon {user_name}, its {tt}")

else:

    speak(f"good evening {user_name}, its {tt}")

    speak("i am zara, your personal assistant. Tell me how may i help you?")

# for news update

def news():

    main_url = 'link of api' # paste here your news api link

    main_page = requests.get(main_url).json()

    articles = main_page["articles"]

    head = []

    day = ["first", "second", "third", "fourth", "fifth"]

    for ar in articles:

        head.append(ar["title"])

    for i in range(len(day)):

        speak(f"today's {day[i]} news is: {head[i]}")

def TaskExecution():

    wish()

    while True:

        query = takecommand()

```

```

# logic building for tasks

# to open notepad

if "open notepad" in query:

    speak('opening notepad...')

    npath = "C:\\WINDOWS\\system32\\notepad.exe"

    os.startfile(npath)

# to close notepad

elif "close notepad" in query:

    speak("closing notepad...")

    os.system("taskkill /f /im notepad.exe")

# to open word

elif "open word" in query:

    speak("opening ms word...")

    npath = "C:\\Program Files\\Microsoft Office\\root\\Office16\\WINWORD.EXE"

    os.startfile(npath)

# to close word

elif "close word" in query:

    speak("closing ms word...")

    os.system("taskkill /f /im WINWORD.EXE")

# to open excel

elif "open excel" in query:

    speak("opening ms excel...")

    npath = "C:\\Program Files\\Microsoft Office\\root\\Office16\\EXCEL.EXE"

    os.startfile(npath)

```



```

# to close excel

elif "close excel" in query:

    speak("closing ms excel...")

    os.system("taskkill /f /im EXCEL.EXE")

# to open power point

elif "open powerpoint" in query or "open power point" in query:

    speak("opening powerpoint...")

    npath = "C:\\Program Files\\Microsoft
Office\\root\\Office16\\POWERPNT.EXE"

    os.startfile(npath)

# to close power point

elif "close powerpoint" in query or "close power point" in query:

    speak("closing powerpoint...")

    os.system("taskkill /f /im POWERPNT.EXE")

# to open command prompt

elif "open command prompt" in query:

    speak('opening command prompt...')

    os.system("start cmd")

# to close command prompt

elif "close command prompt" in query:

    speak("closing...")

    os.system("taskkill /f /im cmd.exe")

# to play music from a specific directory

elif "play music" in query:

```

```

    music_dir = "path_of_the_music_file"    # enter the location of music file

    songs = os.listdir(music_dir)

    rd = random.choice(songs)

    os.startfile(os.path.join(music_dir, rd))

# to know your ip address

elif "ip address" in query:

    ip = get('https://api.ipify.org').text

    speak(f"your ip address is {ip}")

# to search in wikipedia

elif "wikipedia" in query:

    speak("searching wikipedia...")

    query = query.replace("wikipedia", "")

    results = wikipedia.summary(query, sentences=2)

    speak("according to wikipedia")

    speak(results)

# to open youtube

elif "open youtube" in query:

    speak('opening youtube...')

    webbrowser.open("www.youtube.com")

# to close youtube

elif "close youtube" in query:

    speak("closing...")

    os.system("taskkill /f /im chrome.exe")

# to open facebook

```

```
elif "open facebook" in query:

    speak('opening facebook...')

    webbrowser.open("www.facebook.com")

# to close facebook

elif "close facebook" in query:

    speak("closing...")

    os.system("taskkill /f /im chrome.exe")

# to open instagram

elif "open instagram" in query:

    speak('opening instagram...')

    webbrowser.open("www.instagram.com")

# to close instagram

elif "close instagram" in query:

    speak("closing...")

    os.system("taskkill /f /im chrome.exe")

# to open twitter

elif "open twitter" in query:

    speak('opening twitter...')

    webbrowser.open("www.twitter.com")

# to close twitter

elif "close twitter" in query:

    speak("closing...")

    os.system("taskkill /f /im chrome.exe")

# to open google and search
```

```

elif "open google" in query:

    speak("sir, what should i search on google")

    cm = takecommand().lower()

    webbrowser.open(f"{cm}")

# to send message in whatsapp

# it is necessary to login to your whatsapp web account

elif "send message" in query:

    speak('please enter the number: ')

    num = input("")

    speak('what should i write in the message?')

    msg = takecommand().lower()

    pywhatkit.sendwhatmsg_instantly(num, msg, wait_time=5)

    speak(f"message send to {num} successfully")

# to play song on youtube

elif "play a song on youtube" in query:

    speak('which song you want to listen?')

    n = takecommand().lower()

    speak("playing...")

    pywhatkit.playonyt(n)

# to close song of youtube

elif "close song" in query:

    speak("closing...")

    os.system("taskkill /f /im chrome.exe")

# to set an alarm

```

```

elif "set alarm" in query:

    speak("sir please tell me when i set the alarm, for example, set alarm
to 12.30PM")

    tt = takecommand()

    tt = tt.replace("set alarm to ", "")

    tt = tt.replace(".", "")

    tt = tt.upper()

    import MyAlarm

    MyAlarm.alarm(tt)

# for joke

elif "tell me a joke" in query:

    joke = pyjokes.get_joke()

    speak(joke)

# put assistant in sleep

elif "you can sleep now" in query or "sleep now" in query:

    speak("okay sir, i am going to sleep. You can call me anytime")

    break

# to shutdown system

elif "shutdown my system" in query:

    speak("shutting down...")

    os.system("shutdown /s /t 5")

    sys.exit()

# to restart system

elif "restart my system" in query:

```

```

        speak("restarting system...")

        os.system("shutdown /r /t 5")

        sys.exit()

# for volume up

elif "volume up" in query:

    pyautogui.press("volumeup")

# for volume down

elif "volume down" in query:

    pyautogui.press("volumedown")

# for mute

elif "mute" in query or "volume mute" in query:

    pyautogui.press("volumemute")

# to switch the window

elif "switch the window" in query:

    pyautogui.keyDown("alt")

    pyautogui.press("tab")

    time.sleep(1)

    pyautogui.keyUp("alt")

# to know the news

elif "tell me news" in query:

    speak("please wait sir, fetching the latest news... ")

    news()

# send email

elif "send email" in query:

```

```

speak("sir, what should i say?")

query = takecommand().lower()

try:

    if "send file" in query:

        email = 'mail@gmail.com' # your email

        password = 'password' # your password

        speak("Please enter the email id: ")

        send_mail = input("Enter here: ")

        send_mail_to = send_mail + "@gmail.com" # whom you want to send

        print(send_mail_to)

        speak("okay sir, what is the subject for this email?")

        query = takecommand().lower()

        subject = query # subject in the email

        speak("and sir, what is the message for this email?")

        query2 = takecommand().lower()

        message = query2 # message in the email

        speak("sir please enter the correct path of the file into the
shell:")

        file_location = input("Please enter the path here: ") # the file
location for email

        speak("please wait, i am sending email now...")

        msg = MIMEMultipart()

        msg['From'] = email

        msg['To'] = send_mail_to

        msg['Subject'] = subject

```

```

msg.attach(MIMEText(message, 'plain'))

# setup the attachment

filename = os.path.basename(file_location)

attachment = open(file_location, "rb")

part = MIMEBase('application', 'octet-stream')

part.set_payload(attachment.read())

encoders.encode_base64(part)

part.add_header('Content-Disposition', "attachment; filename= %s"
% filename)

# attach the attachment to the MIMEMultipart object

msg.attach(part)

server = smtplib.SMTP('smtp.gmail.com', 587)

server.starttls()

server.login(email, password)

text = msg.as_string()

server.sendmail(email, send_mail_to, text)

server.quit()

speak(f"email has been sent to {send_mail_to}")

else:

    email = 'mail@gmail.com' # your email id

    password = 'password' # your password

    speak("please enter the email id: ")

    send_mail = input("Enter here: ")

    send_mail_to = send_mail + "@gmail.com" # whom you want to send

```



```

        print(send_mail_to)

        speak("okay sir, what is the subject for this email?")

        query3 = takecommand().lower()

        subject1 = query3 # the message in the email

        speak("please wait, i am sending email now...")

        msg1 = MIMEMultipart()

        msg1['From'] = email

        msg1['To'] = send_mail_to

        msg1['Subject'] = subject1

        server = smtplib.SMTP('smtp.gmail.com', 587)

        server.starttls()

        server.login(email, password)

        server.sendmail(email, send_mail_to, query)

        server.quit()

        speak(f"email has been sent to {send_mail_to}")

    except Exception as e:

        print(e)

        speak(f"sorry sir, i am unable to send the email {send_mail_to}")

# to know our location

elif "where i am" in query or "where we are" in query:

    speak("wait sir, let me check")

    try:

        ipAdd = requests.get('https://api.ipify.org').text

        print(ipAdd)

```

```

url = 'https://get.geojs.io/v1/ip/geo/' + ipAdd + '.json'

geo_requests = requests.get(url)

geo_data = geo_requests.json()

# print(geo_data)

city = geo_data['city']

region = geo_data['region']

country = geo_data['country']

speak(

    f"sir i am not sure, but i think we are in {city} city somewhere
in {region} of {country} country")

except Exception as e:

    speak("sorry sir, Due to network issue i am unable to find our
location")

    pass

# to know weather forecast

elif "temperature" in query:

    search = "temperature in kolkata"

    url = f"https://www.google.com/search?q={search}"

    r = requests.get(url)

    data = BeautifulSoup(r.text, "html.parser")

    temp = data.find("div", class_="BNeawe").text

    speak(f"current {search} is {temp}")

# to search anything on internet

elif "activate how to do mod" in query:

    speak("How to do mode is activated.")

```

```

while True:

    speak("please tell me what do you want to know")

    how = takecommand()

    try:

        if "exit" in how or "close" in how:

            speak("okay sir, how to do mode is deactivated.")

            break

        else:

            max_results = 1

            how_to = search_wikihow(how, max_results)

            assert len(how_to) == 1

            how_to[0].print()

            speak(how_to[0].summary)

        except Exception as e:

            speak("sorry sir, i am unable to find this.")

    # to check battery

    elif "how much power left" in query or "check battery" in query or "how much
power we have" in query:

        import psutil

        battery = psutil.sensors_battery()

        percentage = battery.percent

        speak(f"sir our system have {percentage} percent battery")

        if percentage >= 75:

            speak("we have enough power to continue our work.")

```

```

elif percentage >= 40 and percentage < 75:

    speak("we should connect our system to power supply to give power to
our system")

elif percentage <= 15 and percentage < 40:

    speak("we don't have much power to work, please connect to power
supply")

elif percentage < 15:

    speak("we have very low power, please connect to power supply,
system will shutdown soon")

# to take screenshot

elif "take screenshot" in query or "take a screenshot" in query:

    speak("sir, please tell me the name for this screenshot file")

    file = takecommand().lower()

    speak("please sir hold screen for few seconds, i am taking screenshot")

    time.sleep(3)

    img = pyautogui.screenshot()

    img.save(f"{file}.png")

    speak("i am done sir, the screenshot is saved in my memory. now i am
ready for the next task.")

# to calculate

elif "can you calculate" in query or "do some calculations" in query or "do
some calculation" in query:

    r = sr.Recognizer()

    with sr.Microphone() as source:

        speak("say what do you want to calculate, example 5 plus 6")

        print("listening...")

```

```

        r.adjust_for_ambient_noise(source)

        audio = r.listen(source)

        my_string = r.recognize_google(audio)

        print(my_string)

    def get_operator_fn(op):

        return {

            '+': operator.add,          # plus

            '-': operator.sub,          # minus

            'x': operator.mul,          # x

            'divided': operator.__truediv__, # divided

        }[op]

    def eval_binary_expr(op1, oper, op2):

        op1, op2 = int(op1), int(op2)

        return get_operator_fn(oper)(op1, op2)

    speak("your result is")

    speak(eval_binary_expr(*(my_string.split()))))

# to check internet speed

elif "internet speed" in query:

    import speedtest

    st = speedtest.Speedtest()

    dl = st.download()

    up = st.upload()

    speak(f"sir we have {dl} bit per second downloading speed and {up} bit
per second upload speed")

```

```

# to send normal message

elif "send text" in query:

    speak("sir what should i say")

    msz = takecommand()

    account_sid = 'account sid'    # paste your account_sid here

    auth_token = 'auth token'      # paste your token here

    client = Client(account_sid, auth_token)

    message = client.messages \

        .create(

            body=msz,

            from_='the number you got', # the number you got from twilio

            to='sign up number'        # the number with which you sign up

        )

    print(message.sid)

    speak("message has been sent")

# to call

elif "make a call" in query:

    account_sid = 'account sid'    # paste your account_sid here

    auth_token = 'auth token'      # paste your token here

    client = Client(account_sid, auth_token)

    message = client.calls \

        .create(

            twiml='<Response><Say>This is the testing call from
zara.<<Say></Response>',

```

```

        from_='the number you got', # the number you got from twilio

        to='sign up number'          # the number with which you sign up
    )

    print(message.sid)

# to open mobile camera

elif "open mobile camera" in query:

    import urllib.request

    import cv2

    import numpy as np

    import time

    URL = "ip address shown in ip webcam" # paste your own ip address

    while True:

        img_arr = np.array(bytearray(urllib.request.urlopen(URL).read()),
dtype=np.uint8)

        img = cv2.imdecode(img_arr, -1)

        cv2.imshow('IPWebcam', img)

        q = cv2.waitKey(0)

        if q == ord("q"):

            break;

        cv2.destroyAllWindows()

# to open webcam

elif "open camera" in query or "open webcam" in query:

    speak("opening...")

    import cv2

```

```

cap = cv2.VideoCapture(0)

while True:

    ret, frame = cap.read()

    cv2.imshow('frame', frame)

    if cv2.waitKey(10) == ord('q'):

        break

cap.release()

cv2.destroyAllWindows()

# to set reminder

elif "remember that" in query:

    rememberMsg = query.replace("zara", "")

    rememberMsg = rememberMsg.replace("remember that", "")

    speak(f"you tell me to remember that:" + rememberMsg)

    remember = open('reminder.txt', 'w')

    remember.write(rememberMsg)

    remember.close()

# to listen what zara remember

elif "what do you remember" in query:

    remember = open('reminder.txt', 'r')

    speak("you tell me to remember that" + remember.read())

# read pdf

elif "read pdf" in query:

    pdf_reader()

# interaction with zara

```



```

elif "hello" in query or "hi" in query or "what's up" in query:

    speak("hello sir, how may i help you")

elif "how are you" in query:

    speak("I am fine. what about you sir?")

elif "am good" in query or "i'm good" in query:

    speak("that's great, sir.")

elif "thank you" in query or "thanks" in query:

    speak("it's my duty sir.")

if __name__ == "__main__":

    # startExecution()

    while True:

        permission = takecommand()

        if "wake up" in permission or "good morning" in permission:

            TaskExecution()

        elif "goodbye" in permission:

            speak("thanks for using me sir, have a good day")

            sys.exit()

```

Alarm.py

```
import datetime

import winsound

def alarm(Timing):

    altime = str(datetime.datetime.now().strftime(Timing, "%I:%M %p"))

    print(altime)

    altime = altime[11:-3]

    print(altime)

    Horeal = altime[:2]

    Horeal = int(Horeal)

    Mireal = altime[3:5]

    Mireal = int(Mireal)

    print(f"Done, alarm is set for {Timing}")

    while True:

        if Horeal == datetime.datetime.now().hour:

            if Mireal == datetime.datetime.now().minute:

                print("alarm is running")

                winsound.PlaySound('abc', winsound.SND_LOOP)

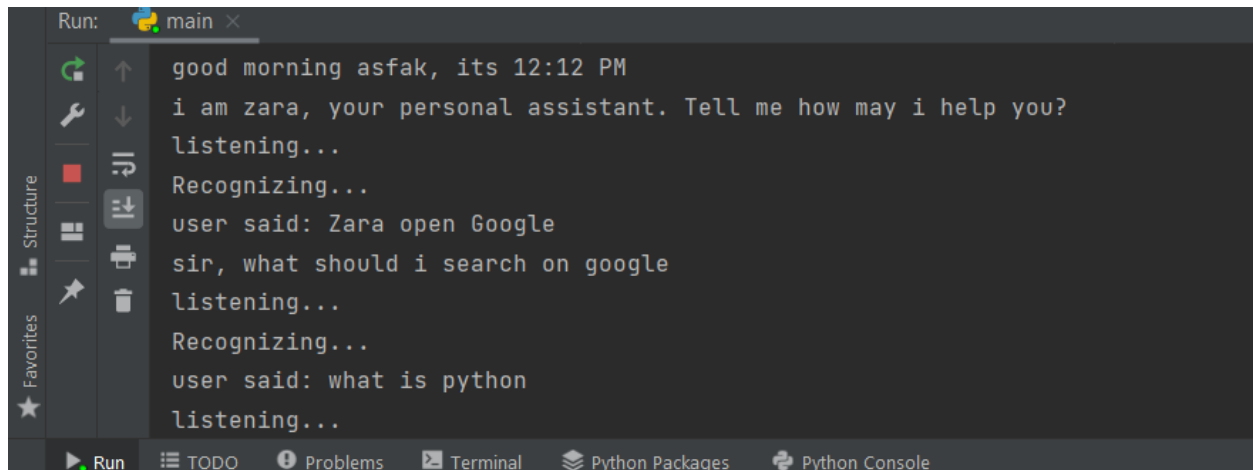
            elif Mireal < datetime.datetime.now().minute:

                break

if __name__ == '__main__':

    alarm('2:40 am')
```

Chapter 6: Input/Output Screenshot



```
Run: main x
good morning asfak, its 12:12 PM
i am zara, your personal assistant. Tell me how may i help you?
listening...
Recognizing...
user said: Zara open Google
sir, what should i search on google
listening...
Recognizing...
user said: what is python
listening...
```

Fig- 6.1 Input for google search

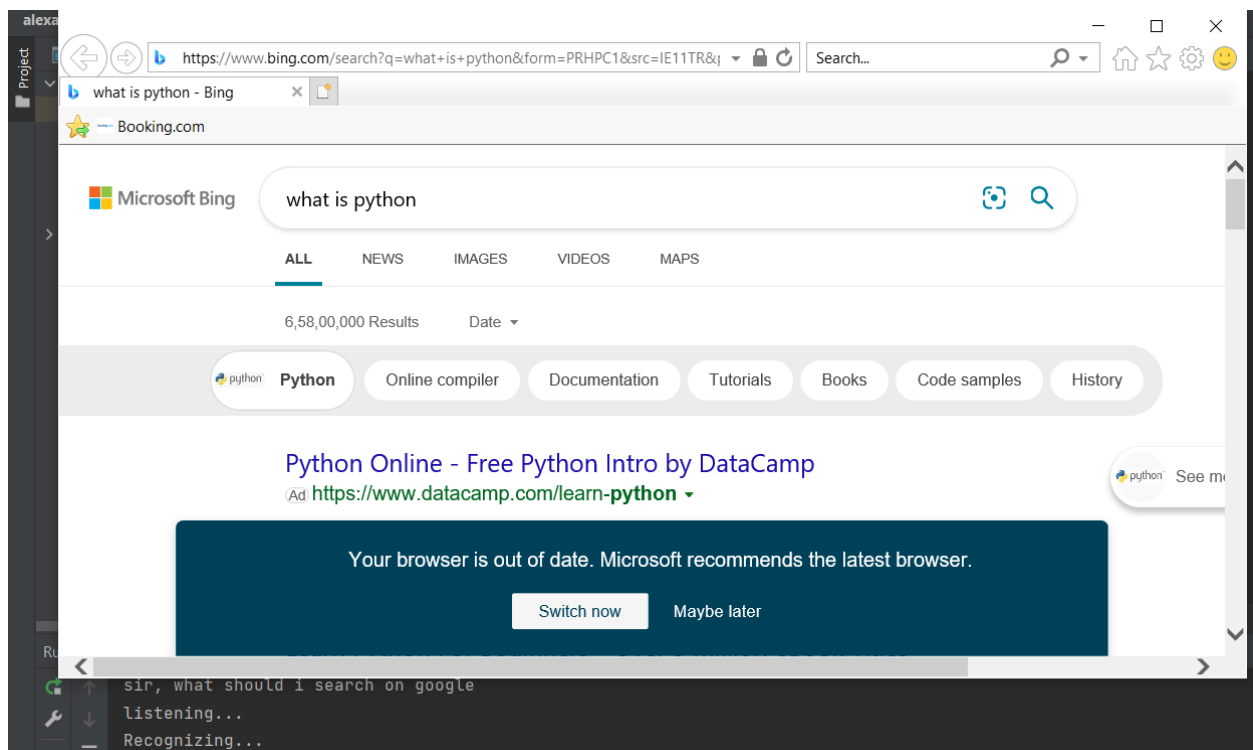
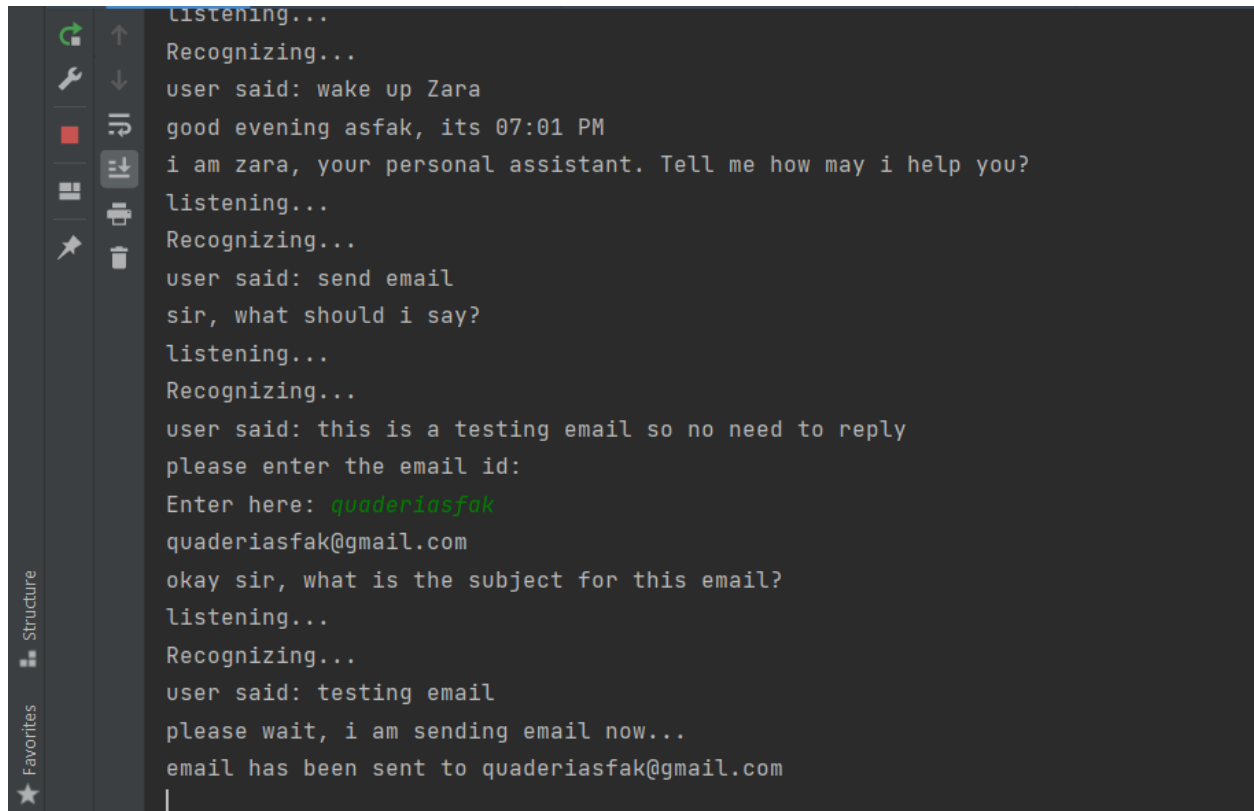


Fig- 6.2 Output for google search



```
listening...
Recognizing...
user said: wake up Zara
good evening asfak, its 07:01 PM
i am zara, your personal assistant. Tell me how may i help you?
listening...
Recognizing...
user said: send email
sir, what should i say?
listening...
Recognizing...
user said: this is a testing email so no need to reply
please enter the email id:
Enter here: quaderiasfak
quaderiasfak@gmail.com
okay sir, what is the subject for this email?
listening...
Recognizing...
user said: testing email
please wait, i am sending email now...
email has been sent to quaderiasfak@gmail.com
```

Fig- 6.3 Input for send email

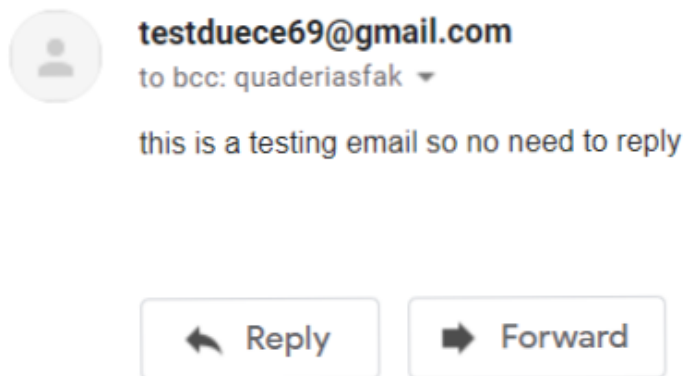


Fig- 6.4 Output for send email

```
Structure
user said: wake up Sara
good evening asfak, its 07:06 PM
i am zara, your personal assistant. Tell me how may i help you?
listening...
Recognizing...
user said: open YouTube
opening youtube...
listening...
```

Fig- 6.5 Input for open youtube

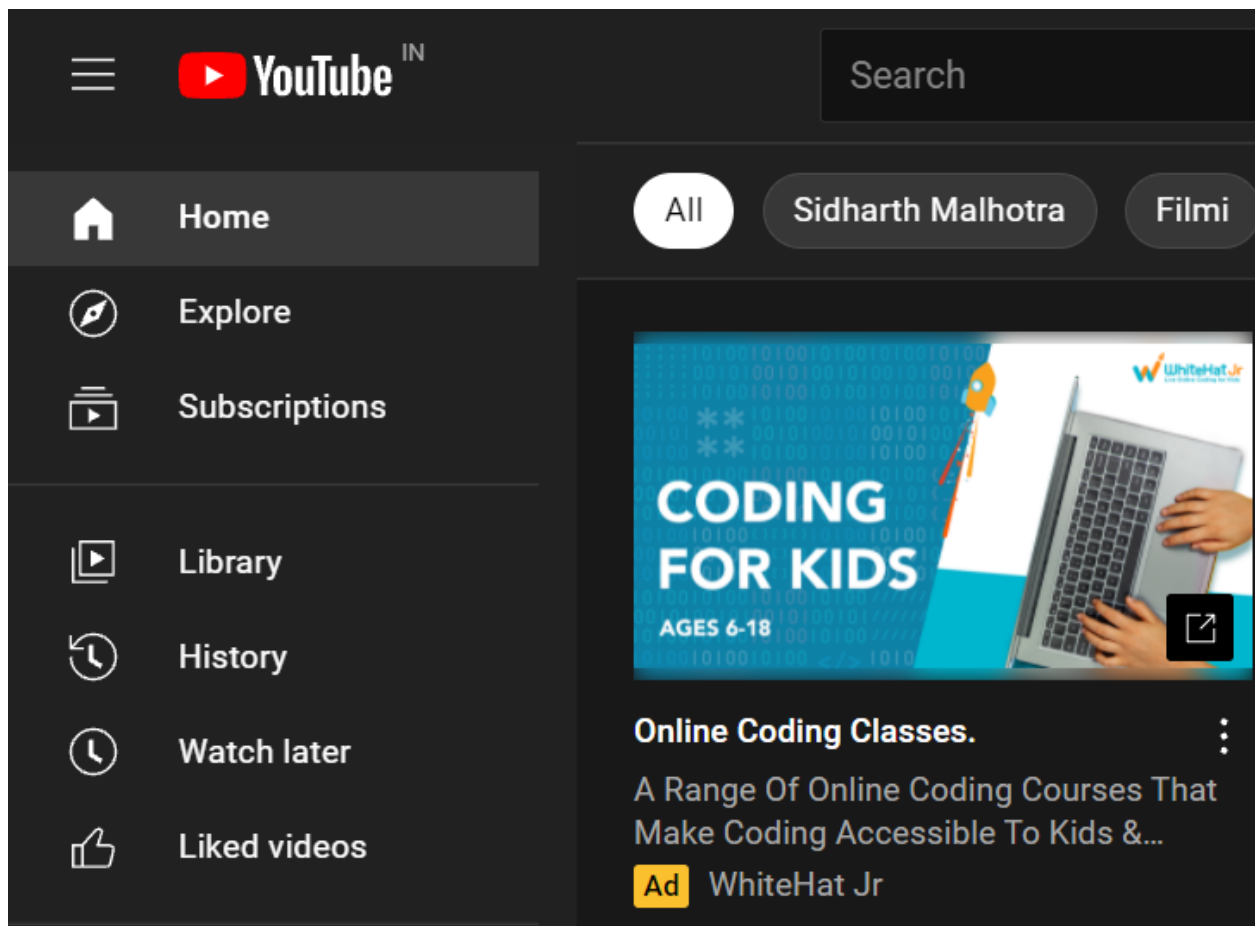
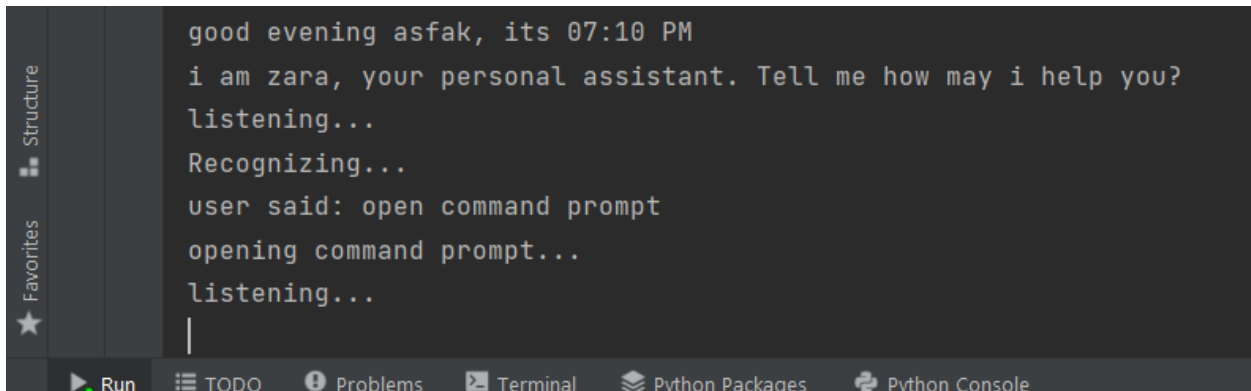


Fig- 6.6 Output for open youtube

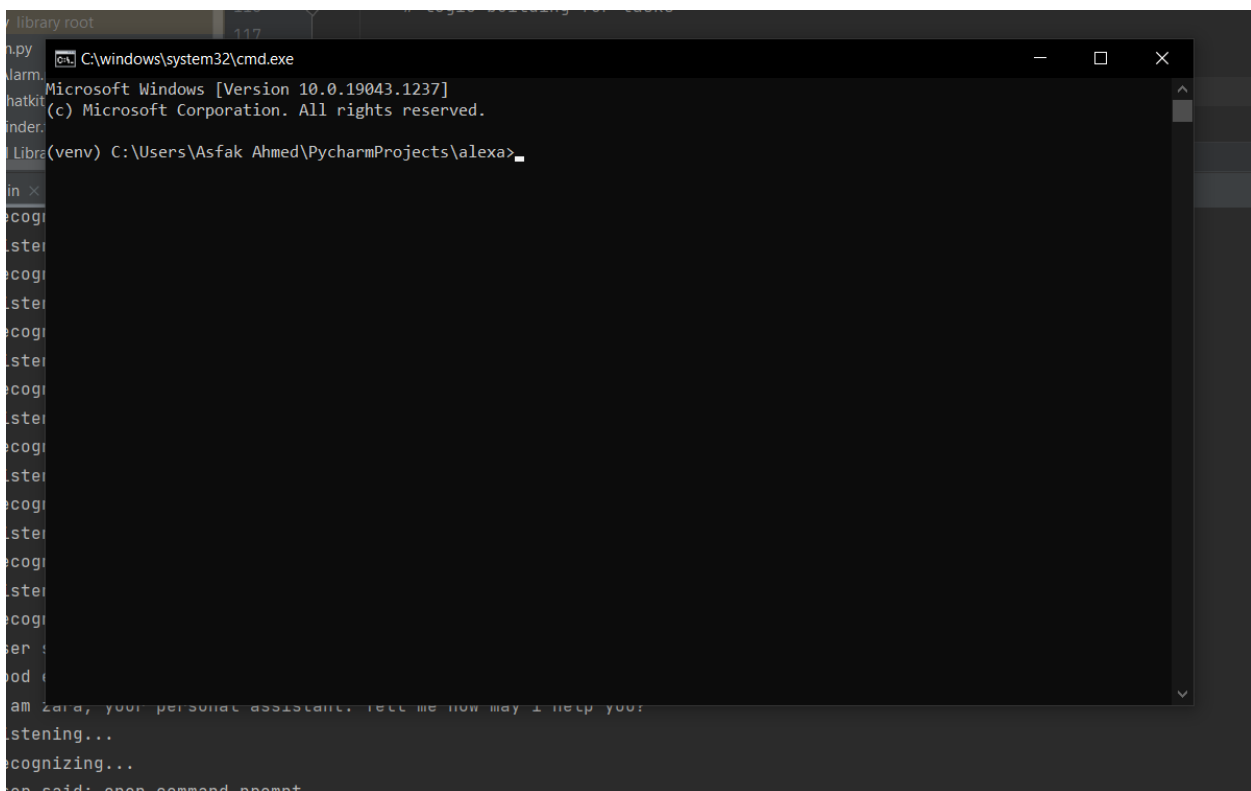


A screenshot of a terminal window within an IDE. The terminal shows a voice assistant's input. On the left side of the IDE, there are panels for 'Structure' and 'Favorites'. The terminal text is as follows:

```
good evening asfak, its 07:10 PM
i am zara, your personal assistant. Tell me how may i help you?
listening...
Recognizing...
user said: open command prompt
opening command prompt...
listening...
|
```

At the bottom of the IDE, there is a toolbar with icons for 'Run', 'TODO', 'Problems', 'Terminal', 'Python Packages', and 'Python Console'.

Fig- 6.7 Input for open cmd

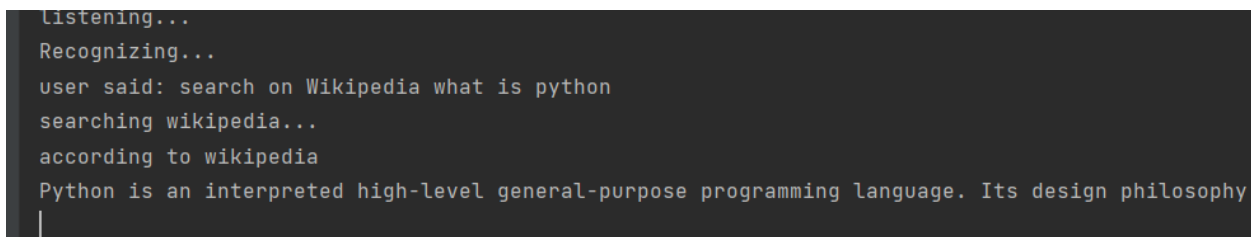


A screenshot of a terminal window titled 'C:\windows\system32\cmd.exe'. The window shows the output of opening a command prompt. The text is as follows:

```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Asfak Ahmed\PycharmProjects\alexa>
```

The terminal window is overlaid on a background that shows the same voice assistant interface as Figure 6.7, with the text 'am zara, your personal assistant. Tell me how may i help you?', 'stening...', 'ecognizing...', and 'er said: open command prompt' visible.

Fig- 6.8 Output for open cmd



A screenshot of a terminal window showing the input and output for a Wikipedia search. The text is as follows:

```
Listening...
Recognizing...
user said: search on Wikipedia what is python
searching wikipedia...
according to wikipedia
Python is an interpreted high-level general-purpose programming language. Its design philosophy
|
```

Fig- 6.9 Input and Output for wikipedia search

Chapter 7: System Testing

The system testing is done on a fully integrated system to check whether the requirements are matching or not. The system testing for ZARA desktop assistant focuses on the following four parameters:

7.1 FUNCTIONALITY

In this we check the functionality of the system whether the system performs the task which it was intended to do. To check the functionality each function was checked and run, if it is able to execute the required task correctly then the system passes in that particular functionality test. For example, to check whether ZARA can search on Google or not, as we can see in the figure 7.1, the user said "Open Google", then Zara asked, "What should I search on Google?" Then the user said, "What is Python?", Zara opened Google and searched for the required input.

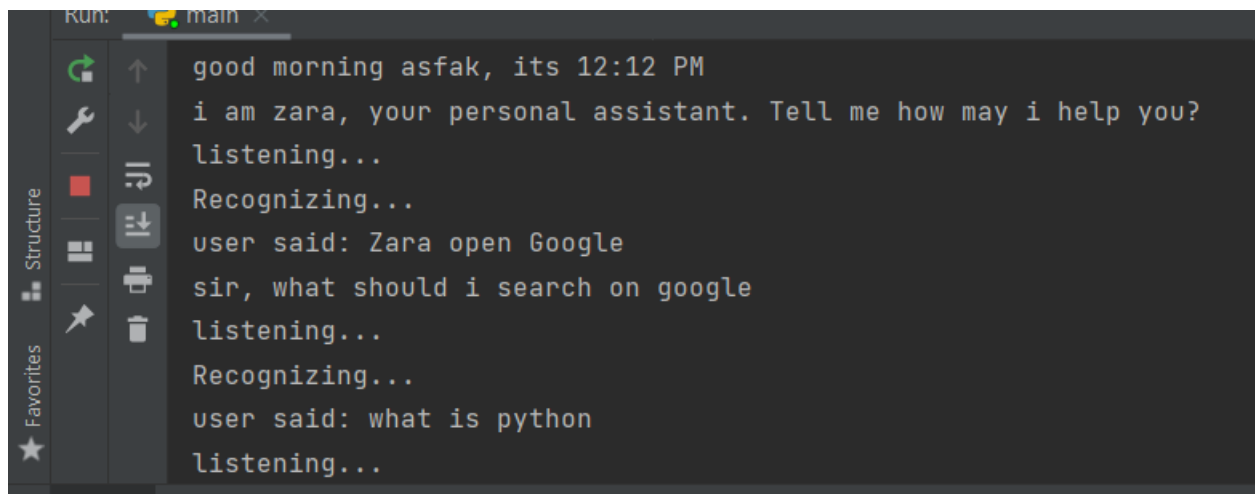


Figure 7.1 Input through voice commands

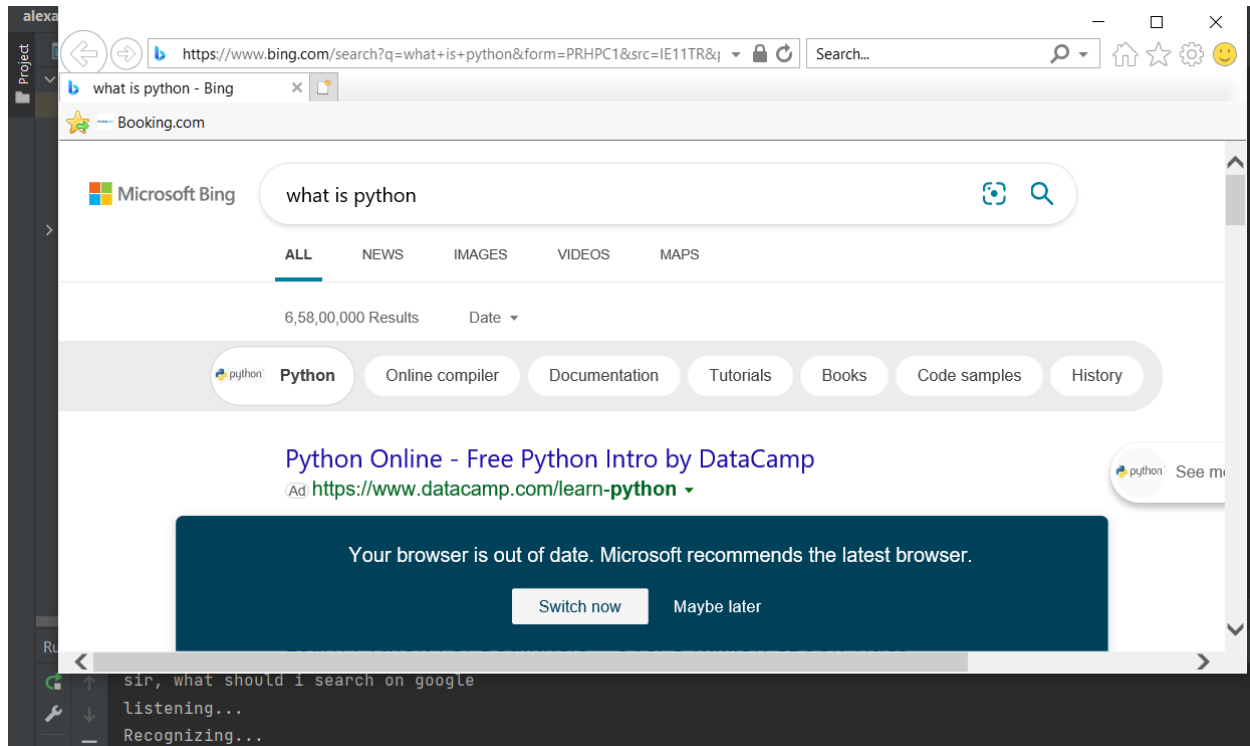


Figure 7.2 Output

7.2. USABILITY

Usability of a system is checked by measuring the ease of the software and how user friendly it is for the user to use, how it responds to each query that is being asked by the user.

It makes it easier to complete any task as it automatically does it by using the essential module or libraries of Python, in a conversational interaction way. Hence any user when instructing any task to it, they feel like giving task to a human assistant because of the **conversational interaction** for giving input and getting the desired output in the form of task done.

The desktop assistant is **reactive** which means it knows human language very well and understands the context that is provided by the user and gives response in the same way, i.e. human understandable language, English. So the user finds its reaction in an informed and smart way.

The main application of it can be its **multitasking** ability. It can ask for continuous instruction one after another until the user quits it. It asks for the instruction and listens to the response that is given by the user without needing any **trigger phase** and then only executes the task.

7.3 SECURITY

The security testing mainly focuses on vulnerabilities and risks. As ZARA is a local desktop application, hence there is no risk of data breaching through remote access. The software is dedicated to a specific system so when the user logs in, it will be activated.

7.4 STABILITY

Stability of a system depends upon the output of the system, if the output is bounded and specific to the bounded input then the system is said to be stable. If the system works on all the poles of functionality then it is stable.

Chapter 8: Individual Contribution

The project titled **“A.I. DESKTOP VOICE ASSISTANT: ZARA”** was designed by me individually. From installing all the packages, importing, creating all the necessary functions, was all done by me individually.

I, myself have done all the research before making this project, designed the requirement documents for the requirements and functionalities, wrote synopsis and all the documentation, code and made the project in such a way that it is deliverable at each stage.

I have written the complete code in Python language and in PyCharm IDE from where it was very easy to install the packages and libraries, I have created the functions like `takeCommand()`, `wishMe()` and `taskExecution()` which has the following functionalities, like `takeCommand()` which is used to take the command as input through microphone of user and returns the output as string, `wishMe()` that greets the user according to the time like Good Morning, Good Afternoon and Good Evening and `taskExecution()` which contains all the necessary task execution definition like `sendEmail()`, `pdf_reader()`, `news()` and many conditions in if condition like “open Google”, “open notepad”, “search on Wikipedia” ,”play music” and “open command prompt” etc.

While making this project I realized that with the advancement ZARA can perform any task with the same effectiveness or can say more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time. Functionalities of this project include, It can send emails, It can read PDF, It can send text on WhatsApp, It can open command prompt, your favorite IDE, notepad etc., It can play music, It can do Wikipedia searches for you, It can open websites like Google, YouTube, etc., in a web browser, It can give weather forecast, It can give desktop reminders of your choice. It can have some basic conversation.

At last, I have updated my report and completed it by attaching all the necessary screen captures of inputs and outputs, mentioning the limitations and scope in the future of this project.

Chapter 9: Conclusion

ZARA is a very helpful voice assistant without any doubt as it saves time of the user by conversational interactions, its effectiveness and efficiency. But while working on this project, there were some limitations encountered and also realized some scope of enhancement in the future which are mentioned below:

9.1 LIMITATIONS

9.1.1. Security is somewhere an issue, there is no voice command encryption in this project.

9.1.2. Background voice can interfere.

9.1.3. Misinterpretation because of accents and may cause inaccurate results.

9.1.4. ZARA cannot be called externally anytime like other traditional assistants like Google Assistant can be called just by saying, "Ok Google!"

9.2 SCOPE FOR FUTURE WORK

9.2.1. Make ZARA to learn more on its own and develop a new skill in it.

9.2.2. Make Zara with a GUI interface to make it more eye-catchy.

9.2.3. ZARA android app can also be developed.

9.2.4. Make more voice terminals, like ZARA.

9.2.5. Voice commands can be encrypted to maintain security.

References :

- <https://pypi.org/project/>
- <https://www.lfd.uci.edu/~gohlke/pythonlibs/>
- <https://wiki.python.org/moin/Documentation>
- <https://www.twilio.com/>