**PROJECT REPORT  
Computer Science Department**

****

**Virtual Assistant**

Submitted by

**Asfand Yar**

**SUBJECT**

**Artificial Intelligence**

Faculty of Information & Communication Technology,   
BUITEMS, Quetta.

**Contents**

[1. Introduction 3](#_Toc184773414)

[2. Literature Review 3](#_Toc184773415)

[3. Methodology 4](#_Toc184773416)

[4. Code 4](#_Toc184773417)

[5. Flowchart: 16](#_Toc184773418)

[6. Component Description 17](#_Toc184773419)

[7. Results & Discussion 17](#_Toc184773420)

[8. Task Distribution: 18](#_Toc184773421)

[9. References: 20](#_Toc184773422)

**1. Introduction**

**Significance**

The purpose of this program is to create a voice-based virtual assistant named COSMOS that is capable of performing multiple tasks, such as searching the web, providing weather updates, sending messages, playing music, and reading PDFs. This virtual assistant integrates a combination of natural language processing, speech recognition, and text-to-speech functionalities to enhance user interaction. The significance of the program lies in its ability to provide an intuitive interface for users to interact with a variety of online resources and perform daily tasks more efficiently.

**Objective**

The primary objective of virtual assistant is to build a robust voice-activated assistant that can understand voice commands, perform actions based on those commands, and interact with external services like Chrome, weather APIs, and WhatsApp. By leveraging libraries like **pyttsx3** (text-to-speech), **speech\_recognition** (speech-to-text), and **Chrome**, the system aims to be a comprehensive assistant for basic and advanced tasks.

**Outline:** The report is organized as follows:

1. Introduction to the assistant program.
2. Literature review on virtual assistants and related works.
3. Methodology behind the development of the virtual assistant.
4. Code description and an explanation of how it works.
5. A flowchart of the system's operation.
6. Component description of different parts of the code.
7. Results and discussion regarding the program’s functionalities.
8. References to sources used in the project.

**2. Literature Review**

**Related Work**

Virtual assistants have been a significant topic of research and development in recent years. Popular systems such as Apple’s Siri, Google Assistant, and Amazon’s Alexa provide users with various functionalities like voice recognition, smart home control, and information retrieval. These systems rely on natural language processing (NLP) and machine learning (ML) to interpret user commands. The code provided aims to emulate some of these functionalities, specifically focusing on voice interaction, online search, and task automation, which are core features of modern virtual assistants.

**3. Methodology**

The system leverages several Python libraries to implement the core functionalities:

* **pyttsx3**: For text-to-speech conversion, enabling the assistant to speak responses.
* **speech\_recognition**: For converting user speech into text that can be processed by the program.
* **wikipedia**: For retrieving summarized information from Wikipedia based on user queries.
* **webbrowser**: For opening URLs like Google and YouTube based on user requests.
* **pywhatkit**: For sending messages on WhatsApp directly from the assistant.
* **pyautogui**: For GUI automation (keyboard and mouse control)
* **Os**: For System operations (e.g., closing programs)
* **Random**: For Generating random outcomes (coin toss, dice roll)
* **psutil**: For Battery status check

The assistant operates in a continuous loop, listening for voice input, processing the command, and providing the corresponding response. Commands are processed using basic string matching to trigger different functions such as weather updates, web searches, and more.

**4. Code**

import pyttsx3  
import speech\_recognition as sr  
import os  
import requests   
import datetime   
from datetime import date  
import time  
import warnings  
import random  
import wikipedia  
import webbrowser  
from pywhatkit import sendwhatmsg\_instantly  
import smtplib  
import sys  
import pyjokes  
import pyautogui  
import PyPDF2  
from tkinter.filedialog import \*  
import psutil  
import speedtest  
import wolframalpha  
import socket  
  
warnings.filterwarnings("ignore") #ignoring all the warnings  
  
if sys.platform == "win32":  
 engine=pyttsx3.init('sapi5')  
 voices=engine.getProperty('voices')  
 engine.setProperty('voice',voices[1].id)  
else:  
 engine=pyttsx3.init('nsss') #sapi5 - SAPI5 on Windows  
 voices=engine.getProperty('voices')  
 engine.setProperty('voice',voices[10].id)  
   
   
def speak(audio):  
 engine.say(audio)  
 print(audio)  
 engine.runAndWait()  
  
def take\_command():  
 r=sr.Recognizer()  
   
 with sr.Microphone() as source:  
 print('Go ahead,I am listening....')  
 r.adjust\_for\_ambient\_noise(source)  
 audio=r.listen(source)  
 try:  
 print('Hold on a momment,Recognizing...')  
 query=r.recognize\_google(audio,language='en-in')  
 print(f'User said:{query}\n')  
 except:  
 speak("Please hold on...")   
 return "None"  
 return query  
  
def wish():  
 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!")   
   
 speak("I am your Virtual Assistant. How may I help you")  
  
def sendEmail(to,content):  
 server=smtplib.SMTP("smtp.gmail.com",587)  
 server.ehlo()  
 server.starttls()  
 server.login("email","password")  
 server.sendmail("email id",to,content)  
 server.close()  
  
def news():  
 api\_key= 'dbc9e0bb65894032bfa6ff2df46f9785'  
 main\_url = f'http://newsapi.org/v2/top-headlines?sources=techcrunch&apiKey={api\_key}'  
  
 main\_page = requests.get(main\_url).json()  
 articles = main\_page["articles"]  
 head = []  
 numbers=["first","second","third","fourth","fifth"]  
   
 for ar in articles:  
 head.append(ar["title"])  
   
 for i in range (len(numbers)):  
 speak(f"today's {numbers[i]} news is: {head[i]}")  
  
def crypto(slug):  
 apiurl='https://pro-api.coinmarketcap.com'  
 headers = {'Accepts': 'application/json', 'X-CMC\_PRO\_API\_KEY': '1104789a-dbfe-4927-870d-4029f1d3cb17',}  
  
 session=requests.session()  
 session.headers.update(headers)  
  
 def coins\_price(apiurl,slug):  
 url=apiurl+'/v1/cryptocurrency/quotes/latest'  
 parameters={'slug':slug}  
 r=session.get(url,params=parameters)  
 data=r.json()['data']  
 all=str(data)  
 x=all.find('price')  
 all=all[x:x+20]  
   
 for p in all.split():  
 try:   
 float(p)  
 price=p  
 except:  
 pass  
 speak(f'{slug} price is {price}')  
 return price  
   
 coins\_price(apiurl,slug)  
  
def weather():  
 def loc():  
 try:  
 ipadd=requests.get("https://api.ipify.org").text  
 url="https://get.geojs.io/v1/ip/geo/"+ipadd+".json"  
 geo\_requests= requests.get(url)  
 geo\_data=geo\_requests.json()  
 city=geo\_data['city']  
 except:  
 city='delhi'  
 return city  
  
 api\_key = '86320b603f8e8b33555dad84e8bca164'  
 base\_url = 'https://api.openweathermap.org/data/2.5/weather?'  
 city\_name = loc()  
 url = base\_url + "&q=" + city\_name + "&appid=" + api\_key   
 session=requests.session()  
 r = session.get(url)  
 data = r.json()  
   
 if data["cod"] != "404":  
 y = data["main"]  
 current\_temperature = y["temp"]  
 current\_humidiy = y["humidity"]  
 z = data["weather"]  
 weather\_description = z[0]["description"]  
 speak(" Temperature is " +str(int(current\_temperature-273.15)) +" degree celcius\n humidity is " +  
 str(current\_humidiy) +"%\n with " + str(weather\_description)+'in '+city\_name)  
   
def pdf\_reader():  
 book=askopenfilename()  
 try:   
 pdfreader=PyPDF2.PdfFileReader(book)  
 pages=pdfreader.numPages  
 speak(f"Total numbers of pages in this pdf are {pages}")  
 speak("sir please enter the page number you want me to read")  
 pg=int(input("please enter the page number:"))  
   
 for num in range(pg,pages):  
 page=pdfreader.getPage(pg)  
 text=page.extractText()  
 speak(text)  
 except :  
 speak("Operation Cancelled !")   
 import speedtest  
  
def check\_internet\_speed():  
 try:  
 speak("Checking internet speed")  
 st = speedtest.Speedtest()  
 dl = round(st.download() / 1\_000\_000, 2) # Convert to megabits (Mb)  
 up = round(st.upload() / 1\_000\_000, 2) # Convert to megabits (Mb)  
 speak(f"Current download speed is {dl} Mb/s and upload speed is {up} Mb/s.")   
 speak("Do you want me to do anything else?")  
 except Exception as e:  
 speak("I encountered an issue while checking the internet speed.")  
 print(f"Error: {e}")   
   
def adv\_search():  
 query=input('Question: ')  
 app\_id='W3KQ4X-3WWETGEJQ2'  
 client=wolframalpha.Client(app\_id)  
   
 if 'no thanks' in query or 'thanks' in query or 'close advance search mode' in query:  
 speak('closing advance search mode')  
 else:  
 res=client.query(query)  
 ans=next(res.results).text  
 speak(ans)  
 speak('want to search anything else?')  
 adv\_search()   
  
def TaskExecution():  
 # function for coin toss task  
 def htLine1():  
 speak("It's " + res)  
 def htLine2():  
 speak("You got " + res)  
 def htLine3():  
 speak("It landed on " + res)  
  
 wish()  
 bye=True  
 while bye:  
 query=take\_command().lower()  
 #query=input()  
  
 if "what is your name" in query:  
 speak('I am Virtual Assistant.')  
 continue  
   
 if "tell me about yourself" in query:  
 speak('I am Virtual Assistant. What can I do for you?')  
 continue  
  
 elif 'price of' in query or 'tell me the price of' in query:  
 query=query.replace('tell me the price of ','')  
 query=query.replace('price of ','')  
 crypto(query)  
 speak('need something else?')  
  
 elif 'weather' in query or "show weather" in query:  
 weather()  
 speak('need something else?')  
  
 elif "open notepad" in query:  
 npath="C:\\WINDOWS\\system32\\notepad.exe"  
 os.startfile(npath)  
  
 elif "open calculator" in query or 'calculator' in query:  
 npath="C:\\WINDOWS\\system32\\calc.exe"  
 os.startfile(npath)  
  
 elif "open chrome" in query or 'chrome' in query:  
 npath="C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe"  
 os.startfile(npath)  
  
 elif "open command prompt" in query:  
 os.system("start cmd")   
 bye=False   
  
 elif 'the time' in query:  
 strTime=datetime.datetime.now().strftime('%H:%M')  
 speak(f'its {strTime}')  
 speak('you want me to do anything else?')  
  
 elif "todays date" in query or "the date"in query:  
 today = date.today()  
 d2 = today.strftime("%B %d, %Y")  
 speak(f"Today is {d2}")   
 speak('you want me to do anything else?')  
  
 elif "ip address" in query:  
 try:  
 hostname = socket.gethostname()  
 local\_ip = socket.gethostbyname(hostname)  
 speak(f"Your current internet connection's IP Address is {local\_ip}")  
 except Exception as e:  
 speak("I couldn't fetch your IP address. Please check your internet connection.")  
 speak('Do you want me to do anything else?')  
  
 elif 'wikipedia' in query:  
 speak('Searching in wikipedia')  
 query=query.replace('wikipedia',' ')  
 results=wikipedia.summary(query,sentences=2)  
 speak('According to wikipedia')  
 #print(results)  
 speak(results)  
 speak('you want me to do anything else')  
   
 elif 'open google' in query:  
 webbrowser.open("https://google.com")  
 bye=False  
  
 elif "email" in query:  
 try:  
 speak("To whom do you want to send mail?")  
 to=input("Enter the mail id to whom you want to send:")  
 speak("what should i say?")  
 subquery=take\_command().lower()  
 sendEmail(to,subquery)  
 speak("Email has been sent.")  
 speak('want to do anything else?')  
   
 except Exception as e:  
 speak("Sorry,The internet connection is not stable I am currently unable to send the email.Please try again later")   
 speak('do you want me to do anything else?')  
   
 elif 'open youtube' in query:  
 webbrowser.open('https://youtube.com')  
 bye=False  
  
 elif 'what is' in query:  
 result=wikipedia.summary(query,sentences=2)  
 speak(result)  
 speak('anything else?')  
   
 elif 'search in youtube' in query or 'open in youtube' in query:  
 query=query.replace('search in youtube',' ')  
 query=query.replace('open in youtube',' ')  
 webbrowser.open(f'https://www.youtube.com/results?search\_query={query}')  
 speak(f'searchin in youtube {query}')  
 bye=False  
  
 #walframalpha  
 elif 'advance search mode' in query or 'advanced search mode' in query:  
 speak('Advance search mode activated')  
 try:  
 adv\_search()  
 except Exception as e:  
 speak("Please hold on we are doing but there is unstable internet connection")   
 speak('do you want me to do anything else?')   
 continue  
  
 elif 'search' in query or 'search in google' in query or 'open in google' in query:  
 query=query.replace('search',' ')  
 query=query.replace('search in google',' ')  
 query=query.replace('open in google',' ')  
 webbrowser.open(f"https://google.com/search?q={query}")  
 speak(f'searching in google {query}')  
 bye=False  
  
  
 elif ("open gfg" in query or "open geeksforgeeks" in query):  
 webbrowser.open("https://www.geeksforgeeks.org")  
 bye=False  
  
 elif "send message on whatsapp" in query or 'send message' in query or 'message on whatsapp' in query:  
 speak("To whom should I send a message")  
 speak(" Please type the number ")  
 no=input("Enter the number:")  
 speak(" what should I send ?")  
 speak('You will have to scan for whatsapp web.')  
 subquery=take\_command().lower()  
 sendwhatmsg\_instantly(f"+92{no}",f"{subquery}")  
 bye=False  
  
 elif "close notepad" in query:  
 speak("okay sir, closing notepad")  
 os.system("taskkill/f /im notepad.exe")  
 speak('you want me to do anything else?')  
   
 elif ("close cmd"in query or "close command prompt" in query):  
 speak("okay sir, closing cmd")  
 os.system("taskkill /f /im cmd.exe")  
 speak('you want me to do anything else?')  
  
 elif 'joke' in query or 'jokes' in query:  
 joke = pyjokes.get\_joke('en','all')  
 #print(joke)  
 speak(joke)  
 speak('anything else?')  
  
 elif 'jobs' in query or 'job' in query or 'job recommandation' in query or 'work' in query:  
 platforms = ['linkedin', 'indeed', 'rozee', 'Mustaqbil', 'bayt','career okay']  
 speak("Select a platform that you prefer:")  
 print('\n'.join(platforms))  
 statement1 = take\_command().lower()  
 #statement1 = input()  
   
 if (statement1 == 0):  
 continue  
   
 if 'linkedin' in statement1 or 'LinkedIn' in statement1 or 'Linkedin' in statement1:  
 webbrowser.open\_new\_tab("https://www.linkedin.com/jobs")  
 speak("LinkedIn is open now")  
 break  
   
 elif 'indeed' in statement1:  
 webbrowser.open\_new\_tab("https://www.indeed.com/jobs")  
 speak("Indeed is open now")  
 break  
   
 elif 'rozee' in statement1:  
 webbrowser.open\_new\_tab("https://www.rozee.pk/")  
 speak("Rozee is open now")  
 break  
   
 elif 'mustaqbil' in statement1:  
 webbrowser.open\_new\_tab(  
 "https://www.mustakbil.com/")  
 speak("Mustaqbil is open now")  
 break  
   
 elif 'bayt' in statement1:  
 webbrowser.open\_new\_tab("https://www.bayt.com/en/pakistan/")  
 speak("Bayt is open now")  
 break  
   
 elif 'career okay' in statement1:  
 webbrowser.open\_new\_tab('https://www.careerokay.com/')  
 speak('Career Okay is open now')  
 break  
   
 else:  
 speak("Sorry we couldn't find your search!!!")  
 speak('you want me to do anything else?')  
 #time.sleep(3)  
   
 elif "shutdown the system" in query or "shutdown" in query:  
 os.system("shutdown /s /t 0")  
   
 elif "restart the system" in query:  
 os.system("shutdown /r /t 5")  
   
 elif "sleep the system" in query:  
 os.system("rundll32.exe powrprof.dll,SetSuspendState 0,1,0")  
   
 elif 'movie ticket booking' in query or 'movie booking' in query or 'movie ticket' in query:  
 speak('opening bookme')  
 webbrowser.open\_new\_tab("https://bookme.pk/movie-tickets-online")  
 speak(" Book me show website is open now")  
 bye=False  
  
 elif 'online courses' in query or 'course' in query:  
 platforms = ['coursera', 'udemy', 'edx', 'skillshare', 'datacamp', 'udacity']  
 speak("Select a platform that you prefer : ")  
 print("\n".join(platforms))  
 statement1 = take\_command().lower()  
   
 if statement1 == 0:  
 continue  
   
 if 'coursera' in statement1:  
 webbrowser.open\_new\_tab("https://www.coursera.org")  
 speak("Coursera is open now")  
 bye=False  
   
 elif 'udemy' in statement1:  
 webbrowser.open\_new\_tab("https://www.udemy.com")  
 speak("udemy is open now")  
 bye=False  
   
 elif 'edx' in statement1:  
 webbrowser.open\_new\_tab("https://www.edx.org/")  
 speak("edx is open now")  
 bye=False  
   
 elif 'skillshare' in statement1:  
 webbrowser.open\_new\_tab("https://www.skillshare.com")  
 speak("skill share is open now")  
 bye=False  
   
 elif 'datacamp' in statement1:  
 webbrowser.open\_new\_tab("https://www.datacamp.com")  
 speak("datacamp is open now")  
 bye=False  
   
 elif 'udacity' in statement1:  
 webbrowser.open\_new\_tab("https://www.udacity.com")  
 speak("udacity is open now")  
 bye=False  
   
 else:  
 speak("Sorry we couldn't find your search!!!")  
 speak('you want me to do anything else?')  
  
 elif 'train ticket booking' in query or 'train booking' in query or 'train ticket' in query or 'train ticket' in query:  
 speak('opening website for train ticket booking')  
 webbrowser.open\_new\_tab("https://www.pakrail.gov.pk/")  
 speak(" Railway website is open now, have a good journey !")  
 bye=False  
  
 elif 'bus ticket booking' in query or 'bus booking' in query or 'bus ticket' in query:  
 speak('opening website for bus ticket booking')  
 webbrowser.open\_new\_tab("https://bookme.pk/buy-bus-tickets-online")  
 speak(" Book me website is open now, have a good journey !")  
 bye=False  
  
 elif 'airplane ticket booking' in query or 'airplane booking' in query or 'airplane ticket' in query:  
 speak('opening website for airplane ticket booking')  
 webbrowser.open\_new\_tab("https://bookme.pk/book-flights-online")  
 speak(" Book me website is open now, have a good journey !")  
 bye=False  
  
 elif "hotel" in query or "hotel booking" in query:  
 speak('Opening go bookme.com.pk')  
 webbrowser.open\_new\_tab('https://bookme.pk/book-hotels-online')  
 bye=False  
  
 elif 'switch the window' in query:  
 if sys.platform == "win32":  
 pyautogui.keyDown("alt")  
 pyautogui.press("tab")  
 time.sleep(1)  
 pyautogui.keyUp("alt")   
 bye=False  
 else:  
 pyautogui.keyDown("command")  
 pyautogui.press("tab")  
 time.sleep(1)  
 pyautogui.keyUp("command")   
 bye=False  
  
 elif ("tell me news" in query or "news" in query):  
 speak("Please wait, Fetching the latest news")  
 news()  
 speak('need something else?')  
  
 elif ("tell me my location" in query or "location" in query):  
 speak("Hold on,Locating our current location")  
 try:  
 ipadd=requests.get("https://api.ipify.org").text  
 url="https://get.geojs.io/v1/ip/geo/"+ipadd+".json"  
 geo\_requests= requests.get(url)  
 geo\_data=geo\_requests.json()  
 city=geo\_data['city']  
 country=geo\_data['country']  
 speak(f"We are in {city},{country}")  
 speak('need something else?')  
  
 except Exception as e:  
 speak("Sorry,I am unable to locate our current location due to poor connectivity. Please try after sometime.")  
 bye=False  
  
 elif "take a screenshot" in query or "take screenshot" in query:  
 timestamp = datetime.datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")  
 speak("Taking a screenshot...")  
 time.sleep(3)  
 screenshot = pyautogui.screenshot()  
 save\_directory = "Screenshots"  
   
 if not os.path.exists(save\_directory):  
 os.makedirs(save\_directory)  
 file\_name = os.path.join(save\_directory, f"screenshot\_{timestamp}.png")  
 screenshot.save(file\_name)  
 speak(f"Screenshot taken and saved in the folder {save\_directory}.")  
 speak("Do you need anything else?")  
   
 elif "how much battery is left" in query or "how much power is left" in query or "battery" in query:  
 battery=psutil.sensors\_battery()  
 percentage=battery.percent  
 speak(f"We have {percentage} percent battery. ")  
 if percentage>=50:  
 speak("We have enough power to go on.")  
 elif percentage>=20 and percentage<50:  
 speak("You shall connect the system to a charging point")   
 elif percentage<20:  
 speak("Battery about to die,connect to a charging point as soon as possible")  
 speak('you want me to do anything else')  
  
 elif "internet speed" in query:  
 speak("Checking internet speed")  
 st=speedtest.Speedtest()  
 dl=round(float(st.download())/8000000,2)  
 up=round(float(st.upload())/8000000,2)  
 speak(f"Current downloading speed is {dl}mb/s while uploading speed is {up}")   
 speak('you want me to do anything else?')  
  
 elif "volume up" in query:  
 pyautogui.press("volumeup")  
 speak('you want me to do anything else?')  
 elif "volume down" in query:  
 pyautogui.press("volumedown")  
 speak('you want me to do anything else?')  
 elif "volume mute" in query or "mute" in query:  
 pyautogui.press("volumemute")   
 speak('you want me to do anything else?')  
  
 elif 'flip the coin' in query or 'toss the coin' in query or 'toss a coin' in query or 'flip a coin' in query:  
 chances = ['Heads', 'Tails']  
 res = random.choice(chances)  
 picLine = random.randint(1, 3)  
 lines = [htLine1, htLine2, htLine3]  
 lines[picLine - 1]()  
 speak('you want me to do anything else?')  
  
 elif 'dice' in query:  
 num = random.randint(1, 6)  
 speak("Your rolled " + str(num))   
 speak('you want me to do anything else?')  
  
 elif 'bye' in query or 'no' in query or ' no thanks' in query:  
 speak('Untill next time Bye Bye')  
 bye=False  
  
 else:  
 speak("Sorry,I don't know how to do that right now but i am still learning how to be more helpful")  
 speak('anything else?')  
 #time.sleep(2)  
   
if \_\_name\_\_=="\_\_main\_\_":  
 TaskExecution()

|  |
| --- |
| yes  No  Start  Initialize the search engine  Wish Function  (Greetings)  Start listening for commands  Command Detected?  Repeat?  Process command  yes  No  Execute Action  (open app etc)  Exit |

**5. Flowchart:**

Below is the flowchart outlining the system's logic:

**6. Component Description**

**Primary Data Structure**

This system doesn’t utilize advanced data structures like trees or linked lists. The primary structure used is simple conditional logic (if-else) to match commands and execute corresponding functions.

**Functional Modules**

The system is divided into multiple functional modules, each responsible for a specific task:

1. **Text-to-Speech**: Uses the **pyttsx3** library to convert text into speech for providing responses.
2. **Speech-to-Text**: Uses **speech\_recognition** to listen to and process the user’s voice.
3. **Web Search**: Uses **wikipedia** and **webbrowser** to perform online searches and retrieve information.
4. **Weather Information**: Queries an external API (OpenWeatherMap) to fetch current weather data based on location.
5. **WhatsApp Messaging**: Uses **pywhatkit** to send WhatsApp messages.

**Storage Mechanism:** There is no persistent storage implemented in this code. The assistant runs as a temporary process and doesn’t retain data beyond the current session. Any data such as weather information, search results, etc., are fetched in real-time.

**7. Results & Discussion**

**Results:** The assistant performs well for basic tasks such as:

* Providing real-time weather updates.
* Searching Wikipedia for information.
* Sending messages on WhatsApp.
* Fetching the current time, date and location.
* Open applications like chrome or YouTube.
* Performs shutting/restarting/sleeping of your PC.
* Sending an email.
* Telling a joke.
* Updating you about your system (IP address, battery).

The speech recognition is efficient, though it might occasionally struggle with unclear or noisy audio inputs. Text-to-speech responses are clear and provide an engaging interaction.

**Discussion:** The assistant can be further enhanced by:

1. **Adding Error Handling**: Better error handling for failed API calls or unrecognized commands.
2. **User Interface**: Integration of a GUI or web interface would make the system more user-friendly.
3. **Extended Functionality**: Adding more features like setting reminders or controlling IoT devices.

There are also opportunities to improve the voice recognition capabilities using more advanced NLP techniques and integrating machine learning models to better understand natural language.

**8. Task Distribution:**

1. **Amna Tahir:** Voice Interaction and Greetings

* **Responsibilities:** Libraries to Handle
* **pyttsx3**: Convert text to speech.
* **speech\_recognition**: Recognize user voice input.
* **os**: Basic operating system-level commands for interacting with files.
* **Functions to Implement**:
* **Speak**: Create a function to output audio responses.
* **take command**: Develop a function that listens to user commands via the microphone.
* **wish**: Write logic to greet the user based on the current time of day (e.g., "Good Morning," "Good Afternoon").
* **Task Execution Contribution**:
* Collaborate on integrating voice commands to trigger task execution seamlessly.

1. **Bibi Ruqia**: Communication and Information Retrieval

* **Responsibilities**: Libraries to Handle:
* **requests**: For fetching data from APIs (e.g., news, weather).
* **datetime**: To manage date and time operations.
* **random**: Generate random responses or decisions.
* **wikipedia**: Search Wikipedia for quick facts.
* **webbrowser**: Open web pages in the browser.
* **smtplib**: Sending emails.
* **Functions to Implement:**
* **Send email**: Enable email-sending capabilities using SMTP.
* **news**: Fetch and read the latest news headlines using an API.
* **crypto**: Fetch cryptocurrency prices.
* **weather**: Provide real-time weather updates.
* **Task Execution Contribution**:
* Help integrate APIs and web-based tasks into the main execution flow.

1. **Muhammad Balach:** File Handling and Utility Features

* **Responsibilities:** Libraries to Handle:
* **pyjokes**: Generate jokes for lighthearted interaction.
* **pyautogui**: Automate user interface actions.
* **PyPDF2**: Read and extract text from PDF documents.
* **Socket**: Implementation of IP Address.
* **Functions to Implement:**
* **pdf reader**: Write a function to read text from PDFs and output it using the Speak function.
* **Basic Command:** Shutdown, Restart, Sleep the system.
* **IP Address:** Identify IP Address of current internet connection.
* **Generate jokes:** using pyjokes.
* **Task Execution Contribution:**
* Integrate utilities like PDF reading and IP Address into the task execution flow.

1. **Asfand Yar:** Advanced Features and Coordination

* **Responsibilities:** Libraries to Handle:
* **Psutil**: For system monitoring (CPU, RAM usage, etc.).
* **Speed test**: To measure internet speed.
* **Wolfram alpha**: Advanced calculations or queries using WolframAlpha.
* **Sys**: Exit or control program execution.
* **Pywhat kit**: Send instant WhatsApp messages.
* **Functions to Implement:**
* **Advance search:** Use APIs like WolframAlpha for enhanced user queries.
* **WhatsApp:** Send message via whatsapp by adding number.
* **Check internet speed:** Test and display internet speed using the speedtest library.
* **Task Execution Integration:**
* Combine all functions into a unified task execution framework.
* Write logic to handle user that execute the functions peacefully

**9. References:**

1. Python documentation on [pyttsx3](https://pyttsx3.readthedocs.io/en/latest/).
2. Python documentation on [speech\_recognition](https://pypi.org/project/SpeechRecognition/).
3. Wikipedia API documentation: <https://wikipedia.readthedocs.io/en/latest/code.html>.
4. [PyWhatKit](https://github.com/Ankit404butfound/PyWhatKit) for WhatsApp messaging functionality.
5. OpenWeatherMap API documentation: <https://openweathermap.org/api>.
6. Wolfram Alpha API documentation: <https://products.wolframalpha.com/api/>.
7. GitHub Repository: <https://github.com/Faizan-Alam-1/Virtual-Assistant-.git>
8. ChatGpt: <https://chatgpt.com/share/6758981f-09dc-8002-9fbc-a84f1143d396>