A

Project On

VIRTUAL ASSITANT

Session: 2020-21

B.Tech 2nd Year

Department of Computer Science and Engineering

Bundelkhand Institute of Engineering and Technology

Jhansi (U.P.) India – 284128

Submitted to:

Er. Umakant Ahirwar

Submitted by:

Ankit Kumar(1904310011),

Vikas Maury(1904310064),

Hardik Jindal(1904310026),

Ajay Kumar Maurya(1904310008)

Sanskar Singh(2004310906)

Bundelkhand Institute of Engineering and Technology

Department of Computer Science and Engineering

Certificate

Certified that this is a bona-fide record of the project entitled "Virtual Assistant" completed successfully by **Vikas Maury (1904310064), Ankit Kumar(1904310011), Ajay Kumar Maurya(1904310008), Hardik Jindal(1904310026), Sanskar Singh(2004310906)** under the guidance of **Er. Umakant Ahirwar** during the academic year 2020-2021.

Verified by Guide:

……..……..…

Er. Umakant Ahirwar

(Department of Computer Science and Engineering)

Acknowledgement

We are greatly indebted to our Head of the Department **Prof. A.K. Solanki** for guiding us. The team is also grateful to our project guide, **Er. Umakant Ahirwar** and respected faculty Members **Dr. Yashpal Singh, Er. Shashank Gupta, Er. Anuj Kumar Yadav, Dr. Shailendra Pratap Singh, Dr. S.K. Gupta, Dr. R.N. Verma and Er. Shivam Pandey** for their indomitable contribution and guidance without which the completion of this project would have been impossible.

Our sincere thanks to all our teachers, seniors and colleagues whose help and guidance brought this project to successful completion.

Submitted By:-

Vikas Maury(1904310064)

Ankit Kumar(1904310011)

Ajay Kumar Maurya(1904310008)

Hardik Jindal(1904310026)

Sanskar Singh(2004310906)

Table of contents

S. No. **Content** **Page No.**

1. INTRODUCTION

A virtual assistant is a software agent that can perform tasks or services for an individual based on commands or questions. Sometimes the term "chatbot" is used to refer to virtual assistants generally or specifically accessed by online chat. In some cases, online chat programs are exclusively for entertainment purposes. Some virtual assistants are able to interpret human speech and respond via synthesized voices. Users can ask their assistants questions, control home automation devices and media playback via voice, and manage other basic tasks such as email, to-do lists, and calendars with verbal (spoken) commands. A similar concept, however with differences, lays under the dialogue systems.[1]

Capabilities and usage of virtual assistants are expanding rapidly, with new products entering the market and a strong emphasis on both email and voice user interfaces. Apple and Google have large installed bases of users on smartphones. Microsoft has a large installed base of Windows-based personal computers, smart-phones and smart speakers.

2. OBJECTIVE

The main objective of our Voice Assistant is to automate and help the user to do the basic things virtually. We generally spend lot of time in clicking and typing to do repetitive task. This problem can be solved by automating the tasks.

With the help of our Virtual Assistant users can do a lot of tasks just by their voice command. Since our program takes only voice command, so it becomes very useful for physically challenged people. Despite of this, user can do simple conversation with our voice assistant.

In future most of the manually done things are going to be automated using technology. Our project is intended to automate almost anything possible to help the user to save time and be efficient.

3. SCOPE OF THE PROJECT

- Our project "Virtual Assistant" is particularly based on Automation so it can be developed further according to user's need.

- Our program is written in python and it is very easy to understand. It contains some very powerful python modules like os module which can use the window terminal to execute desired command.

- Voice interface technology has come of age and is now found on a wide variety of products such as phones, tablets, PCs, TV remotes and many other devices.

4. OVERALL DESCRIPTION

4.1. Hardware and Software Requirement

Software Requirement:-

- Operating System:- Windows 10 (64-bit)
- Python 3.8
- Audio Recognition software
- Internet connection.

Hardware Requirement:-

- At least 500MB of free space.
- At least 2 GB RAM
- Mic (for Audio input)
- Speaker (for Audio Output)

4.2. Summary of the project

Virtual Assistant is a software application which is intended to automate some repetitive and time consuming task. It takes user's voice command as it's input. We can send email, store messages, set alarm and countdown, read news and weather forecasts and much more.

4.3. Advantage of the Project

- It will help in increasing the efficiency of the user.
- It does not require keyboard or mouse input to give command.
- It can be also used by physically impaired as well as visually impaired people.
- It's news feature can be used to hear latest news from Google news API.
- Weather forecast can be listened from a very trusted weather API.
- Send email without typing anything.
- Set Alarm and set Brightness of Screen.
- Open your secret directory or most visited directories.

- Project can be easily modified with the help of it's source code and updated.

5. METHODOLOGY

5.1. Process Flow diagram:-

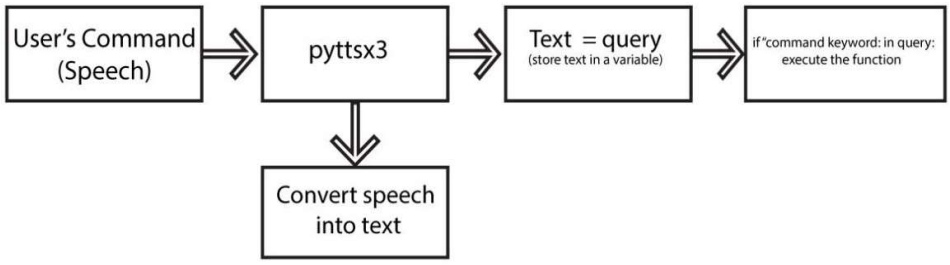

Fig.5.1 Process Flow diagram

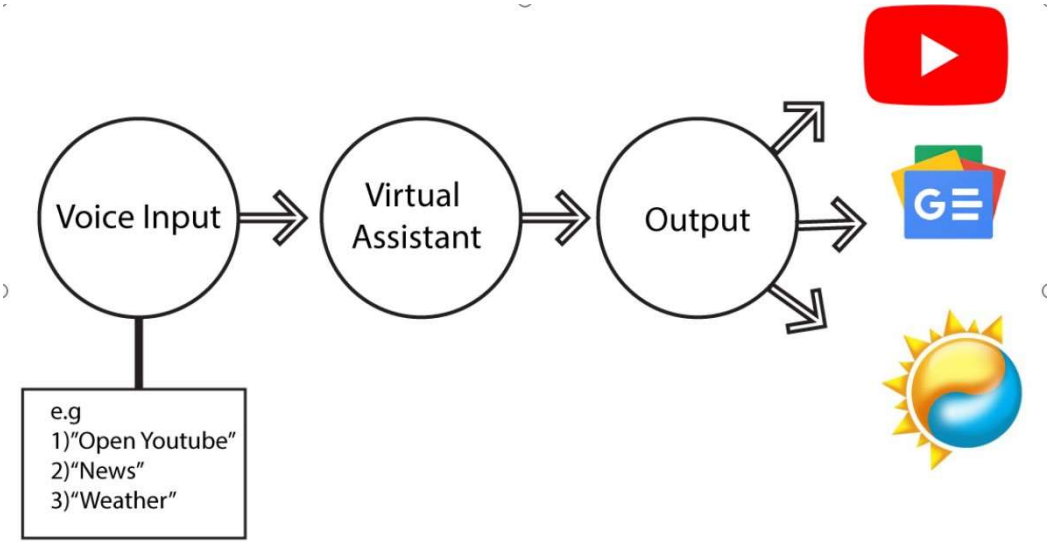

Fig. 5.2 Flow diagram of 3 tasks of Assistant

User's command are input to our Virtual Assistant. For example if the user says "Open YouTube", Virtual Assistant will take the input and open YouTube.

Every input is checked inside if-else statements and if it finds keyword which is available in program, the function is evoked.

6. CODING

```
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)


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

Fig. 6.1

pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible with both Python 2 and 3.[2]

Here we have created the ***speak*** fuction with help of it to produce the output in audio format.

```
def countdown(content):
    while content:
        mins, secs = divmod((content), 60)
        timer = '{:02d}:{:02d}'.format(mins, secs)
        print(timer, end="\r")
        time.sleep(1)
        content -= 1
        winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
    speak("Completed")
```

Fig. 6.2

Here ***countdown*** function takes time input from user and start timer. After completion it inform user with an alert sound.

```
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("I am Jarvis Sir. Please tell me how may I help you?")
```

```
    speak("I am Jarvis Sir. Please tell me how may I help you?")
```
Fig. 6.3

wishMe function runs at the beginning to wish the user with appropriate greeting. In morning, it greets user by “Good Morning” and in Afternoon “Good Afternoon” and so on.

```
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(e)
        print("Say that again please...")
        return "None"
    return query
```
Fig. 6.4

takeCommand is a very crucial part of our program. It’s function is to listen the user’s command using system’s Microphone. Try catch is used to handle the Exception.

```
def sendEmail(to, content):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login('qwertyforwork@gmail.com', 'jyppeoscgghmqlbf')
    server.sendmail('qwertyforwork@gmail.com', toaddr, content)
    server.close()
```
Fig. 6.5

sendEmail function uses **smtp(Standard mail transfer protocol)** to send the email. Content of the email and recievers email address are argument of this function.[3]

```
if 'youtube' in query:
            listening = True
            print("What should I search?")
            speak("What should I search?")
            content = takeCommand()
            url = 'https://www.youtube.com/results?search_query=' + content
            webbrowser.open(url)
            print('Done')
            speak('Done')
```

Fig. 6.6

User's input are stored in **query.** The above code means that if user's instruction sentence contains word 'youtube', it will automatically execute everything under this **if** statement. Further user will be asked what to search.

```
        elif "google" in query:
            listening = True
            url = 'https://www.google.com/'
            webbrowser.open(url)
            print('Done')
            speak('Done')
```

Fig. 6.7

Else if user says open Google, then user is redirected to Google home page in web browser.

```
        elif "calculator" in query:
            listening = True
            speak("Okay Sir, I am opening calculator")
            os.system("C:\\Windows\\System32\\calc.exe")
        elif "notepad" in query:
            listening = True
            speak("Okay Sir, I am opening notepad")
            os.system("C:\\Windows\\System32\\notepad.exe")
        elif "paint" in query:
            listening = True
            speak("Okay Sir, I am opening paint")
            os.system("C:\\Windows\\System32\\mspaint.exe")
```

Fig. 6.8

To open calculator, notepad and paint, user needs to speak '**calculator**','**notepad**', and '**paint**' respectively. These are some daily used apps.

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

Fig. 6.9

It speaks out the current time to the user.

```
        elif 'email' in query:
            try:
                speak("Whom should i sent email to?")
                to = takeCommand()
                edict = {'Shiv': 'mauryashivanand3@gmail.com',
                         'Ankit': 'akumar397800@gmail.com', }
                toaddr = edict[to]
                speak("What should I say?")
                content = takeCommand()
                sendEmail(toaddr, content)
                speak("Email has been sent!")
            except Exception as e:
                print(e)
                speak("Sorry sir , I can not send this")
```

Fig. 6.10

The above code asks the user receiver's name and the subject line with content. After that message sent confirmation is received by user. If the email sending was unsuccessful then it will say

"Sorry sir , I can not send this".

```
        elif "how are you" in query:
            listening = True
            speak("I am fine sir")
        elif "what is your name" in query:
            listening = True
            speak("I don't have a name yet")
        elif "who are you" in query:
            listening = True
            speak("I am Virtual Assistant")
        elif "why virtual friend" in query:
            listening = True
```

```
            speak("Because i will help you to make your life simple")
```
Fig 6.11

The above line of code is to communicate with user. This type of conversation makes our Virtual Assistant more user-friendly.

```
        elif "sound" in query:
            listening = True
            winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
            speak("Done sir!")
```
Fig. 6.12

You can check your speakers by saying **sound**. It will produce windows sound.[5]

```
        elif "message" in query:
            listening = True
            speak("What message you want to store?")
            content = takeCommand()
            speak("Alright!")
            sand.append(content)
```
Fig. 6.13

We can store message by above code.

```
        elif "inbox" in query:
            listening = True
            for x in sand:
                speak(x)
                speak("do you want to delete last messege")
                query = takeCommand()
                if "yes" in query:
                    sand.remove(x)
                if "no" in query:
                    speak("Message not deleted")
                    pass
```
Fig. 6.14

To retrieve the saved message, users need to say **inbox.** Voice Assistant will speak your message for you.


```
        elif "write" in query:
            listening = True
```

```
            speak("speak")
            query = takeCommand()
            f = open("new.txt", "w")
            f.write(query)
            f.close()
```

Fig. 6.15

The above code will help to write a document in .txt extension. This is good use of speech-to-text functionality.

```
        elif "countdown" in query:
            listening = True
            speak("Tell me the time:")
            content = takeCommand()
            try:
                countdown(int(content))
            except:
                speak("try again")
```

Fig. 6.16

The above code asks user to give time input and initiate the countdown function.

```
        elif 'wikipedia' in query:
            speak('What should I search')
            query = takeCommand().lower()
            results = wikipedia.summary(query, sentences=3)
            speak("According to Wikipedia")
            print(results)
            speak(results)
```

Fig. 6.17

The above code uses wikipedia module and extract out 3 paragraph and read them.

```
        elif 'weather' in query:
            print("Which city Sir?")
            speak("Which city Sir?")
            api_key = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
            base_url = "http://api.openweathermap.org/data/2.5/weather?"
            city_name = takeCommand()
            print(str(city_name)+" Weather report:\n")
            complete_url = base_url + "appid=" + api_key + \
                "&q=" + city_name + "&units=metric"
            response = requests.get(complete_url)
            x = response.json()
```

```
            # print(x)
            if x["cod"] != "404":
                y = x["main"]
                current_temperature = y["temp"]
                current_pressure = y["pressure"]
                current_humidiy = y["humidity"]
                z = x["weather"]
                weather_description = z[0]["description"]
                print(" Temperature (Celcius) = " +
                      str(current_temperature) +
                      "\n atmospheric pressure (hPa) = " +
                      str(current_pressure) +
                      "\n humidity () = " +
                      str(current_humidiy) + "%" +
                      "\n description = " +
                      str(weather_description))
                speak(" Temperature (in celcius unit) = " +
                      str(current_temperature) +
                      "\n atmospheric pressure (in hPa unit) = " +
                      str(current_pressure) +
                      "\n humidity (in percentage) = " +
                      str(current_humidiy) +
                      "\n description = " +
                      str(weather_description))
            else:
                print(" City Not Found ")
```

Fig. 6.18

In above code, OpenWeatherMap API has been used to get important weather information.[4]

e.g. Atmospheric Pressure, Humidity, brief Description of current weather.

```
        elif 'play music' in query:
            music_dir = 'D:\\MUSIC\\My Downloaded Music'
            songs = os.listdir(music_dir)
            # print(songs)
            os.startfile(os.path.join(music_dir, songs[0]))
```

Fig. 6.19

The execution of above code plays a song from your music directory.

```
        elif "news" in query:
            news_url = "https://news.google.com/news/rss"
            Client = urlopen(news_url)
```

```
            xml_page = Client.read()
            Client.close()
            count = 0
            speak("How many news headline sir?   ")
            limit = takeCommand()
            soup_page = soup(xml_page, "lxml")
            news_list = soup_page.findAll("item")
            # Print news title, url and publish date
            for news in news_list:
                print(news.title.text)
                print(news.link.text)
                speak(news.title.text)
                speak(news.link.text)
                count += 1
                if count > int(limit):
                    break
```

Fig. 6.20

The function of the above code is to read the latest news from Google news. It asks user number of headlines to read.

```
        elif "exit" in query:
            speak("See you later, Sir")
            exit()
```

Fig. 6.21

To exit from the Voice Assistant program **exit** command can be used.

```
        elif "shutdown" in query:
            speak("Do you wish to shutdown your computer ? (yes or no): ")
            shutdown = takeCommand().lower()
            if shutdown == 'yes':
                os.system("shutdown /s /t 1")
            else:
                Pass
```

Fig 6.22

To shutdown the computer we can give assistant command “shutdown”. It prompts user if he/she really want to shutdown his/her PC.

```
        elif "restart" in query:
            speak("Do you wish to restart your computer ? (yes / no): ")
            shutdown = takeCommand().lower()
```

```
            if shutdown == 'yes':
                os.system("shutdown /r /t 20")
                speak("restarting in 20 seconds")
            else:
                pass
```

Fig 6.23

Like Shutdown function, we have restart function which too asks user if he/she really want to restart their PC.

```
        elif "sleep" in query:
            speak("Do you wish to sleep your computer ? (yes / no):
")
            shutdown = takeCommand().lower()
            if shutdown == 'yes':
                os.system("shutdown /h /t 2")
            else:
                pass
```

Fig. 6.24

We can also put our computer on sleep mode with above code.

```
        elif "alarm" in query:
            speak("Please enter time manually")
            alarm_hour = int(input("Set hour: "))
            alarm_minutes = int(input("Set minutes: "))
            am_pm = input("am or pm? ")
            winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
            print(f"Waiting for time: {alarm_hour}:{alarm_minutes} {am_pm}")
            if am_pm == 'pm':  # to convert pm to military time
                alarm_hour += 12
            elif alarm_hour == 12 and am_pm == 'am':

                alarm_hour -= 12
            else:
                pass
            while True:
            #infinite loop that runs until alarm time
                if alarm_hour == datetime.datetime.now().hour and alarm_minutes == datetime.datetime.now().minute:
                    print("\nIt's the time!")
                    winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
                    winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
                    winsound.PlaySound("SystemQuestion", winsound.SND_ALIAS)
                    break
```

Fig 6.25

The function of above code is to Set alarm in your voice assistant. Users have to enter time manually. When it's time, Assistant alert the user with sound.

```
        elif "brightness" in query:
            ans = "no"
            curr_brightness = sbc.get_brightness()
            while(ans != "yes"):
                speak("How much brightness do you want?")
                try:
                    br = takeCommand()
                    sbc.set_brightness(br)
                    speak("Do you want this much ??")
                    ans = takeCommand().lower()
                    if (ans == "no"):
                        sbc.set_brightness(curr_brightness)
                except:
                    speak("Please speak clearly")
                    ans = "no"
            speak("Brightness has been set to " + curr_brightness)
            print(curr_brightness)
```

Fig 6.26

With "Brightness" command the above code is executed. It sets Brightness of the screen according to the user's need. It keeps confirming until user is satisfied with the brightness intensity.

7. SCREENSHOTS

1) Sending Email

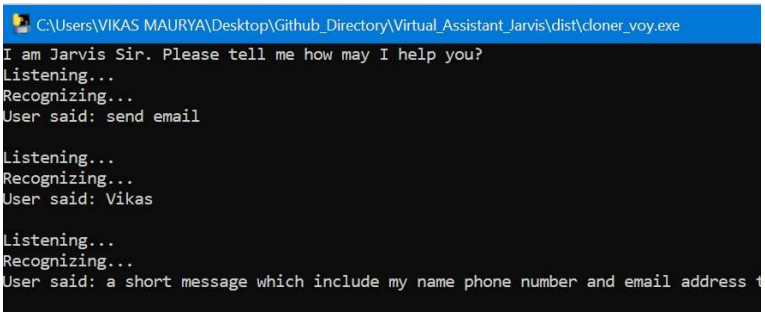

Fig.7.1 Sending Email

2) Store and Read Messages

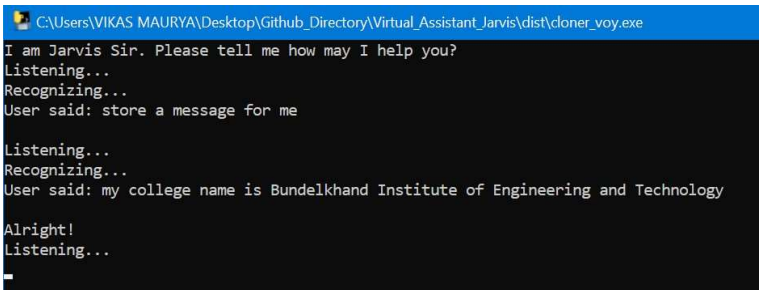

Fig. 7.2.1 Storing Message

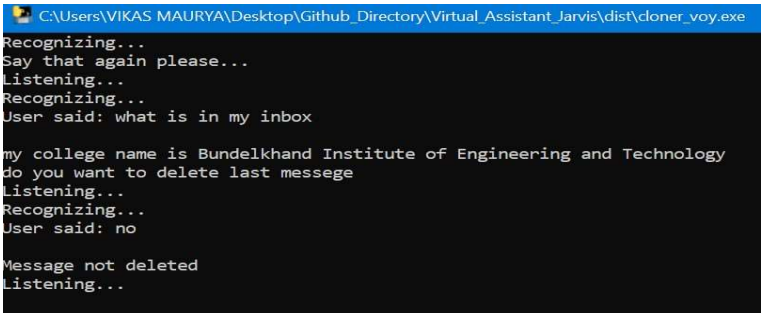

Fig. 7.2.2 Reading stored message

3) Open Google and YouTube

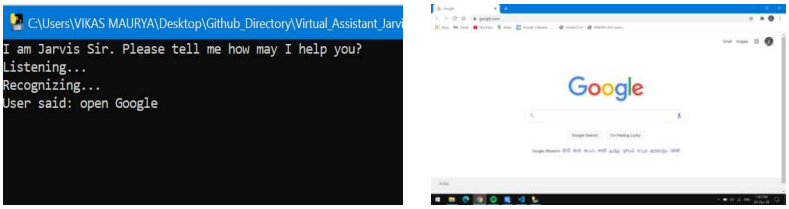

Fig. 7.3.1 Opening Google Search Home page

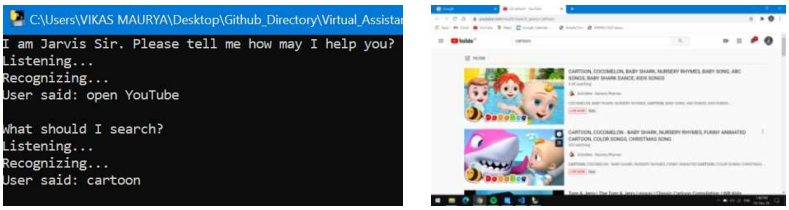

Fig. 7.3.2 Opening YouTube with a topic

4) Read News Headlines

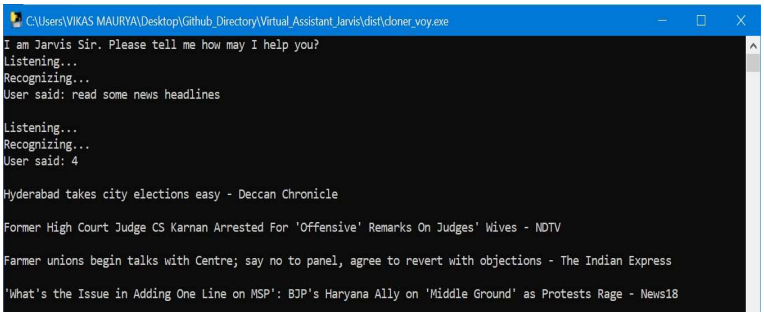

Fig. 7.4 Reading news directly from Google news

5) Write a file (.txt)

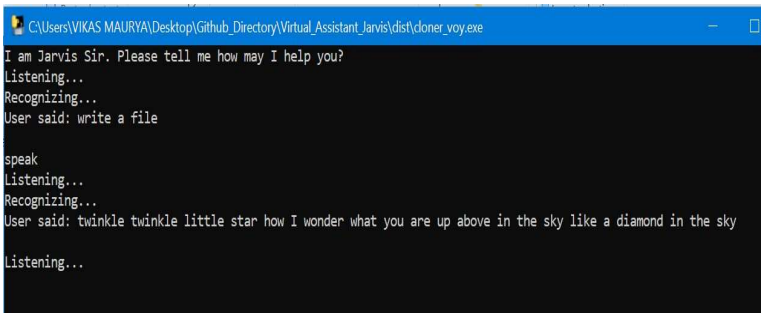

Fig. 7.5 Writing a text file

6) Weather forecast

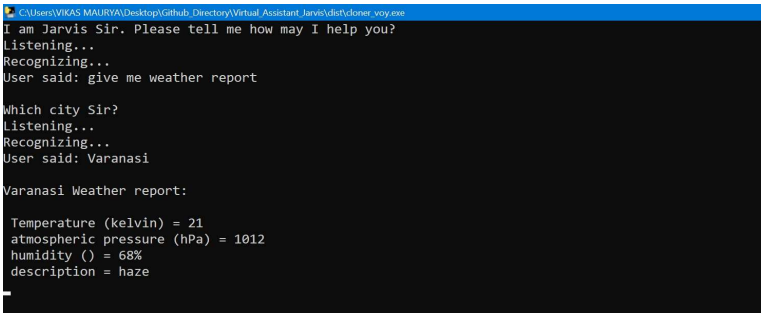

Fig. 7.6 Weather report

7) Power option(Shutdown and Restart)

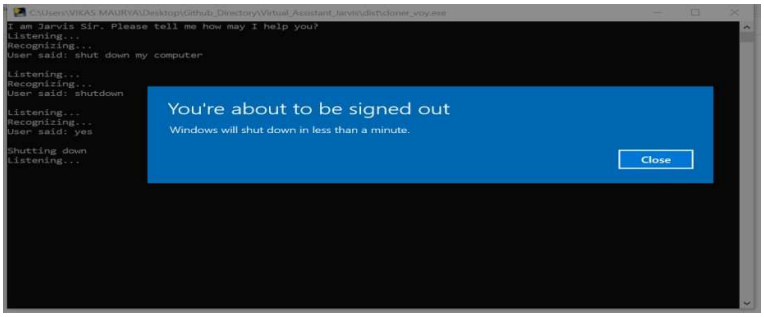

Fig. 7.7 Shutting down the computer

8) Alarm

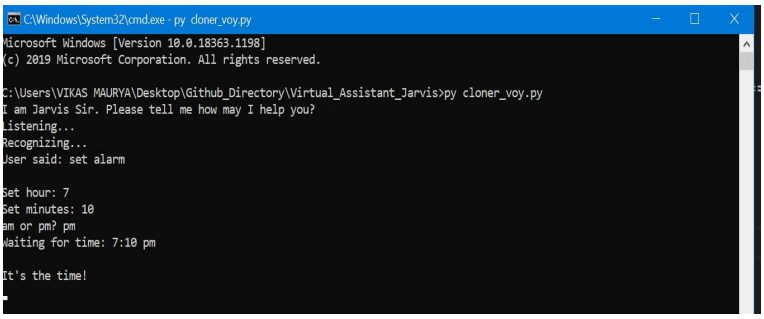

Fig. 7.8 Alarm Clock with alert sound

9) Wikipedia

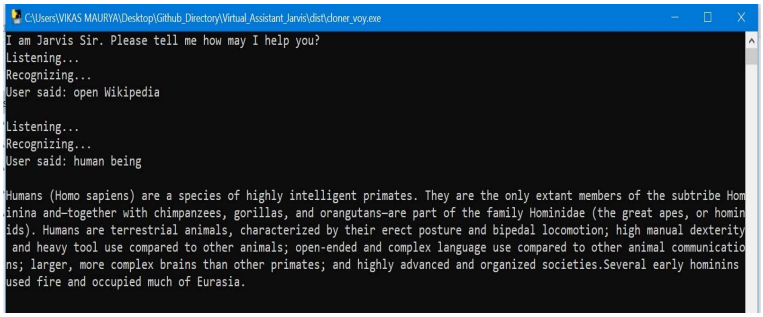

Fig. 7.9 Searching a topic on Wikipedia

10) Play Music

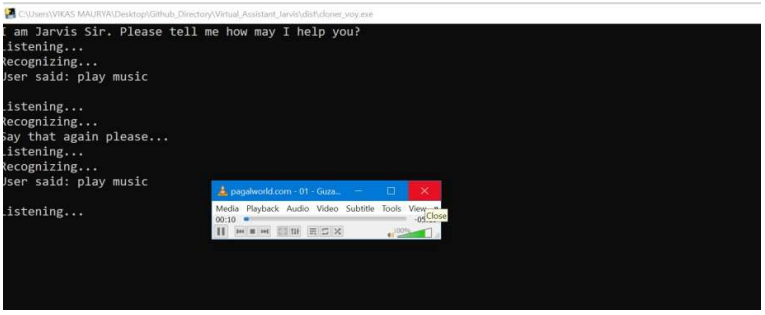

Fig. 7.10 Playing music

11) Set Brightness

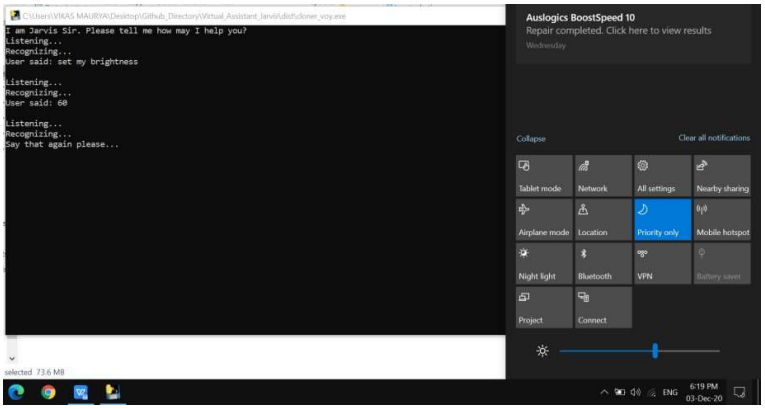

Fig. 7.11 Setting brightness

8. TESTING

Testing is a process of executing a program with the aim of finding error. To make our software perform well it should be error free.If testing is done successfully it will remove all the errors from the software.

8.1. UNIT TESTING

Unit testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

In SDLC, STLC, V Model, Unit testing is first level of testing done before integration testing. Unit testing is a WhiteBox testing technique that is usually performed by the developer. Though, in a practical world due to time crunch or reluctance of developers to tests, QA engineers also do unit testing

Unit testing Unit is important because software developers sometimes try saving time doing minimal unit testing and this is myth because inappropriate unit testing leads to high cost Defect fixing during System Testing, Integration Testing and even Beta Testing after application is built. If proper unit testing is done in early development, then it saves

time and money in the end. Here are the key reasons to perform unit testing.

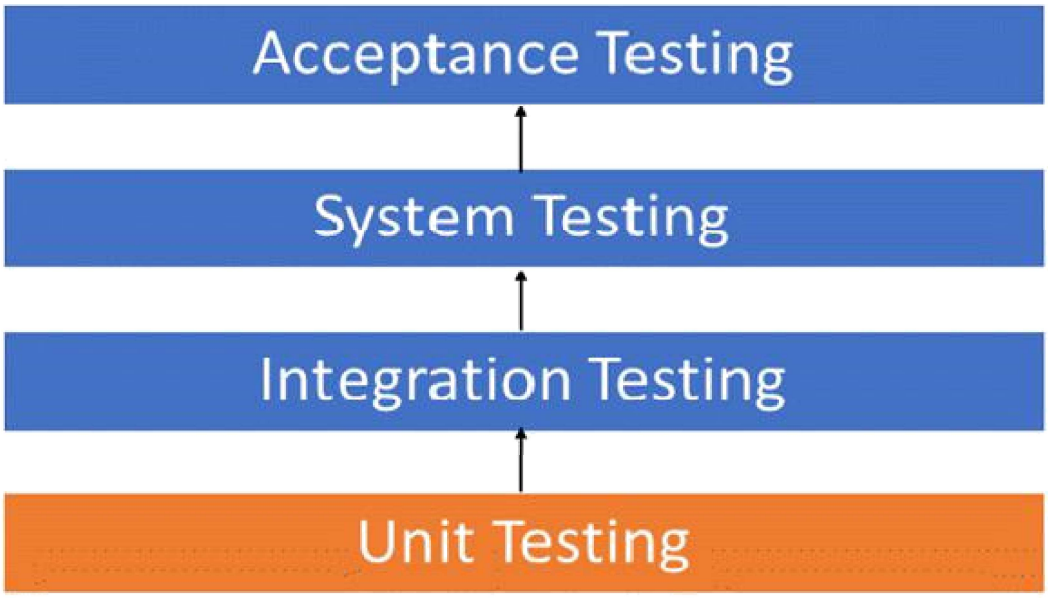

Fig 8.1

1.Unit tests help to fix bugs early in the development cycle and save costs.

2.It helps the developers to understand the code base and enables them to make changes quickly

3.Good unit tests serve as project documentation

4.Unit tests help with code re-use. Migrate both your code and your tests to your new project. Tweak the code until the tests run again.

8.2. INTEGRATED TESTING

Integration testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction

between these software modules when they are integrated.

Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as 'I & T' (Integration and Testing), 'String Testing' and sometimes 'Thread Testing'.

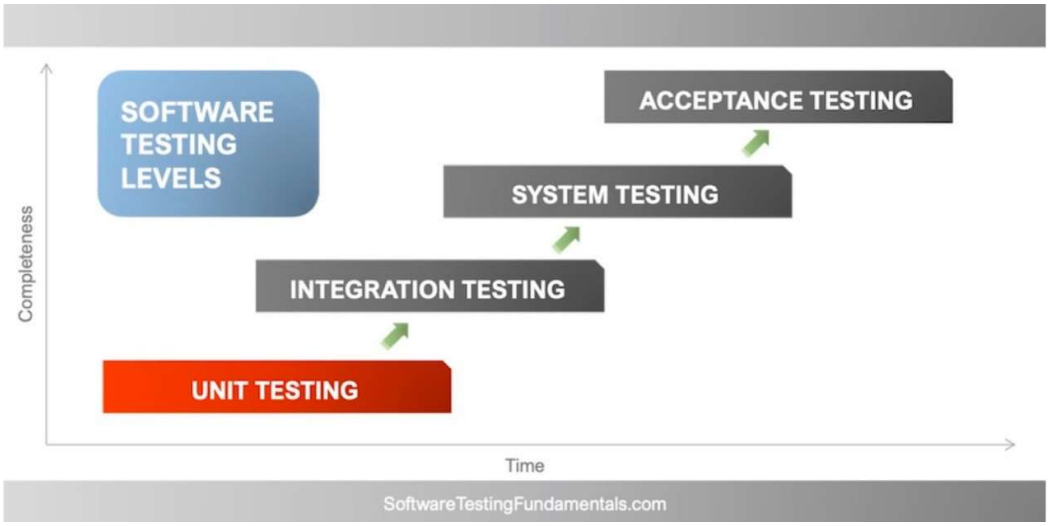

Fig. 8.2

Although each software module is unit tested, defects still exist for various reasons like -

• A Module, in general, is designed by an individual software developer whose understanding and programming logic may differ from other programmers. Integration Testing becomes necessary to verify the software modules work in unity.

• At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.

• Interfaces of the software modules with the database could be erroneous

• External Hardware interfaces, if any, could be erroneous

• Inadequate exception handling could cause issues.

8.3. BLACK BOX TESTING

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

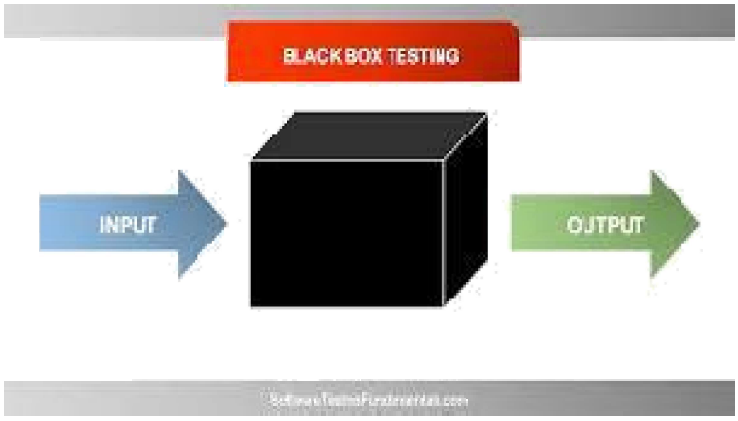

Fig.8.3

The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Here are the generic steps followed to carry out any type of Black Box Testing Initially, the requirements and specifications of the system are examined.

• Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.

• Tester determines expected outputs for all those inputs.

• Software tester constructs test cases with the selected inputs.

• The test cases are executed.

• Software tester compares the actual outputs with the expected outputs.

• Defects if any are fixed and re-tested.

9. BENEFIT OF THE PROJECT:-

• **Sending Email** - This will help you to send someone email using your credentials and voice input, without giving much effort for manually typing it.

• **Writing File** - By just verbally giving the proper command, the user can write a text file. The contents of the file will also be taken by voice input.

• **Weather forecast –** Weather information like Temperature, humidity, pressure can be retrieved from the assistant with the help of API.

• **Power Option –** Computer can be shut down or restart by assistant.

• **Set Brightness –** Screen brightness can be adjusted with voice assistant.

• **Set Alarm –** Alarm setting facility is also present.

- **Current Time -** This voice assistant can help the user to get to know about the current time just by asking to the assistant.
- **News Headline –** No need to search for news. Listen it directly via Google news.
- **Conversation with user –** Conversation with our Assistant is also possible.

10. CONCLUSION

Time is precious and this project will let user save their time efficiently. Voice assistant can be personalized for a user's day to day life. There is tremendous possibility of growth of this virtual assistant. It will reduce dependency on human resources This project is very beneficial for differently abled people due to the fact that this project can use voice or text for communicating with the user. Also this Voice Assistant can be used for personal as well as commercial use.

11. REFERENCE

[1] https://en.wikipedia.org/wiki/Virtual_assistant

[2] https://pypi.org/project/pyttsx3/

[3] https://docs.python.org/3/library/smtplib.html

[4] https://openweathermap.org/api

[5] https://www.geeksforgeeks.org/python-winsound-module/