

TABLE OF CONTENTS

1	Introduction
1.1	Introduction
1.2	Background
1.3	Objectives
1.4	Purpose and Scope
1.4.1	Purpose
1.4.2	Scope
2	Survey of Technologies
2.1	Justification of Selection of Technology
3	System and Analysis
3.1	Problem Definition
3.2	Proposed System
3.3	Requirement Analysis
3.4	Hardware Requirements
3.5	Software Requirements
3.6	Module Division
4	System Design
4.1	Basic Idea Diagram
4.2	Flow Chart Diagram
4.3	Use Case Diagram
4.4	Package Diagram
4.5	Class Diagram
4.6	Activity Diagram
5	Implementation Approaches
5.1	Coding details and code efficiency
5.2.1	Code Efficiency
5.3	Testing Approach
5.3.1	Unit Testing
5.3.2	Integrated Testing
5.3.3	Beta Testing
5.4	Modifications and Improvements

5.5	Test Cases
6	Results and Discussion
6.1	Test Reports
6.2	User Documentation
7	Conclusions
7.1	Conclusions
7.1.1	Significance of the system
7.2	Limitations of the system
7.3	Future Scope of the Project
8	References

1.INTRODUCTION

An **intelligent voice assistant (IVA)** or **intelligent personal assistant (IPA)** is a software agent that can perform tasks or services for an individual based on verbal commands.

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 commands.

As of 2017, the capabilities and usage of virtual assistants are expanding rapidly, with new products entering the market and a strong emphasis on 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, smartphones and smart speakers. Amazon has a large install base for smart speakers.

1.2 Background

Voice recognition technology was around long before Apple's Siri debuted in 2011. At the Seattle World's Fair in 1962, IBM presented a tool called Shoebox. It was the size of a shoebox and could perform mathematical functions and recognize 16 spoken words as well as digits 0-9.

In the 1970s, scientists at Carnegie Mellon University in Pittsburgh, Pennsylvania — with the substantial support of the United States Department of Defense and its Defense Advanced Research Projects Agency (DARPA) — created Harpy. It could recognize 1,011 words, which is about the vocabulary of a three-year-old.

Once organizations came up with inventions that could recognize word sequences, companies began to build applications for the technology. The Julie doll from the Worlds of Wonder toy company came out in 1987 and could recognize a child's voice and respond to it.

Throughout the 1990s, companies like IBM, Apple, and others created items that used voice recognition. Apple began building speech recognition features into its Macintosh computers with PlainTalk in 1993. In April 1997, Dragon came out with Dragon NaturallySpeaking, which was the first continuous dictation product. It could understand about 100 words per minute and turn it into text. Medical dictation devices were one of the earliest adopters of voice recognition technology.

Timeline of Mainstream Voice Assistants



Technology companies are working to create increasingly sophisticated technology that will automate more processes and tasks we do throughout the day. Even Siri, Google Assistant, and Alexa can “learn” new words and tasks.

1.2 Objectives

- The main objective is to make user friendly computer.
- Both Technical as well as Non-Technical person can use.
- No age limit.
- Fully voice controlled.
- Any Python language lover can customize.
- Easy to use.
- Can perform multiple tasks.
- Time saver.
- Software based application.
- Future.

1.3 Purpose and Scope

1.3.1 Purpose

- **Time saver-** As it is voice controlled, and if user need to draft a mail then user can just gave a command “Draft a mail for me” and M.O.N.T.Y Voice Assistant will draft a mail on behalf of user which will reduce the time.
- **Office friendly-** This M.O.N.T.Y Voice Assistant (AI) can be use in office as well. Eg. user can just say a command for.
- **Home friendly-** As we are aware with Amazon echo same we can use this application with some additional components.

1.3.2 Scope

- As we know that AI is the future of the digital age.
- We can hope that we can use this application in various fields.
- For example we can use this application at our home.
- As it is voice controlled we just need to give command and it will work accordingly.
- It can as use for home safety system by enabling secured mode in application.

2.SURVEY OF TECHNOLOGIES

2.1 Justification of Selection of Technology

As this M.O.N.T.Y Desktop Assistant (AI) is made in Python language, which has dynamic modules, easy to use, less lines of code, easy and simple syntax, and can even run in lower versions of windows ie can even run on windows XP.

Python language can be use as Front end as well as for backend which makes easier for any developer to work on python projects.

The main reason for selection of this technology is, as python is most loved and preferred language and has much advantage as mentioned above. Working on Python language is time saving and fun as number of line reduce not like in C/C++.

Python is supported in all most all operating system, so there will be no issue of selecting operating system.

Developer can develop in Linux and can run python code in windows operating system.

Python has extensive support libraries; the best thing of python is, it is open source and has community development. Python has user friendly data structure.

3.SYSTEM AND ANALYSIS

3.1 Problem Definition

- It should support on any windows operating system including XP,7,8 and 10.
- Desktop should perform multitasks.
- It should have accuracy as user tuned its settings.
- It should be customizable.
- It should be easy to use.
- It should perform tasks very smoothly.
- It should have consistency.
- It should be easy to understand and easy to control.

3.2 Proposed System

- Desktop Assistant is made in Python language and majority of operating system (OS) supports Python language.
- Can run in all OS as majority of OS supports Python.
- This Assistant can perform multiple task like plays music, weather forecast and can even communicate with users(ie can answer personal questions)
- This Assistant can run even in lower version of windows and other OS
- Not like Cortana which is only available in windows 10, this Assistant can run even in windows XP.
- Accuracy can be tuned as per need.
- One step enables.
- If the user is programmer, He/she can customize as per own choice.
- Easy to use by one click
- Fully voice controlled.

3.3 Requirement Analysis

- **For developer requirements-**
- Python 3.0(preferred)
- Python modules like pyaudio, gTTS, winspeech, speech recognition, google speech recognition, pygame, pyttsx3 and as per requirement modules can be downloaded by using pip.
- Operating System- doesn't require any specific OS but for easy and faster development developer can use Windows 7 or higher version. Developer can even use Windows XP.
- **For user point of view Requirements-**
- Just internet connectivity required.
- Any Operating system laptop or PC.
- Required a good quality mic or in-build microphone so that Desktop Assistant (AI) can listen to user command and respond back effectively.

3.4 Hardware Requirements

- **For Developer requirement-**

- Regular laptop or Pc.
- Laptop/Pc specification:
- Processor: Intel i5 CPU @ 1.7 GHz (min)
- RAM: 4GB(MIN)
- Hard disk: 80GB min
- Monitor: As per user
- Keyboard: As per user
- Mouse: As per user
- Internet connectivity.
- High quality microphone.

- **For user end requirement-**

- Internet connectivity.
- High quality (preferred for optimal use) or in-build microphone.
- Laptop or Pc with minimum of 4GB RAM.
- Intel or AMD processor.
- Graphic card optional.

3.5 Software Requirements

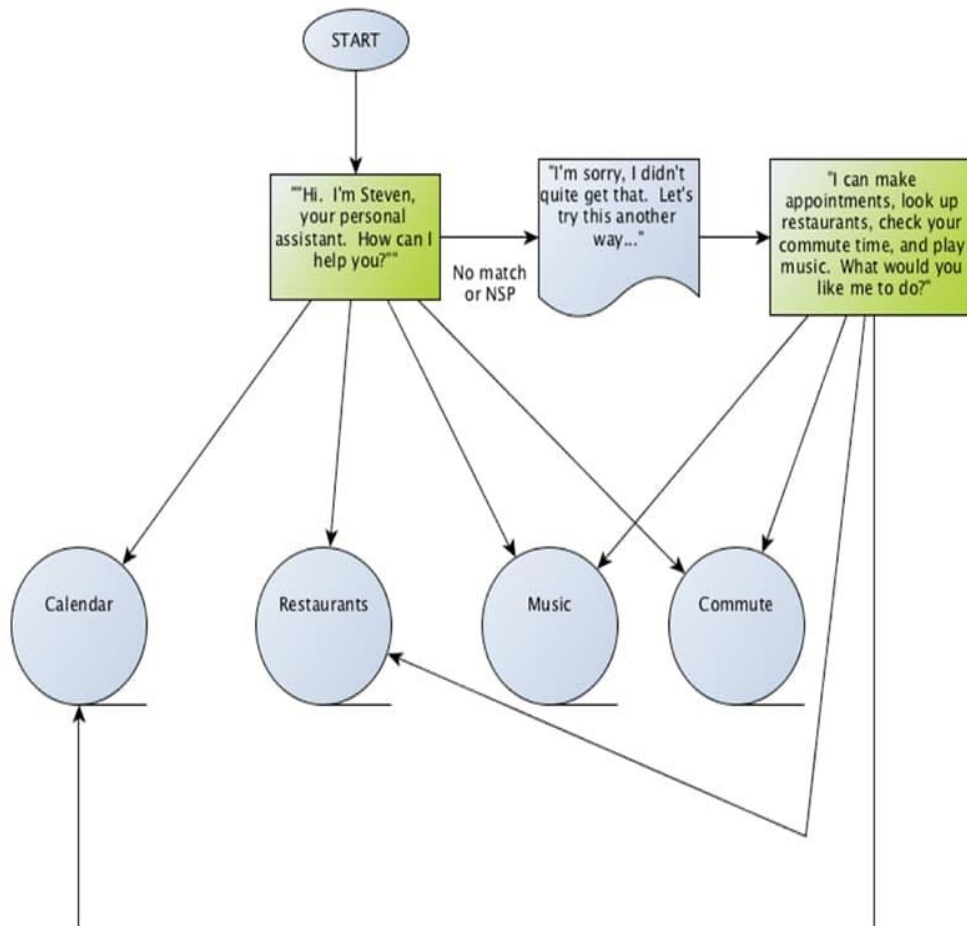
- **For developer end requirements-**
- As python is supported in all operating system, preferred OS is windows 7.
- Python 3.0 is preferred because of its modules
- **Note- Some modules might be different in python 3.5, python 3.6.**
- Basic modules like pyaudio, gTTS, winspeech, speech recognition, google speech recognition, pygame, pyttsx3 and as per requirement modules can be downloaded by using pip.
- **For User end requirements-**
- In windows operating system
- Windows 7(preferred), Windows 10, user can even use windows XP
- Mac ios.
- Linux system.

3.6 Module Division

- The different types of module used in python are as follows
1. **Pyaudio**- Pyaudio is a port audio; the audio player of python which means python can easily play or record sounds.
 2. **gTTs**.- gTTS stands for google Text To Speech. In python user can install this module. With the help of gTTS python can convert text into speech with the help of google so that the word can be accurate.
 3. **Pytsx3**- Pytsx3 is an offline cross platform text to speech library and supports multiple TTS engines.
 4. **Google speech recognizer**- It is speech recognizer which understands your speech and converts into text.
 5. **Winspeech**- It is a speech recognizer library for windows operating system.
 6. **Microphone**- This is use with pyaudio to listen the speech with default microphone of computer or laptop if pyaudio is not import then microphone will be undefined.
 7. **Pygame**- It is cross platform set of modules for writing video games.

4.SYSTEM DESIGN

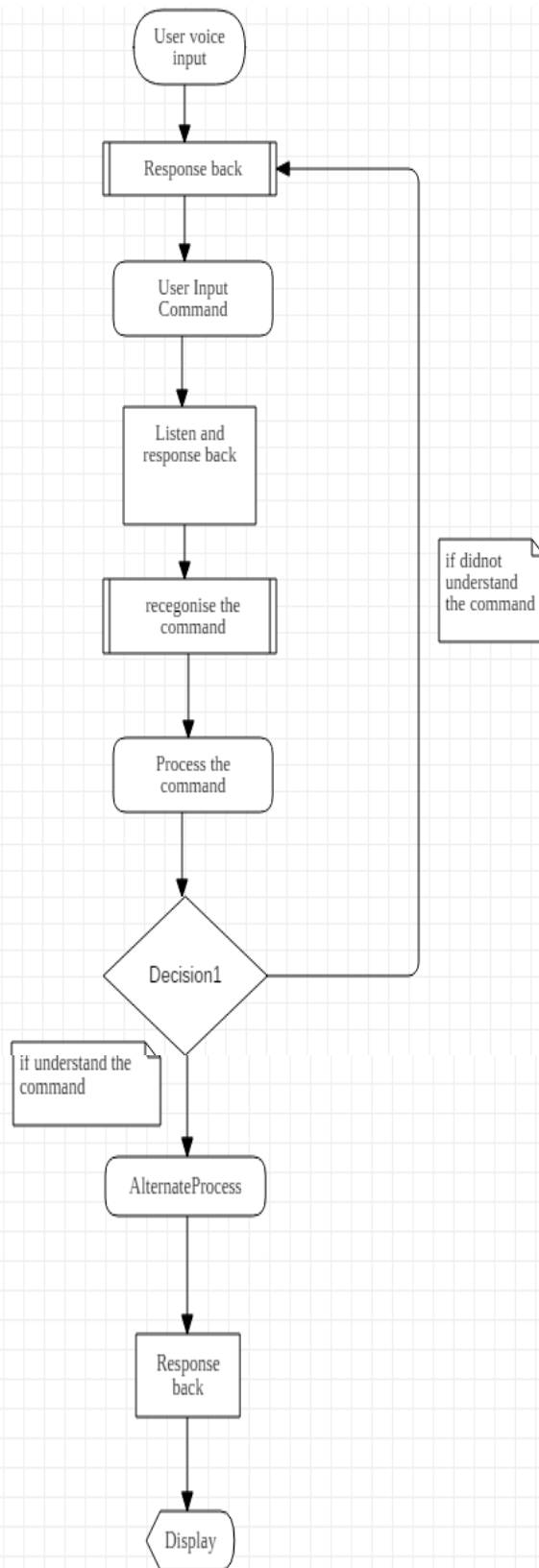
4.1 BASIC IDEA



Description

- This diagram gives the basic idea of the working of voice assistant.
- If voice message is recognized by the voice assistant it will say “HI” and then the user is ready to give command as he desires.
- If voice message did not get recognized then voice assistant will display sorry message and also if the voice assistant is not capable of doing a task then it will also display a sorry message.

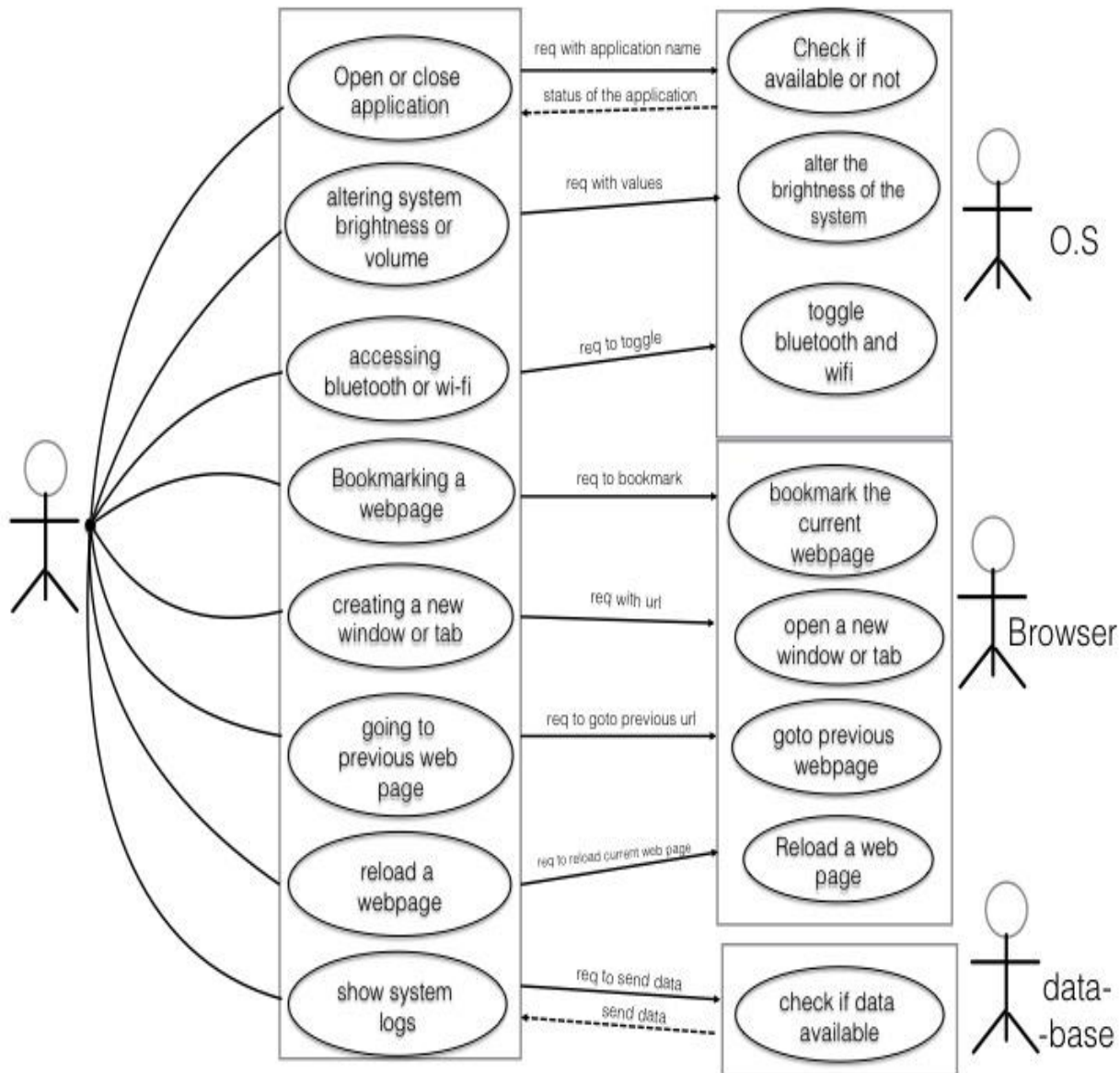
4.2 FLOWCHART DIAGRAM



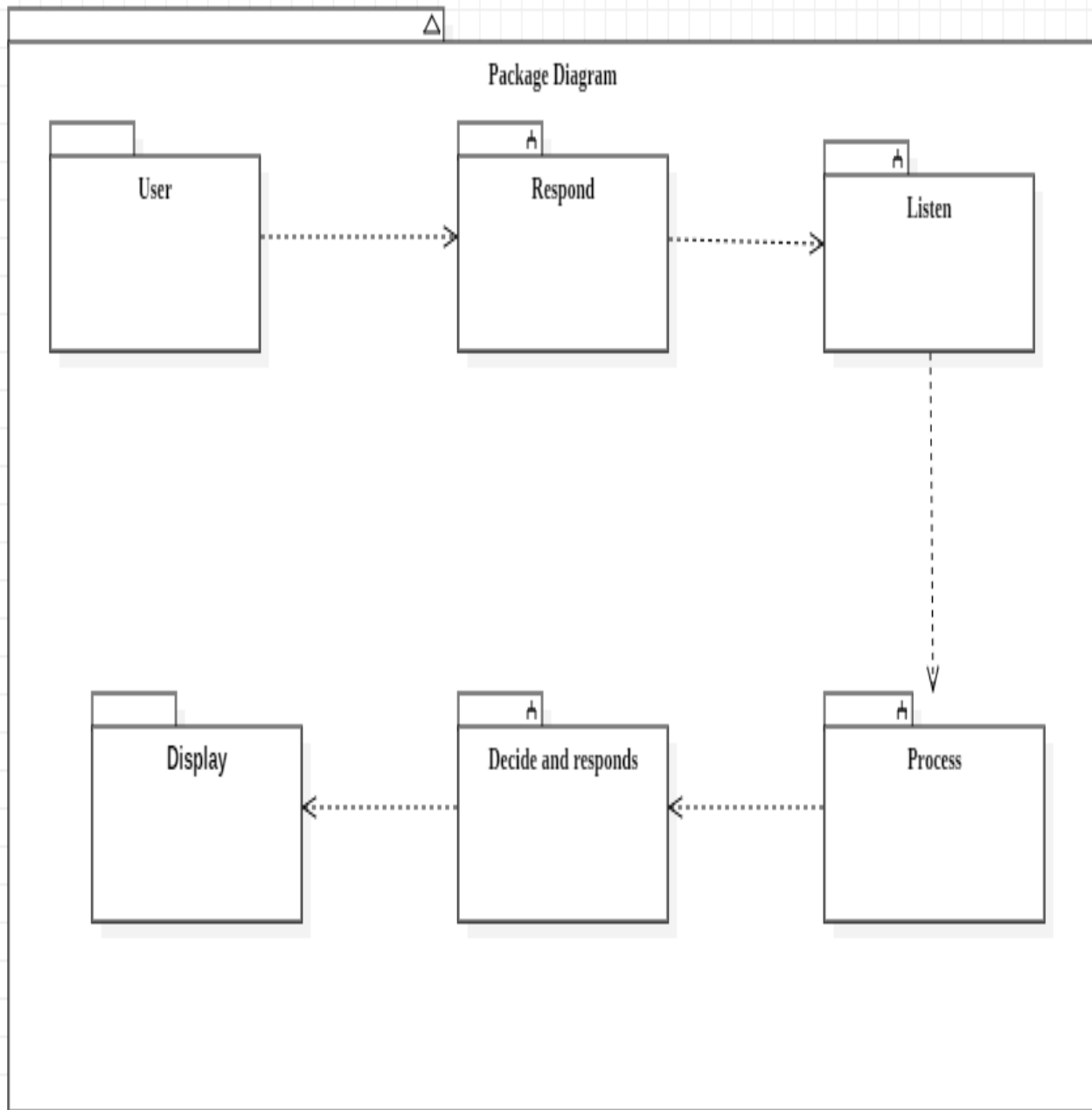
Flowchart Description

- AI is on Standby mode
- User wake up the AI by voice input.
- User gives the command to the application.
- AI response back to the user and listen to commands.
- AI then process the commands and then understands
- After understands it responds back and performs the task.
- After performing the task AI waits for next command.

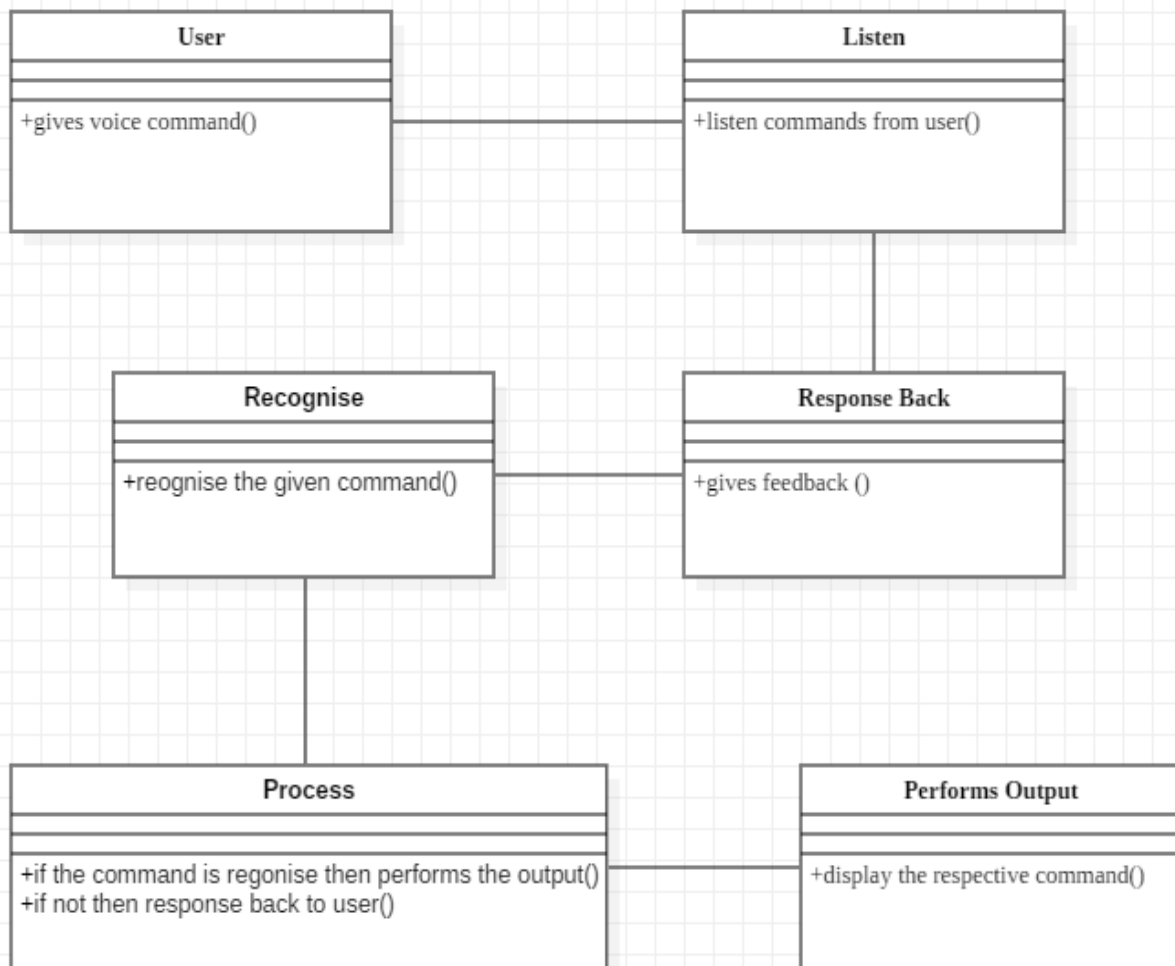
4.4 USE CASE DIAGRAM



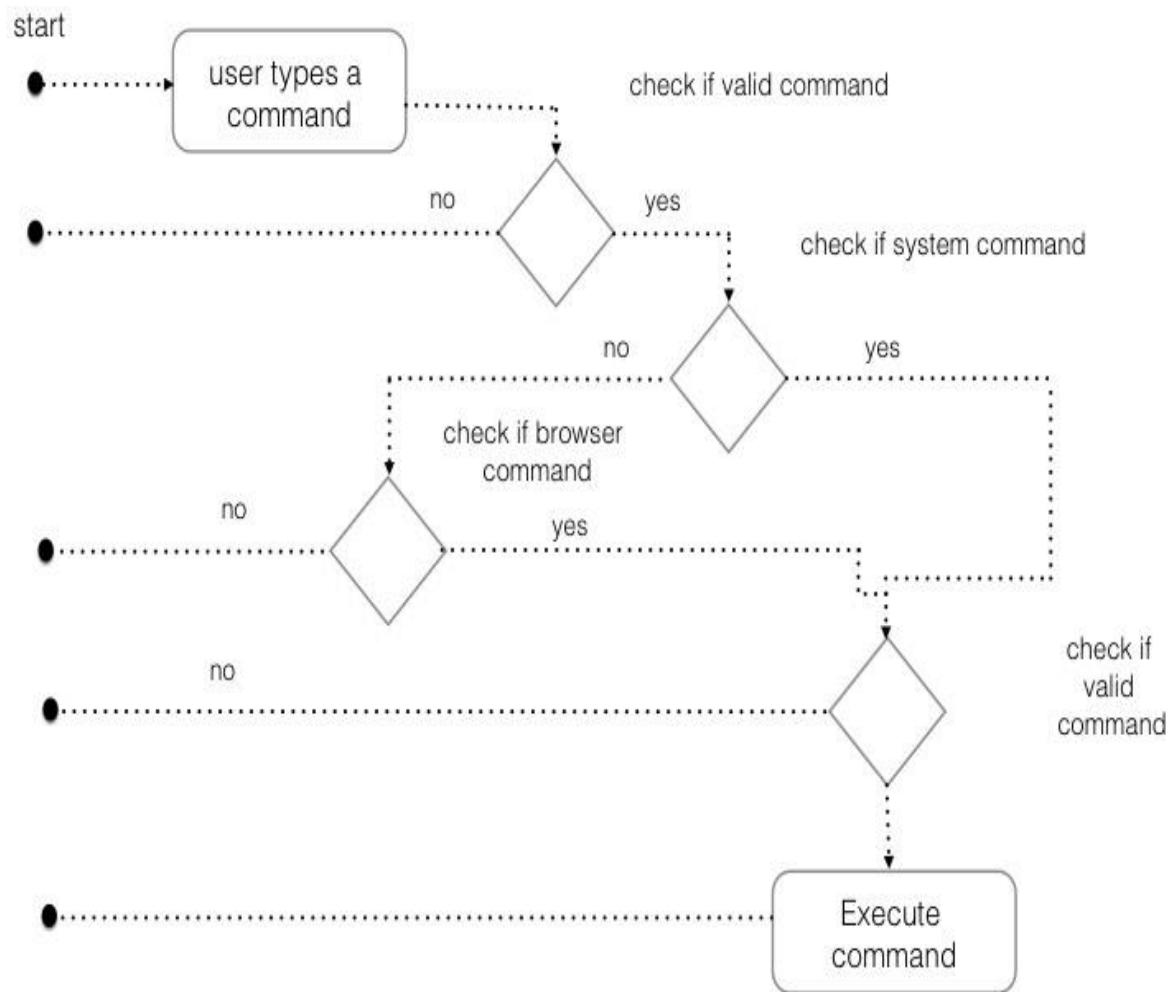
4.5 PACKAGE DIAGRAM



4.6 CLASS DIAGRAM



4.7 ACTIVITY DIAGRAM



5.IMPLEMENTATION APPROCHES

Once the source code is developed, next stage is Implementation of the code as well as testing of code and hardware device (if any).

The main aim of Implementation and Testing is to debug the errors and test the code for satisfaction output.

Once the code passes the Implementation and Testing phase then it is sent to another phase.

5.1.CODING DETAILS AND CODE EFFICIENCY

5.1.1 CODE EFFICIENCY

```
import pyttsx3
import datetime
import speech_recognition as sr
import wikipedia
import webbrowser
import os
import random
import math
from translate import Translator
import pyaudio
from countryinfo import *
import os.path
import subprocess
import smtplib
import pygame

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

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

```

def wishMe():

    speak('starting all system applications')
    speak('installing all drivers')
    speak('every driver is installed')
    speak('all systems have been started')
    speak('now i am online sir')

    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 MONTY")
    speak("Sir Please tell me how may I help you")

def takeCommand():
    #It takes microphone input from the user and returns string output

    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:
        speak("Say that again please ...")
        print("Say that again please ...")
        return "None"
    return query

def note(text):
    date = datetime.datetime.now()

```

```
file_name = str(date).replace(":", "-") + "-note.txt"
with open(file_name, "w") as f:
    f.write(text)
subprocess.Popen(["notepad.exe", file_name])
```

```
def media():
    try:
        speak('ok sir')
        speak('starting required application')
        speak('what do you want me to play for you')
        k = takeCommand()
        os.startfile('M:\\SONGS\\Playlists\\' + k + '.mp3')
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
def shutdown():
    speak('understood sir')
    speak('connecting to command prompt')
    speak('shutting down your computer')
    os.system('shutdown -s')
```

```
def gooffline():
    speak('ok sir')
    speak('closing all systems')
    speak('disconnecting to servers')
    speak('going offline')
    speak('have a good day sir')
    quit()
```

```
def factorial(num):
    if num == 1:
        return num
    else:
        return num * factorial(num-1)
```

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

```
#Logic for executing task based on query
```

```
if 'wikipedia' in query:  
    speak('Searching Wikipedia ...')  
    query = query.replace("wikipedia","")  
    results = wikipedia.summary(query,sentences=2)  
    speak("According to Wikipedia")  
    print(results)  
    speak(results)
```

```
elif 'open youtube' in query:  
    speak("Okay sir")  
    webbrowser.open("https://www.youtube.com")
```

```
elif 'open google' in query:  
    speak("Here you go sir")  
    webbrowser.open("https://www.google.com")
```

```
elif 'current affairs' in query:  
    webbrowser.open("https://www.jagranjosh.com/current-affairs")
```

```
elif 'weather forecast' in query:  
    speak("here you go sir")  
    webbrowser.open("https://www.accuweather.com/en/world-weather")
```

```
elif 'open map' in query:  
    speak("ok sir")  
    webbrowser.open("https://www.google.com/maps/@19.0352007,72.9578672,15z")
```

```
elif 'stack overflow' in query:  
    webbrowser.open("stackoverflow.com")
```

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

```
elif 'date' in query:  
    strdate = datetime.datetime.now().strftime("%D:%Y")  
    speak (f"Sir,the date is {strdate}")
```

```
elif 'open visual studio' in query:  
    codepath = "C:\\Users\\Monty\\AppData\\Local\\Programs\\Microsoft VS Code\\Code.exe"  
    os.startfile(codepath)
```

```
elif '.com' in query:  
    speak("here you go sir")  
    webbrowser.open("www."+query)
```

```
elif '.in' in query:  
    speak("here you go sir")  
    webbrowser.open("www."+query)
```

```
elif '.net' in query:  
    speak("here you go sir")  
    webbrowser.open("www."+query)
```

```
elif "creed" in query:  
    codepath = "F:\\Assassin's Creed - Syndicate\\ACS.exe"  
    os.startfile(codepath)
```

```
elif 'name' in query:  
    speak(f"my name is Multimedia Operating Nexus with Thunderbolt Yabber")  
    print("my name is Multimedia Operating Nexus with Thunderbolt Yabber")  
    speak("but in short everyone calls me MONTY")  
    print("but in short everyone calls me MONTY")
```

```
elif 'created you' in query:  
    speak(f"my creator is Mayuresh Shridhar Koli")
```

```
elif 'bye' in query:  
    speak(f"Good Bye sir have a good day")
```

```
elif 'hello' in query or 'hey' in query or 'hi' in query:  
    stMsgs = ['hello sir', 'hey sir', 'hi sir']  
    print(random.choice(stMsgs))  
    speak(random.choice(stMsgs))
```

```
elif 'play music' in query:  
    media()
```

```
elif 'down' in query:  
    gooffline()
```

```
elif 'offline' in query:
```

```
gooffline()
```

```
elif 'shutdown' in query:  
    shutdown()
```

```
elif 'open my computer' in query:  
    speak("ok sir")  
    os.system('explorer C:\\\\"{}"'.format(query.replace('open', '')))  
    continue
```

```
elif 'thank you' in query:  
    speak("your welcome sir")
```

```
elif 'how are you' in query:  
    speak("i am fine")  
    speak("thank you sir")  
    speak("and how are you sir")
```

```
elif 'i am fine' in query:  
    speak("good")  
    speak("ok sir now tell me how may i help you")
```

```
elif 'add' in query:  
    try:  
        speak("what will be your first number")  
        a = float(takeCommand())  
        speak("what will be your second number")  
        b = float(takeCommand())  
        speak("do you want me to add more number")  
        if 'no' in takeCommand():  
            ab = a+b  
            print(ab)  
            speak("the answer will be")  
            speak(ab)  
        else:  
            speak("what will be your another number")  
            c = float(takeCommand())  
            abc = a+b+c  
            print(abc)  
            speak("the answer will be")  
            speak(abc)  
    except:  
        print("something went wrong")  
        speak("something went wrong")
```



```

elif 'subtract' in query:
    try:
        speak("what will be your first number")
        a = float(takeCommand())
        speak("what will be your second number")
        b = float(takeCommand())
        speak("do you want to subtract more number")
        if 'no' in takeCommand():
            ab = a-b
            print(ab)
            speak("the answer will be")
            speak(ab)
        else:
            speak("what will be your another number")
            c = float(takeCommand())
            abc = a-b-c
            print(abc)
            speak("the answer will be")
            speak(abc)
    except:
        print("something went wrong")
        speak("something went wrong")

```

```

elif 'multiply' in query:
    try:
        speak("what will be your first number")
        a = float(takeCommand())
        speak("what will be your second number")
        b = float(takeCommand())
        speak("do you want me to multiply with more number")
        if 'no' in takeCommand():
            ab = a*b
            print(ab)
            speak("the answer will be")
            speak(ab)
        else:
            speak("what will be your another number")
            c = float(takeCommand())
            abc = a*b*c
            print(abc)
            speak("the answer will be")
            speak(abc)
    except:

```

```
print("something went wrong")
speak("something went wrong")
```

```
elif 'divide' in query:
    try:
        speak("what will be your first number")
        a = float(takeCommand())
        speak("what will be your second number")
        b = float(takeCommand())
        ab = a/b
        print(ab)
        speak("the answer will be")
        speak(ab)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'cube' in query:
    try:
        speak("please tell me a number of which do you want me to find a cube")
        cu = int(takeCommand())
        cube = cu**3
        print(cube)
        speak(cube)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'square' in query:
    try:
        speak("please tell me a number of which do you want me to find a square")
        sq = int(takeCommand())
        square = sq**2
        print(square)
        speak(square)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'route' in query:
    try:
        speak("please tell me a number of which do you want me to find a square root")
```

```

        x = float(takeCommand())
        z = math.sqrt(x)
        print(z)
        speak(z)
    except:
        print("something went wrong")
        speak("something went wrong")

elif 'binary' in query:
    try:
        speak("please tell me a number which do you want me to convert in binary")
        x = int(takeCommand())
        z = bin(x)
        print(z)
        speak(z)
    except:
        print("something went wrong")
        speak("something went wrong")

```

```

elif 'octal' in query:
    try:
        speak("please tell me a number which do you want me to convert in octal")
        x = int(takeCommand())
        z = oct(x)
        print(z)
        speak(z)
    except:
        print("something went wrong")
        speak("something went wrong")

elif 'hexadecimal' in query:
    try:
        speak("do you want me to convert integer number or float number")
        takeCommand()
        if 'integer' in query:
            x = int(takeCommand())
            z = hex(x)
            print(z)
            speak(z)
        else:
            x = float(takeCommand())
            z = float.hex(x)
            print(z)
    except:
        print("something went wrong")
        speak("something went wrong")

```

```
        speak(z)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'bitwise and' in query:
    try:
        speak("what will be your first number")
        a = int(takeCommand())
        speak("what will be your second number")
        b = int(takeCommand())
        c = a&b
        print(c)
        speak(c)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'bitwise or' in query:
    try:
        speak("what will be your first number")
        a = int(takeCommand())
        speak("what will be your second number")
        b = int(takeCommand())
        c = a|b
        print(c)
        speak(c)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'bitwise xor' in query:
    try:
        speak("what will be your first number")
        a = int(takeCommand())
        speak("what will be your second number")
        b = int(takeCommand())
        c = a^b
        print(c)
        speak(c)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```

elif 'perfect' in query:
    try:
        speak("what will be your first number")
        a = int(takeCommand())
        speak("what will be your second number")
        b = int(takeCommand())
        ab = range(a,b)
        for i in ab:
            if i**2<=b:
                print(i**2)
                speak(i**2)
    except:
        print("something went wrong")
        speak("something went wrong")

elif 'prime number' in query:
    try:
        speak("which number should i take")
        x = int(takeCommand())
        for i in range (2,x):
            if x%i==0:
                print("it is not a prime number")
                speak("it is not a prime number")
                break
            else:
                print("it is a prime number")
                speak("it is a prime number")
                break
    except:
        print("something went wrong")
        speak("something went wrong")

```

```

elif 'factorial' in query:
    try:
        speak("which number should i take")
        num = int(takeCommand())
        if num<0:
            print("factorial cannot be found for negative number")
            speak("factorial cannot be found for negative number")
        elif num==0:
            print("factorial of 0 is 1")
            speak("factorial of 0 is 1")
        else:

```

```
        print(factorial(num))
        speak(factorial(num))
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'german translator' in query:
    try:
        speak("what do you want me to translate sir")
        ge = str(takeCommand())
        translator = Translator(to_lang="german")
        translation = translator.translate(ge)
        print(translation)
        speak(translation)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'spanish translator' in query:
    try:
        speak("what do you want me to translate sir")
        sp = str(takeCommand())
        translator = Translator(to_lang="spanish")
        translation = translator.translate(sp)
        print(translation)
        speak(translation)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'capital' in query:
    try:
        speak("which country capital do you want to know about")
        country = CountryInfo(takeCommand())
        a = country.capital()
        speak("the capital of this country is")
        print(a)
        speak(a)
    except:
```

```
print("something went wrong")
speak("something went wrong")
```

```
elif 'currency' in query:
    try:
        speak("which country currency do you want to know about")
        country = CountryInfo(takeCommand())
        a = country.currencies()
        speak("the currency used in this country is")
        print(a)
        speak(a)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'time zone' in query:
    try:
        speak("which country timezones do you want to know about")
        country = CountryInfo(takeCommand())
        a = country.timezones()
        speak("the timezone of this country is")
        print(a)
        speak(a)
    except:
        print("something went wrong")
        speak("something went wrong")
```

```
elif 'make a note' in query:
    speak("what would you like me to write down?")
    note_text = takeCommand()
    note(note_text)
    speak("I've made a note of that")
```

```
elif 'email' in query:
    try:
        print('Who is the recipient? ')
        speak('Who is the recipient? ')
        recipient = takeCommand()
        recipient = recipient.lower()
```

```
        if 'mayuresh' in recipient:
            print('What should I say? ')
            speak('What should I say? ')
            content = takeCommand()
            server = smtplib.SMTP('smtp.gmail.com', 587)
```

```
server.ehlo()
server.starttls()
server.login("email id", 'password')
server.sendmail('email id', "recipient:email id", content)
server.close()
print('Email sent!')
speak('Email sent!')
```

```
elif 'durgesh' in recipient:
    print('What should I say? ')
    speak('What should I say? ')
    content = takeCommand()
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login("email id", 'password')
    server.sendmail('email id', "recipient:email id", content)
    server.close()
    print('Email sent!')
    speak('Email sent!')
```

```
elif 'kp' in recipient:
    print('What should I say? ')
    speak('What should I say? ')
    content = takeCommand()
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login("email id", 'password')
    server.sendmail('email id', "recipient:email id", content)
    server.close()
    print('Email sent!')
    speak('Email sent!')
```

```
elif 'pawan' in recipient:
    print('What should I say? ')
    speak('What should I say? ')
    content = takeCommand()
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login("email id", 'password')
    server.sendmail('email id', "recipient:email id", content)
    server.close()
    print('Email sent!')
    speak('Email sent!')
```



```

elif 'harsh' in recipient:
    print('What should I say? ')
    speak('What should I say? ')
    content = takeCommand()
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login("email id", 'password')
    server.sendmail('email id', "recipient:email id", content)
    server.close()
    print('Email sent!')
    speak('Email sent!')

```

```

else:
    print('Sorry ' + 'Sir' + '!', I am unable to send your message at this moment!')
    speak('Sorry ' + 'Sir' + '!', I am unable to send your message at this moment!')

```

```

except:
    print('Sorry ' + 'Sir' + '!', I am unable to send your message at this moment!')
    speak('Sorry ' + 'Sir' + '!', I am unable to send your message at this moment!')

else:
    try:
        try:
            res = client.query(query)
            results = next(res.results).text
            print(results)
            speak(results)
        except:
            results = wikipedia.summary(query, sentences=2)
            print(results)
            speak(results)
    except:
        print("sorry")

```

```
input()
```

5.2.TESTING APPROACH

5.2.1.UNIT TESTING

The Unit Testing is of two types

- a) Software Testing
- b) Hardware Testing.

- A) Software Testing:** Software Testing is done by set of source codes or computer programming modules. The Tester test the module to determine whether they are fit to use or not.
- B) Hardware Testing:** Hardware Testing can be performed on software as well as on physical hardware products to verify whether they perform as expected and desired output.

5.2.2.INTEGRATED TESTING

integrated testing is done to check whether the system is running as expected or as documented. If the system pass the system testing then it will sent to next stage and if it fail then it is resent to integrated testing phase

Each and every module is tested in a group from pyaudio to pygame and other modules like os,wikipedia,etc. Modules are as follow

- **Pyaudio**
- **Pygame**
- **Pyttsx3**
- **Speech recognition**

And so on...

- **PYAUDIO:** The tester will test the audio output by using pyaudio module. If the audio is clear and understandable then Tester will set the audio according to the customer requirement or the given protocol.
- **PYGAME:** The tester will test this cross platform set of python to check the running module and check desire output graphics.
- **PYTTSX3:** This is cross platform text to speech module. The tester will test module by simply adding text and then listing further.
- **Speech recognition:**The tester will test the speech by simply gibing voice command to the module.

5.2.3.BETA TESTING

In software development, a **beta test** is the second phase of software **testing** in which a sampling of the intended audience tries the product out. **Beta** is the second letter of the Greek alphabet. Originally, the term alpha **test** meant the first phase of **testing** in a software development process.

In this testing I tried to target some of my friends for testing and given them 2 days to test it properly and told them to give me a report of bugs.

After that period of testing the outcome has come very positive but there were some problem with the code not major problems but some minor problems.

After the report of the testers I tried to fix them and now it is ready to use.

PROBLEM REPORT OF BETA TESTING:-

1. In calculation when the program don't understand the number we say then it gets stop functioning.
2. When we say make a note but after making a note the user don't get the note file.
3. The program do not open the website when we say .in and .net at the last of the website.
4. Why the user needs to always say wikipedia to search.

5.3.MODIFICATION AND IMPROVEMENTS

Modification and improvements are done after receiving the beta testing reports.

In report there are 4 problems.

1. Solution of the 1st problem was inside the calculations code and the problem got fixed when I used the try: and except: block.
2. Solution of the 2nd problem was to make that program file executable python file. After this the note is stored on the desktop of the computer.
3. Solution of the 3rd problem was to add the .in and .net query in the code like the .com query after this it will access .net and .in websites.
4. Solution of the 4th problem is that I make sure to include the wikipedia program at the end of the program.

This are the modifications and improvements are done after beta testing.

5.4.TEST CASES

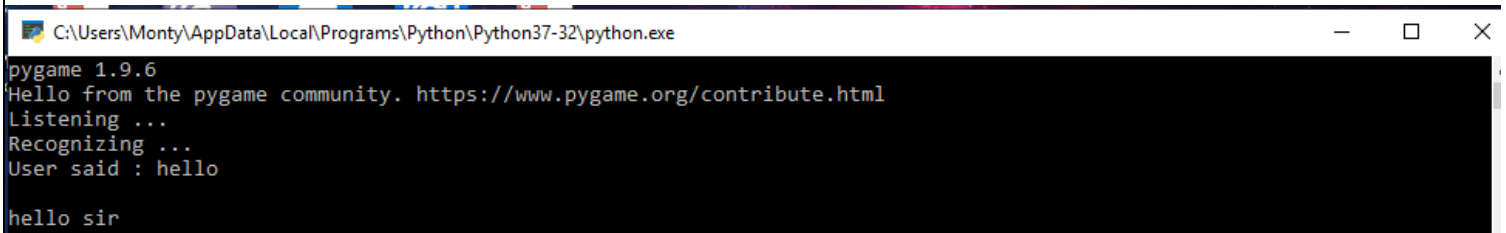
Sr.No	Test case id	Test case name	Test case desc	Step	Expected result	Actual Result	Test case status pass/fail
1	Pyaudio	Validate audio	To play output audio	Voice modulation	Playing desire output audio.	Inserted successful	Pass
2	Pygame	Validate output graphics	To play output graphics	Graphics designing	Display graphics.	Inserted successful	Pass
3	pytsx	Validate Text to speech conversion	To convert text to audio file	Text is added and converted into audio file	Text is converted into audio file.	Inserted successful	Pass
4	Speech recognition	Recognise the input speech	To verify and recognise the input speech	Users gives voice as input and it is recognise by the module	Recognise the voice of user.	Inserted successful	Pass

6.RESULTS AND DISCUSSION

6.1.TEST REPORTS

Modules:

1. Pyaudio

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Users\Monty\AppData\Local\Programs\Python\Python37-32\python.exe'. The terminal output is as follows:

```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Listening ...
Recognizing ...
User said : hello
hello sir
```

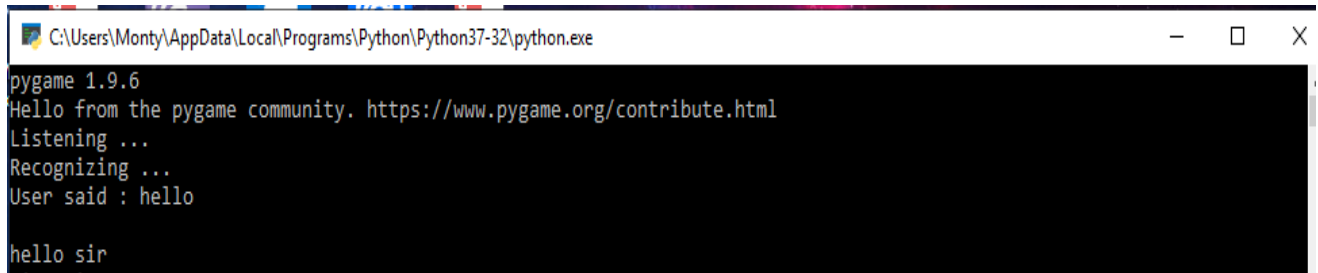
PyAudio is use as an audio port/cross platform audio port for python. With the help of pyaudio, you can record as well as play audio file on the platform.

In the above screen shot, it is displayed what the output audio is played.

The user will give input audio command and with the help of pyaudio module, the output will be given by voice and with print function it will print on the output screen.

Note:

- The audio should be clear and loudly if you are recording or giving command.
- Use external mic for better input.

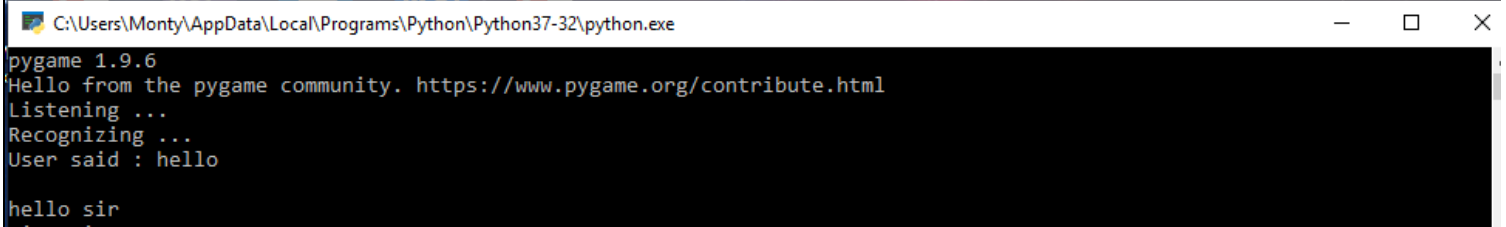
A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Users\Monty\AppData\Local\Programs\Python\Python37-32\python.exe'. The terminal output is as follows:

```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Listening ...
Recognizing ...
User said : hello

hello sir
```

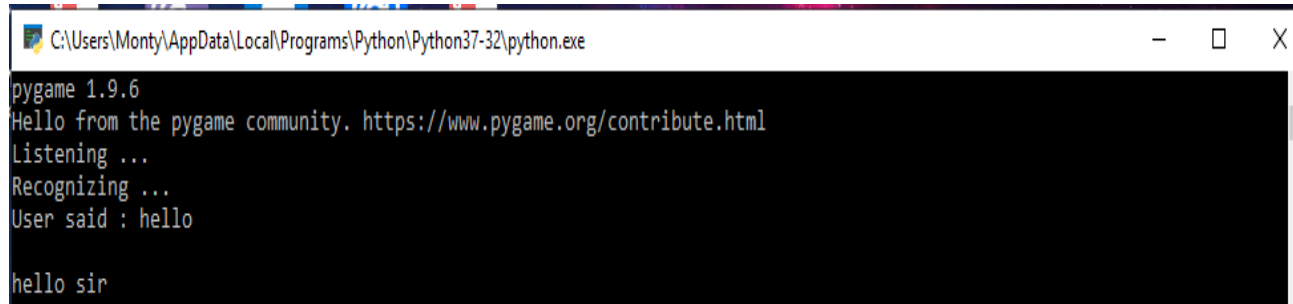
- In the above screen shot, the user is interacting with the system and the system in response gives the feedback to the user.
- In this way two way communications is on between the user and system with the help of pyaudio.
- “pip install PyAudio” can be used to install.

2. gTTS.

A screenshot of a Windows command prompt window titled "C:\Users\Monty\AppData\Local\Programs\Python\Python37-32\python.exe". The terminal shows the following text: "pygame 1.9.6", "Hello from the pygame community. https://www.pygame.org/contribute.html", "Listening ...", "Recognizing ...", "User said : hello", and "hello sir".

```
C:\Users\Monty\AppData\Local\Programs\Python\Python37-32\python.exe
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Listening ...
Recognizing ...
User said : hello
hello sir
```

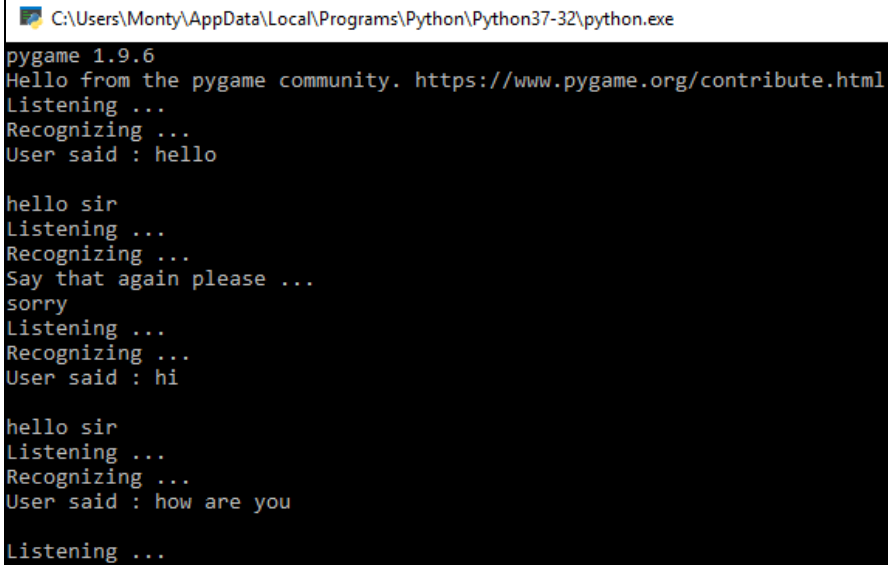
- gTTS is known as google text to speech.
- This module helps the program to convert text to speech with the help of google.
- With the help of google the conversion of text to speech will be highly accurate.
- Python library and CLI tool to interface with Google Translator's text-to-speech API.



```
C:\Users\Monty\AppData\Local\Programs\Python\Python37-32\python.exe
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Listening ...
Recognizing ...
User said : hello
hello sir
```

- In above screen shot we can see that the language is evaluating the words of the user and trying to convert the words help of gtts API.
- This API needs internet connection.
- As gTTS API is google API which requires internet connections.
- By using “pip install gTTS” you can install.

3. speech_recognition.



A screenshot of a Windows command prompt window titled "C:\Users\Monty\AppData\Local\Programs\Python\Python37-32\python.exe". The terminal displays the output of a Python script using the speech_recognition library. The text is as follows:

```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Listening ...
Recognizing ...
User said : hello

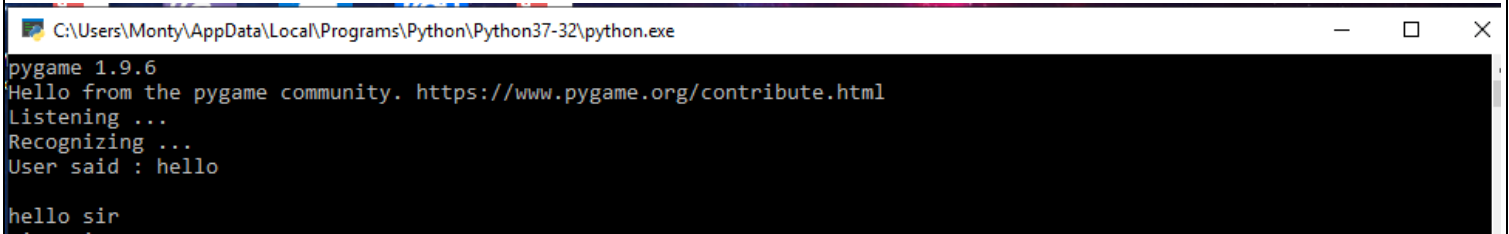
hello sir
Listening ...
Recognizing ...
Say that again please ...
sorry
Listening ...
Recognizing ...
User said : hi

hello sir
Listening ...
Recognizing ...
User said : how are you

Listening ...
```

- Speech recognition is basically Recognition of spoken words by the users.
- It recognizes the words, phase, and sentences and so on.
- Speech recognition is the medium between Humans and Computer Machine.
- Computer/python can understand the Humans language with the help of speech_recognition.
- With the help of speech recognition you can give any command to the program.
- By using “pip install SpeechRecognition==1.2.3” you can install.

4. Microphone:

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Users\Monty\AppData\Local\Programs\Python\Python37-32\python.exe'. The terminal output is as follows:

```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Listening ...
Recognizing ...
User said : hello

hello sir
```

- Basically this module is use for machine to listen what users gives voice input.
- This act as an ear for the program.
- By using “pip install SpeechRecognition==1.2.3” you can install.

5. Pocketsphinx:

```
import pocketsphinx  
import pyaudio  
  
engine = pyttsx.init()
```

- Pocketsphinx is a pocket dictionary for python program.
- This is an offline dictionary which can be used as an alternative option for gtts.
- It contains almost everything which is required to understand the human speech text words or sentences.
- By using “pip install pocketsphinx” you can install pocket sphinx

6.2.USER DOCUMENTATION

Make sure you are using a desktop or laptop which meets the following hardware and software requirements (**refer page no.25 and 26**) and make sure you have python install in your computer.
Trigger words to use this products:-

1. The time:-speaks the time.
2. Date:-speaks the date.
3. Current affairs:-directs user to the website.
4. Weather forecast:-directs user to the website.
5. Open map:-directs user to the website.
6. Websitename.com:-directs user to the following website.
7. Websitename.in:-directs user to the following website.
8. Websitename.net:-directs user to the following website.
9. Open visual code:-opens visual code
10. Play music:-it plays music but make sure that song should be in your computer and the given directory to the program.
11. Open my computer:-opens my computer.
12. Add :-asks which numbers you have to add.
13. Subtract :-asks which numbers you have to subtract.
14. Multiply :-asks which numbers you have to multiply.
15. Divide :-asks which numbers you have to divide.
16. Cube :-asks which number you have to find cube.
17. Square :-asks which number you have to find square.
18. Route :-asks which number you have to find square root.
19. Binary:-asks which number you have to find binary.
20. Bitwise or ,bitwise and,bitwise xor:-asks the numbers to find the following.
21. Perfect:-asks the range in which you have to find perfect squares
22. Prime number:-tells it is a prime no. Or not.
23. Factorial:-gives factorial of a following number.
24. German translator or Spanish translator:-gives translation of what user says.
25. Capital ,Currency or time zone:- gives the following of a country.
26. Make a note:- makes a note and store on your desktop.
27. Email:-send email to the following recipient but the recipient email id should be in program.
28. What is or who is (eg:-what is theory of relativity? or who is this?):-gives the information according to wikipedia.
29. Offline:-quits the program.

7.CONCLUSIONS

7.1.CONCLUSIONS

7.1.1.SIGNIFICANCE OF THE **SYSTEM**

This Project is use for daily usage for every person from school going kids to Developers. This application has wide scope in next 5-10yrs. This application is user-friendly that means both Technical as well as not technical person can use. User just needs to give a voice command and that's it. Desktop Assistant will work accordingly. This project is not only limited to Windows OS but can also run on MAC and other OS. Any User with great knowledge on python can customize as per their requirement.

There will be two mode “default mode” for general users and “pro mode” for developers.

Errors might occur as this project is in beta mode.

7.2.LIMITATIONS OF THE SYSTEM

LIMITATIONS:

- ☐ The major limitations of this project are that, it can be used only for desktop computers and laptops.
- ☐ It cannot use in smartphones.

ENHANCEMENT:

- ☐ If the users have the good idea of knowledge of Python language then the user can customise it according to their need and requirement.
- ☐ These modules are compactable in any versions of windows and can develop accordingly.
- ☐ With the help of neural network connection it can be more accurate both in decision making and development.

7.3.FUTURE SCOPE OF THE PROJECT

Currently the AI industry is about to boom, each and everything will be handle by AI. As this project develops in near future more and more functions can be developed such as “game mode” “privacy manager” “Hacker mode” “mining” and so on.

As this project develops, this system will become more smart and efficient both in decision making and power savings.

Neural network systems can be combined together to make this system more intelligent and advance.

All men have limits, they learn what they are and learn not to exceed them.

Similarly this future AI should be in good hands to help mankind.

8.REFERENCES

Content

- 1.<https://pypi.org/project/PyAudio/>
- 2.<https://pypi.org/project/gTTS/>
- 3.https://www.tutorialspoint.com/artificial_intelligence_with_python/artificial_intelligence_with_python_speech_recognition.htm
- 4.<https://pypi.org/project/SpeechRecognition/>
- 5.<https://pypi.org/project/pocketsphinx/>
- 6.<https://pypi.org/project/winspeech/>
- 7.<https://pythonhosted.org/winspeech/>
- 8.<https://www.google.co.in/>
- 9.<https://www.smartsheet.com/voice-assistants-artificial-intelligence>
- 10.https://en.wikipedia.org/wiki/Virtual_assistant
- 11.<https://whatis.techtarget.com/definition/voice-assistant>
- 12.<https://www.youtube.com/watch?v=Lp9Ftuq2sVI>
- 13.<https://www.youtube.com/watch?v=5mu6qlFY3x0>
- 14.<https://www.youtube.com/watch?v=JuygPoTr4X8&t=263s>
- 15.<https://www.youtube.com/watch?v=drFbAc80-cE>
- 16.<https://www.geeksforgeeks.org/>
- 17.<https://codewithharry.com/videos/python-tutorials-for-absolute-beginners-120>
- 18.<https://github.com/Ashutosh-kv/JARVIS/blob/master/assistant.py>
- 19.https://drive.google.com/file/d/10LNKvwfla8VAg0Rhr_aG7DZDoGOU9CRn/view
- 20.<https://github.com/Its-Triggy/Jarvis>