21AIE211

Introduction to computer networks

PROJECT REPORT

B Tech -AIE

Raspberry Pi Alexa: Build your own Amazon Echo

AMRITA VISHWA VIDYAPEETHAM | School of Engineering

TEAM MEMBERS:

HARI PRANAV J M	-	CB.EN.U4AIE20021
PRAVIN RAJ A K	-	CB.EN.U4AIE20054
SABHARISH A L	-	CB.EN.U4AIE20061
SAI SANGAVI C	-	CB.EN.U4AIE20063
SAIVARSHA R	-	CB.EN.U4AIE20064

Index

Abstract:

In this project we have tried to design a voice assistant with few functions like playing music, giving information on wiki search and providing us with the information like day, date, time, weather and temperature.

Introduction:

A voice assistant, also called an intelligent personal assistant or a connected speaker, are new types of products marketed by Apple, Amazon and Google and are based on natural language speech recognition. Automating repeated tasks to a voice-activated personal assistant frees up the human time and resources. Also, it can efficiently perform these mundane tasks with no errors, which often leads to an improvement in customer satisfaction. Today, voice assistants are integrated into many of the devices we use on a daily basis, such as cell phones, computers, and smart speakers. Because of their wide array of integrations, there are several voice assistants who offer a very specific feature set, while some choose to be open ended to help with almost any situation at hand.

Theory:

Socket programming

Socket is an interface between application process and end to end transport protocol. Socket programming is the communication between the client and server applications using sockets.

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection.

Stream sockets: This is the most common type. The two communicating parties first establish a socket connection, after which any data passed through that connection is guaranteed to arrive in the same order in which it was sent
Datagram sockets: Offer connection-less semantics. Either party sends datagrams as needed and waits for the other to respond. Messages can be lost in transmission or received out of order
Libraries that implement sockets for internet protocol use TCP for streams, UDP for datagrams.
Transmission Control Protocol (TCP) is connection-oriented, meaning once a connection has been established, data can be transmitted in two directions. TCP

has built-in systems to check for errors and to guarantee data will be delivered in the order it was sent, making it the perfect protocol for transferring information like still images, data files, and web pages.

User Datagram Protocol (UDP) is a simpler, connectionless Internet protocol wherein error-checking and recovery services are not required. With UDP, there is no overhead for opening a connection, maintaining a connection, or terminating a connection; data is continuously sent to the recipient, whether or not they receive it.

Threading:
Threading is the process of implementing parallelism of processes or functions. By the concept of threading, we achieve it, by creating a separate thread for a specific function and run it parallelly.

Packages used:

1) **Speech recognition**
   a) Speech_recognition: Speech recognition in Python is used to convert the spoken words into text, make a query or give a reply.
   b) Pyaudio and portaudio: PyAudio provides Python bindings for PortAudio, it is a cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple Mac OS X / macOS.
   c) Pyttsx3: pyttsx3 is a text-to-speech conversion library in Python.

2) **Socket**
   a) Socket: The socket module provides various objects, constants, functions and related exceptions for building full-fledged network applications including client and server programs.
   b) Json: JavaScript Object Notation (JSON) is a standardized format commonly used to transfer data as text that can be sent over a network.

3) Playing music

a) Youtube-dl: youtube-dl is a command-line program to download videos from YouTube.com.
b) Pafy: Pafy is a Python library to download YouTube content and retrieve metadata.
c) Python-vlc: This is used to play the audio in python using VLC media player.

4) Other packages:

a) Wikipedia : Wikipedia is a python package which allows us to access data from Wikipedia and extract the required amount of data.
b) Random : Random is a python built in package, which allows us to randomize replies.
c) Datetime : Datetime package allows us to access and process the date and time values.
d) Urllib : Urllib package is the URL handling module for python. It is used to fetch the song URL form youtube.
e) Threading : This package is used for using the concepts of parallelization of processes.
f) Requests : This package can be used to make HTTP requests such as GET, POST, PUT

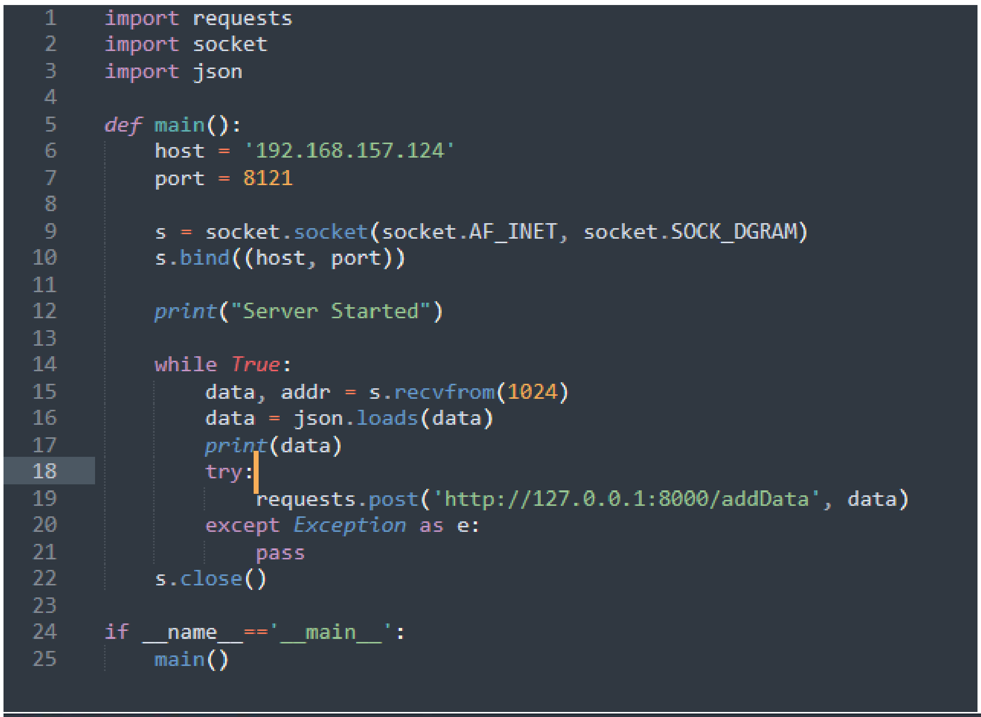
Overview:

We will be performing functions like getting time, date, day, weather, music(audio) and Wikipedia search.
There are 3 modules used here are:

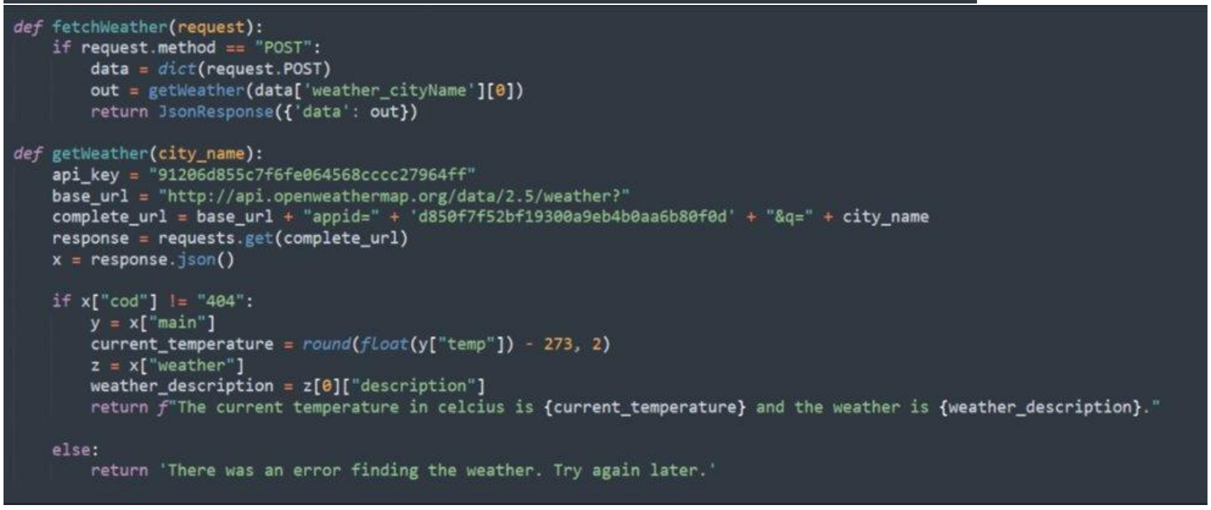
- **smallFunctions.py:** This contains the functions getTime, getDate, getDay, getWiki, getWeather.
- **playMusic.py:** This contains the function required for getting the information of the audio requested by the user to play the music.
- **tara.py:** This is the driver module which incorporates all the other modules and gives the desired output.
- **server.py:** In the server, we will be running this, this collects the log from the raspberry pi and send to our django server for storing and visualizing.

Implementation:

server.py

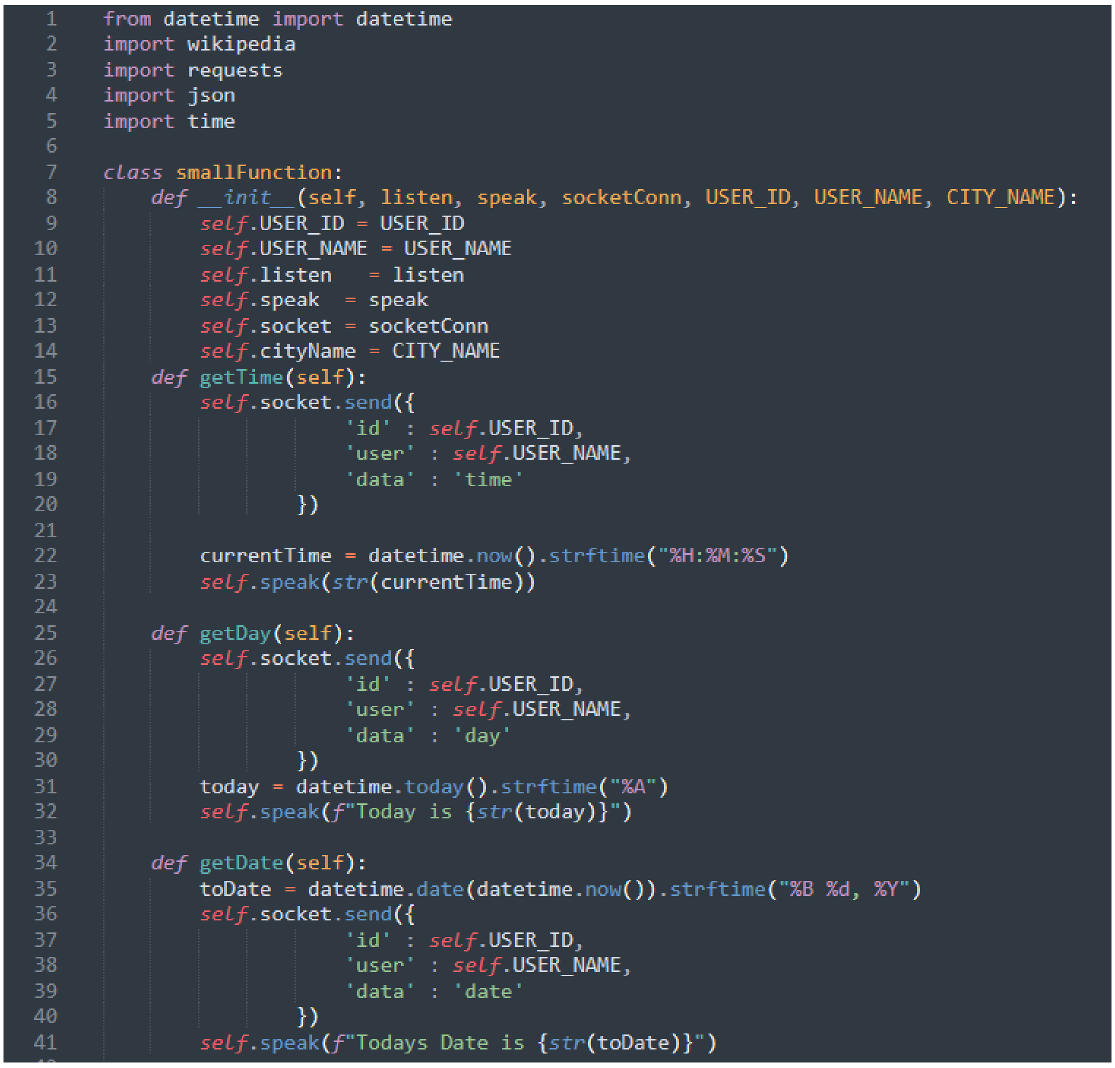


```python
import requests
import socket
import json

def main():
    host = '192.168.157.124'
    port = 8121

    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.bind((host, port))

    print("Server Started")

    while True:
        data, addr = s.recvfrom(1024)
        data = json.loads(data)
        print(data)
        try:
            requests.post('http://127.0.0.1:8000/addData', data)
        except Exception as e:
            pass
    s.close()

if __name__=='__main__':
    main()
```

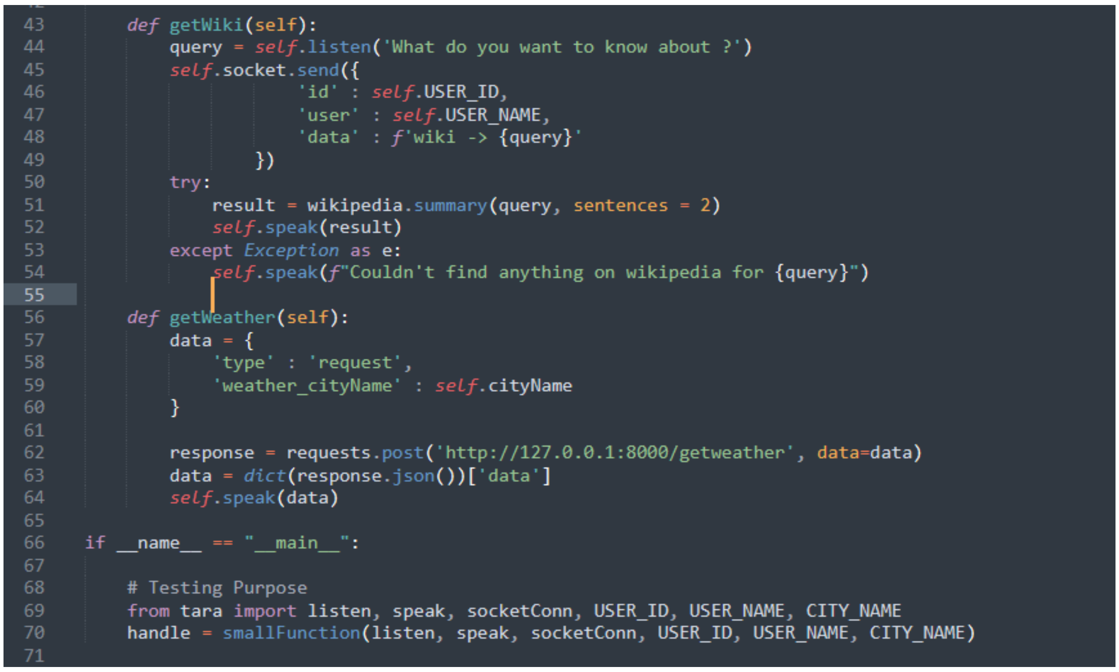


```python
def fetchWeather(request):
    if request.method == "POST":
        data = dict(request.POST)
        out = getWeather(data['weather_cityName'][0])
        return JsonResponse({'data': out})

def getWeather(city_name):
    api_key = "91206d855c7f6fe064568cccc27964ff"
    base_url = "http://api.openweathermap.org/data/2.5/weather?"
    complete_url = base_url + "appid=" + 'd850f7f52bf19300a9eb4b0aa6b80f0d' + "&q=" + city_name
    response = requests.get(complete_url)
    x = response.json()

    if x["cod"] != "404":
        y = x["main"]
        current_temperature = round(float(y["temp"]) - 273, 2)
        z = x["weather"]
        weather_description = z[0]["description"]
        return f"The current temperature in celcius is {current_temperature} and the weather is {weather_description}."

    else:
        return 'There was an error finding the weather. Try again later.'
```

smallFunctions.py

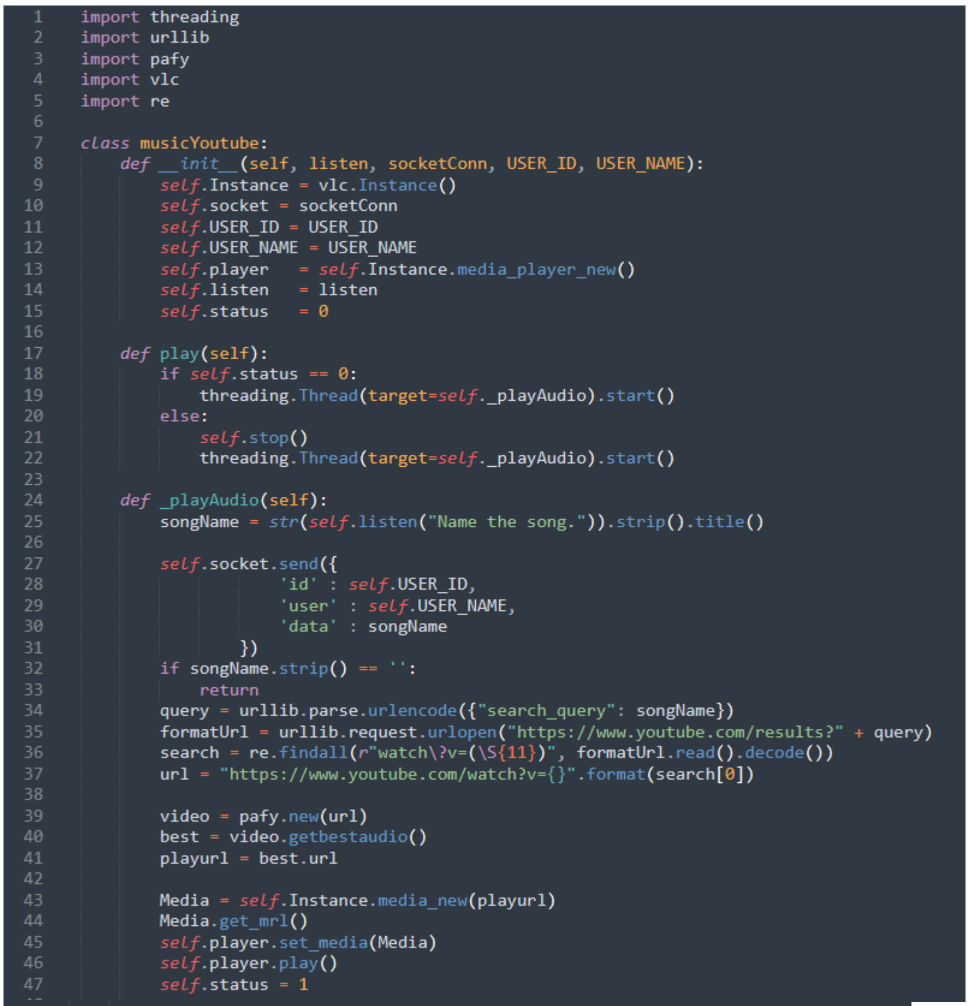


```python
from datetime import datetime
import wikipedia
import requests
import json
import time

class smallFunction:
    def __init__(self, listen, speak, socketConn, USER_ID, USER_NAME, CITY_NAME):
        self.USER_ID = USER_ID
        self.USER_NAME = USER_NAME
        self.listen   = listen
        self.speak  = speak
        self.socket = socketConn
        self.cityName = CITY_NAME
    def getTime(self):
        self.socket.send({
                    'id' : self.USER_ID,
                    'user' : self.USER_NAME,
                    'data' : 'time'
                })

        currentTime = datetime.now().strftime("%H:%M:%S")
        self.speak(str(currentTime))

    def getDay(self):
        self.socket.send({
                    'id' : self.USER_ID,
                    'user' : self.USER_NAME,
                    'data' : 'day'
                })
        today = datetime.today().strftime("%A")
        self.speak(f"Today is {str(today)}")

    def getDate(self):
        toDate = datetime.date(datetime.now()).strftime("%B %d, %Y")
        self.socket.send({
                    'id' : self.USER_ID,
                    'user' : self.USER_NAME,
                    'data' : 'date'
                })
        self.speak(f"Todays Date is {str(toDate)}")
```

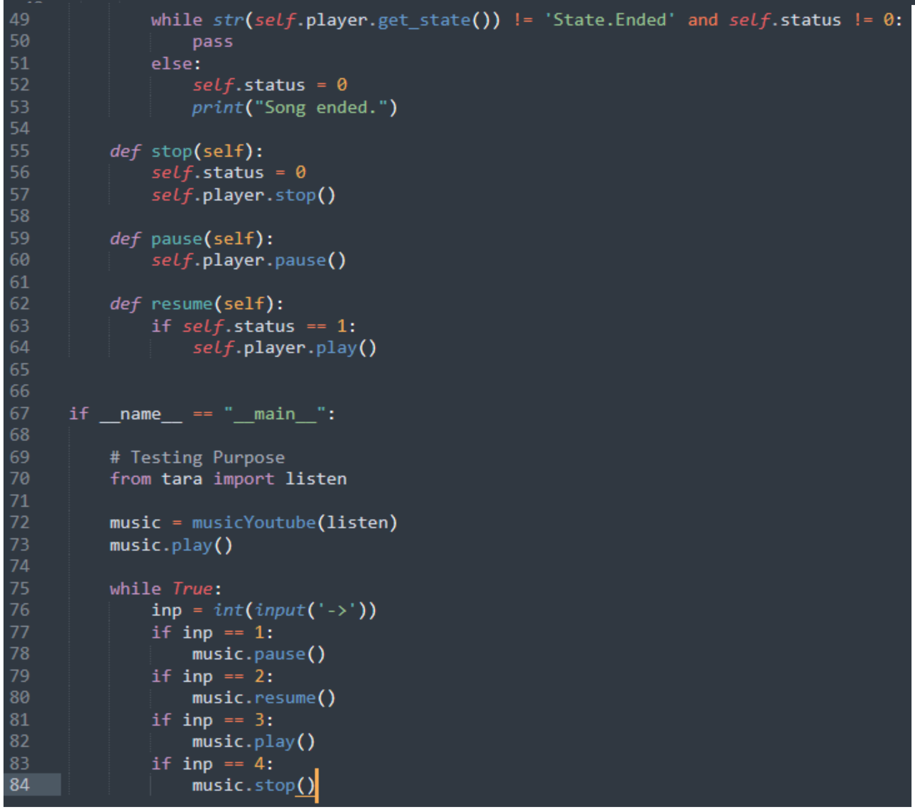


```python
    def getWiki(self):
        query = self.listen('What do you want to know about ?')
        self.socket.send({
                    'id' : self.USER_ID,
                    'user' : self.USER_NAME,
                    'data' : f'wiki -> {query}'
                })
        try:
            result = wikipedia.summary(query, sentences = 2)
            self.speak(result)
        except Exception as e:
            self.speak(f"Couldn't find anything on wikipedia for {query}")

    def getWeather(self):
        data = {
            'type' : 'request',
            'weather_cityName' : self.cityName
        }

        response = requests.post('http://127.0.0.1:8000/getweather', data=data)
        data = dict(response.json())['data']
        self.speak(data)

if __name__ == "__main__":

    # Testing Purpose
    from tara import listen, speak, socketConn, USER_ID, USER_NAME, CITY_NAME
    handle = smallFunction(listen, speak, socketConn, USER_ID, USER_NAME, CITY_NAME)
```

playMusic.py

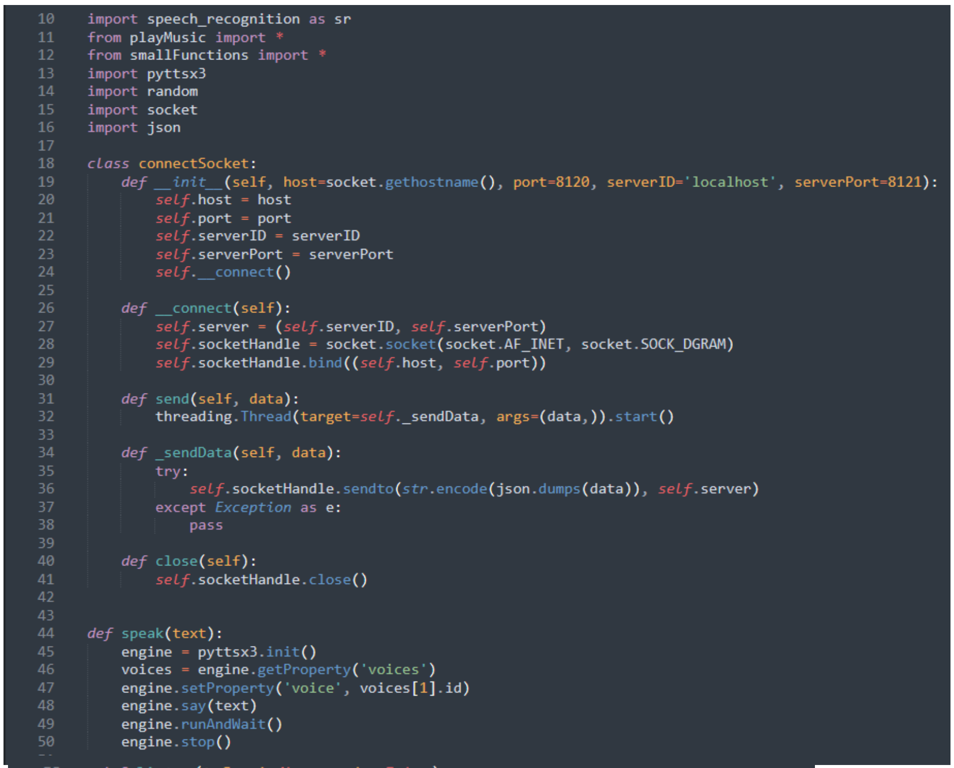


```python
import threading
import urllib
import pafy
import vlc
import re

class musicYoutube:
    def __init__(self, listen, socketConn, USER_ID, USER_NAME):
        self.Instance = vlc.Instance()
        self.socket = socketConn
        self.USER_ID = USER_ID
        self.USER_NAME = USER_NAME
        self.player   = self.Instance.media_player_new()
        self.listen   = listen
        self.status   = 0

    def play(self):
        if self.status == 0:
            threading.Thread(target=self._playAudio).start()
        else:
            self.stop()
            threading.Thread(target=self._playAudio).start()

    def _playAudio(self):
        songName = str(self.listen("Name the song.")).strip().title()

        self.socket.send({
                    'id' : self.USER_ID,
                    'user' : self.USER_NAME,
                    'data' : songName
                })
        if songName.strip() == '':
            return
        query = urllib.parse.urlencode({"search_query": songName})
        formatUrl = urllib.request.urlopen("https://www.youtube.com/results?" + query)
        search = re.findall(r"watch\?v=(\S{11})", formatUrl.read().decode())
        url = "https://www.youtube.com/watch?v={}".format(search[0])

        video = pafy.new(url)
        best = video.getbestaudio()
        playurl = best.url

        Media = self.Instance.media_new(playurl)
        Media.get_mrl()
        self.player.set_media(Media)
        self.player.play()
        self.status = 1
```

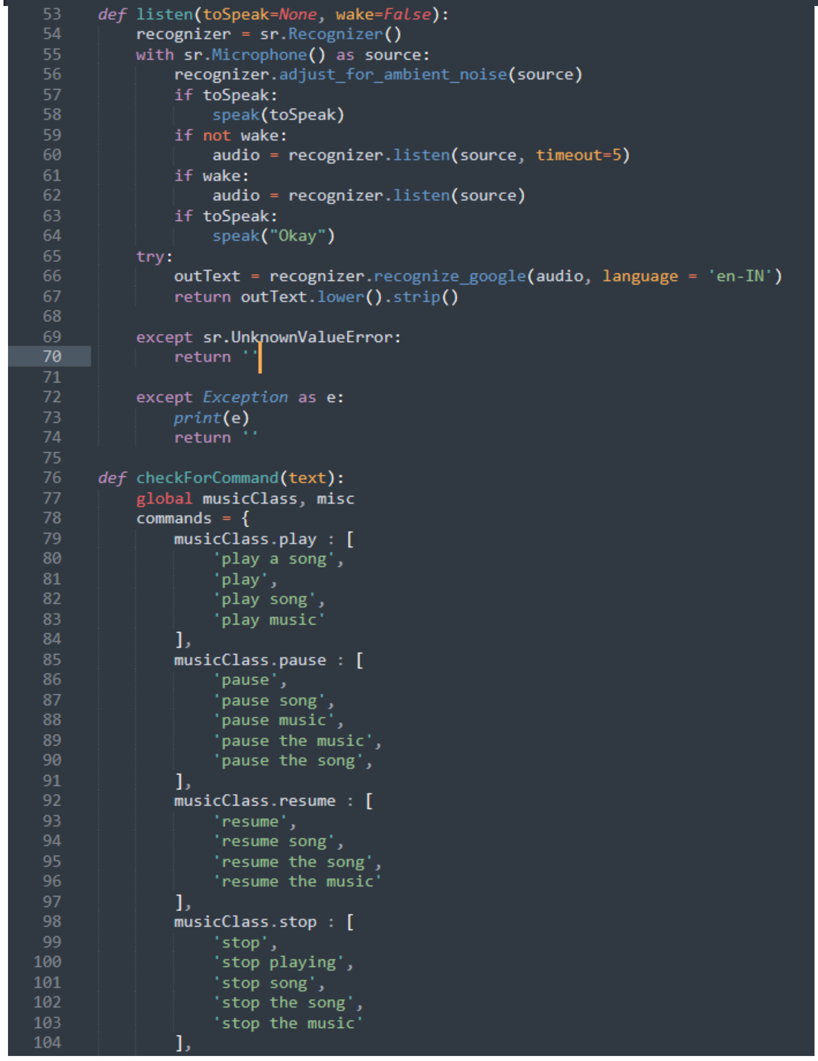


```python
        while str(self.player.get_state()) != 'State.Ended' and self.status != 0:
            pass
        else:
            self.status = 0
            print("Song ended.")

    def stop(self):
        self.status = 0
        self.player.stop()

    def pause(self):
        self.player.pause()

    def resume(self):
        if self.status == 1:
            self.player.play()


if __name__ == "__main__":

    # Testing Purpose
    from tara import listen

    music = musicYoutube(listen)
    music.play()

    while True:
        inp = int(input('->'))
        if inp == 1:
            music.pause()
        if inp == 2:
            music.resume()
        if inp == 3:
            music.play()
        if inp == 4:
            music.stop()
```

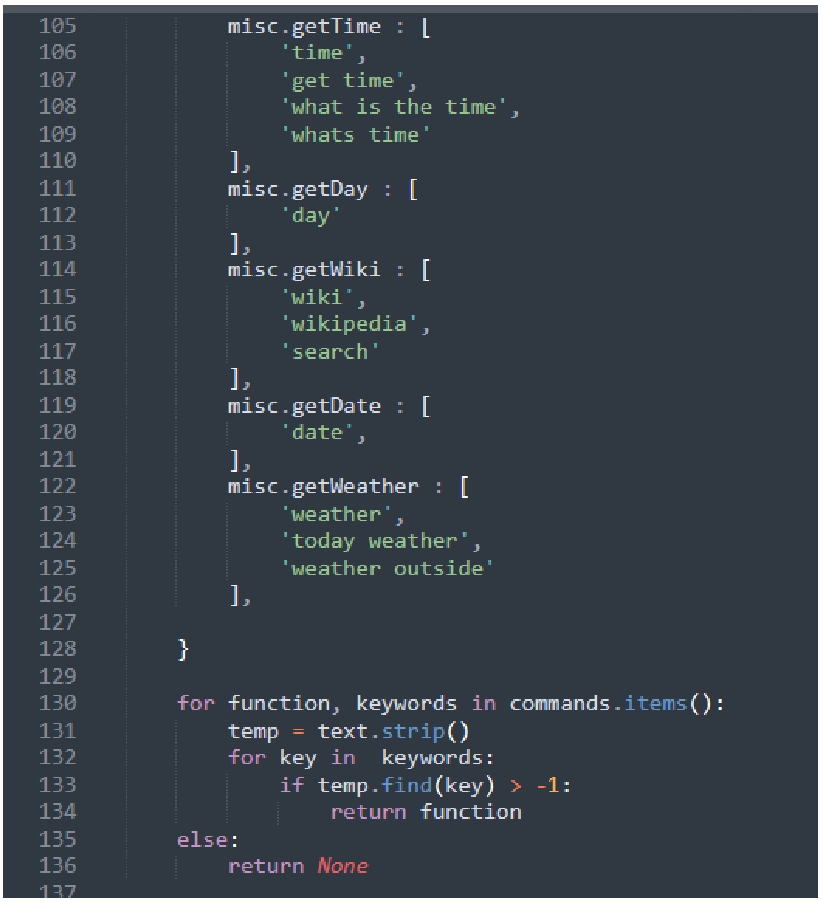
tara.py



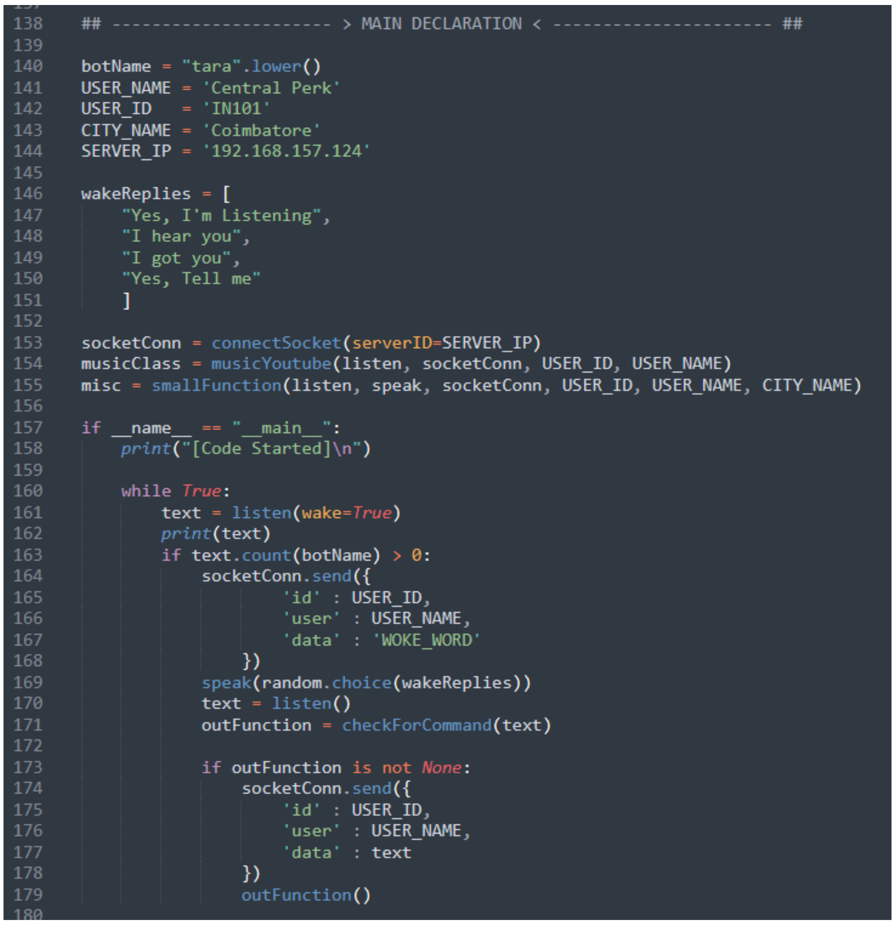
```python
import speech_recognition as sr
from playMusic import *
from smallFunctions import *
import pyttsx3
import random
import socket
import json

class connectSocket:
    def __init__(self, host=socket.gethostname(), port=8120, serverID='localhost', serverPort=8121):
        self.host = host
        self.port = port
        self.serverID = serverID
        self.serverPort = serverPort
        self.__connect()

    def __connect(self):
        self.server = (self.serverID, self.serverPort)
        self.socketHandle = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        self.socketHandle.bind((self.host, self.port))

    def send(self, data):
        threading.Thread(target=self._sendData, args=(data,)).start()

    def _sendData(self, data):
        try:
            self.socketHandle.sendto(str.encode(json.dumps(data)), self.server)
        except Exception as e:
            pass

    def close(self):
        self.socketHandle.close()


def speak(text):
    engine = pyttsx3.init()
    voices = engine.getProperty('voices')
    engine.setProperty('voice', voices[1].id)
    engine.say(text)
    engine.runAndWait()
    engine.stop()
```



```python
def listen(toSpeak=None, wake=False):
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        recognizer.adjust_for_ambient_noise(source)
        if toSpeak:
            speak(toSpeak)
        if not wake:
            audio = recognizer.listen(source, timeout=5)
        if wake:
            audio = recognizer.listen(source)
        if toSpeak:
            speak("Okay")
    try:
        outText = recognizer.recognize_google(audio, language = 'en-IN')
        return outText.lower().strip()

    except sr.UnknownValueError:
        return ''

    except Exception as e:
        print(e)
        return ''

def checkForCommand(text):
    global musicClass, misc
    commands = {
        musicClass.play : [
            'play a song',
            'play',
            'play song',
            'play music'
        ],
        musicClass.pause : [
            'pause',
            'pause song',
            'pause music',
            'pause the music',
            'pause the song',
        ],
        musicClass.resume : [
            'resume',
            'resume song',
            'resume the song',
            'resume the music'
        ],
        musicClass.stop : [
            'stop',
            'stop playing',
            'stop song',
            'stop the song',
            'stop the music'
        ],
```



```
        misc.getTime : [
            'time',
            'get time',
            'what is the time',
            'whats time'
        ],
        misc.getDay : [
            'day'
        ],
        misc.getWiki : [
            'wiki',
            'wikipedia',
            'search'
        ],
        misc.getDate : [
            'date',
        ],
        misc.getWeather : [
            'weather',
            'today weather',
            'weather outside'
        ],

    }

    for function, keywords in commands.items():
        temp = text.strip()
        for key in  keywords:
            if temp.find(key) > -1:
                return function
    else:
        return None
```



```
## --------------------- > MAIN DECLARATION < --------------------- ##

botName = "tara".lower()
USER_NAME = 'Central Perk'
USER_ID   = 'IN101'
CITY_NAME = 'Coimbatore'
SERVER_IP = '192.168.157.124'

wakeReplies = [
    "Yes, I'm Listening",
    "I hear you",
    "I got you",
    "Yes, Tell me"
    ]

socketConn = connectSocket(serverID=SERVER_IP)
musicClass = musicYoutube(listen, socketConn, USER_ID, USER_NAME)
misc = smallFunction(listen, speak, socketConn, USER_ID, USER_NAME, CITY_NAME)

if __name__ == "__main__":
    print("[Code Started]\n")

    while True:
        text = listen(wake=True)
        print(text)
        if text.count(botName) > 0:
            socketConn.send({
                    'id' : USER_ID,
                    'user' : USER_NAME,
                    'data' : 'WOKE_WORD'
                })
            speak(random.choice(wakeReplies))
            text = listen()
            outFunction = checkForCommand(text)

            if outFunction is not None:
                socketConn.send({
                    'id' : USER_ID,
                    'user' : USER_NAME,
                    'data' : text
                })
                outFunction()
```

Advantages:

- It is very effective as an end product
- It can perform multiple functions
- It is very time efficient

Disadvantages:

- We are not in the control of the data that is being collected by it
- It is not secure and safe
- There are possibilities of data breach

Conclusion:

So in this project, we have created out own Alexa/Echo with some basic functionality and also have implemented the concepts of server-client and cloud architecture.