­­­A Report on

TOPIC: - PERSONAL ASSISTANT USING PYTHON

SUBJECT: - PYTHON PROGRAMMING

Submitted By

HARSHIT MALIK

SAP ID: 70472100021

Under the Guidance of

Sachin Bhandari

Logo, company name

Description automatically generated

**Mukesh Patel School of Technology & Management Engineering**

**Department of Computer Engineering**

**Shirpur - 425405**

ABSTRACT

In this modern era, day to day life became smarter and interlinked with technology. We already know some voice assistance like google, Siri. etc. Now in our voice assistance system, it can act as a basic music player, joke teller, Wikipedia. This project works on voice input and give output through voice and displays the text on the screen. The main agenda of our voice assistance makes people smart and give instant and computed results. The voice assistance takes the voice input through our microphone (Bluetooth and wired microphone) and it converts our voice into computer understandable language gives the required solutions and answers which are asked by the user. This assistance connects with the world wide web to provide results that the user has questioned. Natural Language Processing algorithm helps computer machines to engage in communication using natural human language in many forms.

INTRODUCTION

Today the development of artificial intelligence (AI) systems that can organize a natural human-machine interaction (through voice, communication, gestures, facial expressions, etc.) are gaining in popularity. One of the most studied and popular was the direction of interaction, based on the understanding of the machine by the machine of the natural human language. It is no longer a human who learns to communicate with a machine, but a machine learns to communicate with a human, exploring his actions, habits, behaviour and trying to become his personalized assistant.

Virtual assistants are software programs that help you ease your day to day tasks, such as showing weather reports, creating remainders, making shopping lists etc. They can take commands via text (online chatbots) or by voice. Voice-based intelligent assistants need an invoking word or wake word to activate the listener, followed by the command. We have so many virtual assistants, such as Apple’s Siri, Amazon’s Alexa and Microsoft’s Cortana.

This system is designed to be used efficiently on desktops. Personal assistants software improves user productivity by managing routine tasks of the user and by providing information from an online source to the user.

This project was started on the premise that there is enough openly available data and information on the web that can be utilized to build a virtual assistant that has access to making intelligent decisions for routine user activities.

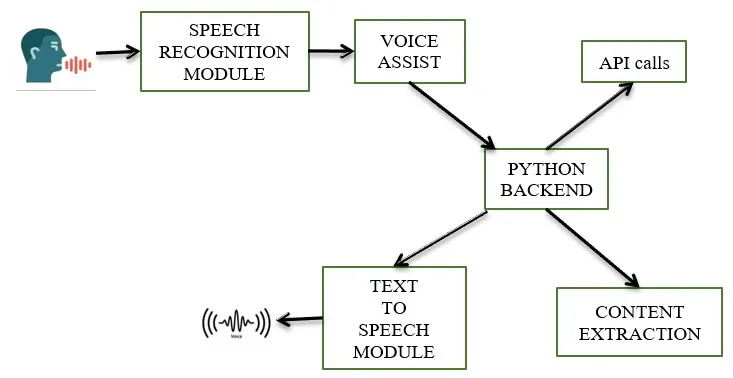
PROBLEM STATEMENT

Nowadays there are many devices which you can use as personal assistants which can make, you're work easily like play music tell you about news and etc, but these assistants are very expensive for example you can take Alexa some people cannot afford it.

PROBLEM SOLUTION

I have made a personal assistant using Python what we have learned in date and the knowledge I have about Python. The assistant can do all the works that an Alexa, or any other personal assistant can do. For example, it can play music tell you about the news shutdown your PC, play a video for you etc.

ARCHITECTURE / FLOW DIAGRAM



CODE DISCRIPTION AND IMPLEMENTATION

import speech\_recognition as sr

import pyttsx3

import pywhatkit

import datetime

import wikipedia

import pyjokes

listener = sr.Recognizer()

engine = pyttsx3.init()

voices = engine.getProperty('voices')

engine.setProperty('voice', voices[1].id)

def talk(text):

    engine.say(text)

    engine.runAndWait()

def take\_command():

    try:

        with sr.Microphone() as source:

            print('listening...')

            voice = listener.listen(source)

            command = listener.recognize\_google(voice)

            command = command.lower()

            if 'alexa' in command:

                command = command.replace('alexa', '')

                print(command)

    except:

        pass

    return command

def run\_alexa():

    command = take\_command()

    print(command)

    if 'play' in command:

        song = command.replace('play', '')

        talk('playing ' + song)

        pywhatkit.playonyt(song)

    elif 'time' in command:

        time = datetime.datetime.now().strftime('%I:%M %p')

        talk('Current time is ' + time)

    elif 'who the heck is' in command:

        person = command.replace('who the heck is', '')

        info = wikipedia.summary(person, 1)

        print(info)

        talk(info)

    elif 'date' in command:

        talk('sorry, I have a headache')

    elif 'are you single' in command:

        talk('I am in a relationship with wifi')

    elif 'joke' in command:

        talk(pyjokes.get\_joke())

    else:

        talk('Please say the command again.')

while True:

    run\_alexa()

PYTHON LIBRARIES USED

* speech\_recognition
* pyttsx3
* pywhatkit
* wikipedia
* pyjokes

DISCRIPTION

* speech\_recognition

Library for performing speech recognition, with support for several engines and APIs, online and offline

* pyttsx

pyttsx is a Python package supporting common text-to-speech engines on Mac OS X, Windows, and Linux.

* pywhatkit

PyWhatKit is a Simple and Powerful WhatsApp Automation Library with many useful Features

* wikipedia  
  **Wikipedia** is a Python library that makes it easy to access and parse data from Wikipedia.Search Wikipedia, get article summaries, get data like links and images from a page, and more. Wikipedia wraps the mediaWiki API so you can focus on using Wikipedia data, not getting it.
* pyjokes

One line jokes for programmers (jokes as a service)

CODE DISCRIPTION

* 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. **An application invokes the pyttsx3.** **init() factory function to get a reference to a pyttsx3.** **Engine instance**.
* getProperty(name : string) – **Gets the current value of an engine property**. setProperty(name, value) – Queues a command to set an engine property. The new property value affects all utterances queued after this command.
* setProperty(name, value) – **Queues a command to set an engine property**. The new property value affects all utterances queued after this command. say(text : unicode, name : string) – Queues a command to speak an utterance. The speech is output according to the properties set before this command in the queue.
* engine. runAndWait(): **This function will make the speech audible in the system**, if you do not write this command then the speech will not be audible to you. engine. setProperty(): This method sets different properties of the model.
* The replace() in Python **returns a copy of the string where all occurrences of a substring are replaced with another substring**.

MEATHODOLOGY

Speech Recognition module

The system uses Google’s online speech recognition system for converting speech input to text. The speech input Users can obtain texts from the special corpora organized on the computer network server at the information centre from the microphone is temporarily stored in the system which is then sent to Google cloud for speech recognition. The equivalent text is then received and fed to the central processor.

Python Backend:

The python backend gets the output from the speech recognition module and then identifies whether the command or the speech output is an API Call and Context Extraction. The output is then sent back to the python backend to give the required output to the user.

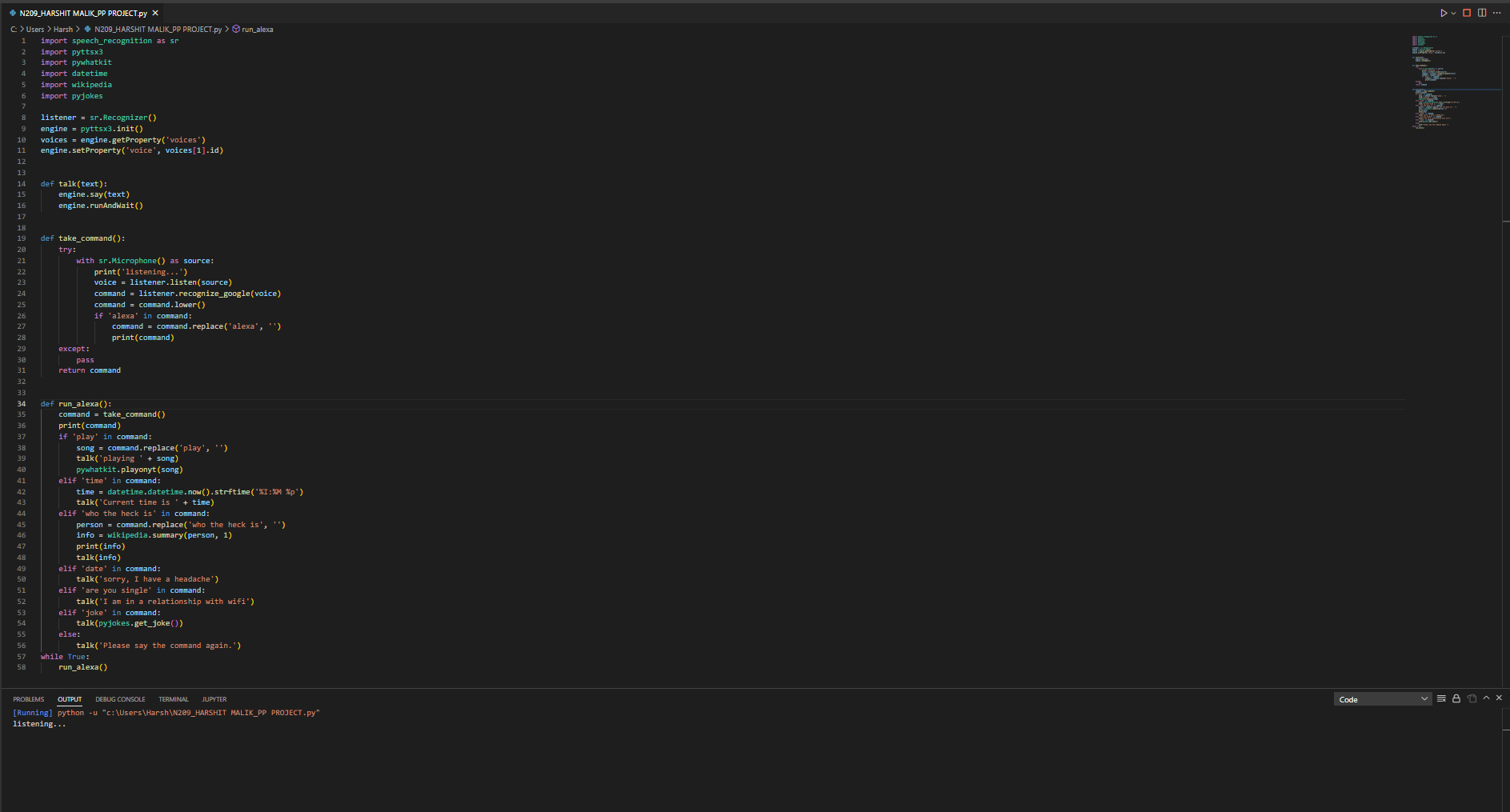
API calls

API stands for Application Programming Interface. An API is a software intermediary that allows two applications to talk to each other. In other words, an API is a messenger that delivers your request to the provider that you’re requesting it from and then delivers the response back to you.

Text-to-speech module

Text-to-Speech (TTS) refers to the ability of computers to read text aloud. A TTS Engine converts written text to a phonemic representation, then converts the phonemic representation to waveforms that can be output as sound. TTS engines with different languages, dialects and specialized vocabularies are available through third-party publishers.

SCREENSHOT



CONCLUSION

In this report “Virtual Assistant Using Python” we discussed the design and implementation of Digital Assistance. The project is built using open source software modules with VS CODE backing which can accommodate any updates shortly. The modular nature of this project makes it more flexible and easier to add additional features without disturbing current system functionalities.

It not only works on human commands but also give responses to the user based on the query being asked or the words spoken by the user such as opening tasks and operations. It is greeting the user the way the user feels more comfortable and feels free to interact with the voice assistant. The application should also eliminate any kind of unnecessary manual work required in the user life of performing every task. The entire system works on the verbal input rather than the next one.

REFERENCES

* docs.python.org
* Youtube.com
* W3Schools.com
* Geeksforgeeks.com

GITHUB