**Personal Voice Assistant**

**PEPSA**

**By**

**AKRITY KUMARI(MIT2020087)**

**HIMANSHU KUMAR(MIT2020090)**

**ANKITA SHAH(MIT2020097)**

**Introduction**

A virtual assistant, also called an AI assistant or digital assistant, is an application program that understands natural language voice commands and completes tasks for the user. It is an example of weak AI that is it can only execute and perform quest designed by the user.

**1. What we are building**

**Our virtual assistant will able to do the followings things-**

Define itself, Tell about its creators, Launch Games, Launch Windows Applications, Open Websites, tells you about almost everything you ask, tells you date and time, greetings, news, play a music for you, etc.

You can interact with your laptop’s microphone/console. The response generated by the assistant will display on the console or as a speech via the speaker.

**2. Background**

The computer needs to be able to recognize our speech. This is where natural language processing comes in. This is a less complicated form of natural language processing. Basically, NLP is a form of AI that lets the computer take in and store information. Then, analyse that information and relay what it is programmed to do. To harness this library all that needs to be typed is import speech\_recognition as sr.

Pepsa draws its inspiration from virtual assistants like Cortana for Windows, and Siri for iOS. It has been designed for carrying out a variety of tasks by employing certain well-defined commands.

**3. Functionality**

**Our virtual assistant has the followings functionalities-**

**1.** assistant\_speaks(input)

2. get\_audio()

3. search\_web(input)

4. process\_text(input)

5. open\_application(input)

6. close\_application(input)

7. playMusic()

8. wishMe()

## 4. Code Explanation-

**So let’s create our own virtual assistant.**

### ****4.1. Required packages and libraries-****

**Install the following essential library to make this assistant work-**

pip install gtts

**pip install playsound**

pip install SpeechRecognition

**pip install requests**

pip install selenium

**pip install pyjokes**

pip install pygame

**4.2 Import the module-**

**Import the following essential library to make this assistant work-**

import speech\_recognition as sr *# to understand whatever the humans speak and converts the speech to text.*

import playsound *# to play saved mp3 file*

from gtts import gTTS # google text to speech

import os # to save/open files

from time import ctime # to get time details

import webbrowser # to open browser

from selenium import webdriver # to open browser

from datetime import datetime #it works on date and time

from datetime import date #it works on date and time

import pyjokes # to drop a random joke

import ctypes #to change desktop wallpaper

import time #helps us to display time

import winsound #to play audio files

from pygame import mixer #to display time audio/music files

import random #to generate a random number

**4.3 How does assistant work**

Assistant speaks function is created to provide the output according to the processed data.

**Playsound Package** is used to play the saved mp3 sound from the system.

***gTTS****– Google Text To Speech, for converting the given text to speech****speech\_recognition****– for recognizing the voice command and converting to text*

def assistant\_speaks(output):

    global num

    # num to rename every audio file

    # with different name to remove ambiguity

    num += 1

    print("PerSon : ", output)

    toSpeak = gTTS(text = output, lang ='en', slow = False)

    # saving the audio file given by google text to speech

    file = str(num)+".mp3

    toSpeak.save(file)

    # playsound package is used to play the same file.

    playsound.playsound(file, True)

    os.remove(file)

**4.4 How does audio received**

get\_audio() function is created to get the audio from user using microphone, the phrase limit is set to 5 seconds. We create the object of sr as ‘rObject’ which will invoke the Microphone method and will record the audio for 5 secs in variable ‘audio’, which can be changed according to convenience. We will also print the query given by the user on the console.

def get\_audio():

rObject = sr.Recognizer()

audio = ''

with sr.Microphone() as source:

print("Speak...")

audio = rObject.listen(source, phrase\_time\_limit = 5)

print("Stop.")

text = rObject.recognize\_google(audio, language ='en-US')

print("You : ", text)

return text

**4.5 How does the assistant address us with our name**

the ‘get\_audio()’ will try to fetch audio from the computer’s microphone. It will process the audio and return text in ‘name’ variable. The assistant can then greet us with our name.

assistant\_speaks("What's your name, Human?")

name ='Human'

name = get\_audio()

assistant\_speaks("Hello, " + name + '.')

**4.6** **We want to continuously listen to input from the user**

the ‘get\_audio()’ will try to fetch audio from the computer’s microphone continuously. It will process the audio and return text in ‘text’ variable. We can use this ‘text’ variable to perform some action according to user input.

while(1):

assistant\_speaks("What can i do for you?")

text = get\_audio().lower()

**4.7 Process the Query**

If the text contain “exit” or “sleep” or “bye”, then the assistant will stop its working and would terminate.

if "exit" in str(text) or "bye" in str(text) or "sleep" in str(text):

assistant\_speaks("Ok bye, "+ name+'.')

break

Otherwise, we will call the process\_text(input) to process the query to perform the desired actions.

process\_text(text)

**4.8 Process the text**

We are using the membership operator ‘in’ to match queries in user input.

Membership operators are operators used to validate the membership of a value. It test for membership in a sequence, such as strings, lists, or tuples.

**If ‘search’ or ‘play’ is found in user input ‘text’ then we will call the search\_web(input) functionality.**

if 'search' in input or 'play' in input:

# a basic web crawler using selenium

search\_web(input)

return

**If ‘open’ is found in user input ‘text’ then we will call the open\_application(input) functionality**.

if 'open' in input:

# function to open different application availaible

open\_application(input.lower())

return

**If ‘close’ is found in user input ‘text’ then we will call the close\_application(input) functionality**.

if "chrome" in input:

assistant\_speaks("Closing Google Chrome")

os.system("taskkill /f /im chrome.exe")

return

**4.9 How does it search web**

**search\_web**is just a web crawler which uses selenium package to process. Here, we can search **google**, **wikipedia**and can open **YouTube**. We included the name and it will open it in the Firefox browser.

def search\_web(input):

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

browser = webdriver.Firefox()

We have again used the membership operator ‘in’ to check whether the user query

It will split the query after matching with the desired keys and store the other part into var ‘query’ and join with the url to launch the user request by using browser.get() method

if 'wikipedia' in input.lower():

assistant\_speaks("Opening Wikipedia")

indx = input.lower().split().index('wikipedia')

query = input.split()[indx + 1:]

url = f"https://en.wikipedia.org/wiki/" + '\_'.join(query)

browser.get(url)

return

elif 'youtube' in input.lower():

assistant\_speaks("Opening in youtube")

indx = input.lower().split().index('youtube')

query = input.split()[indx + 1:]

url = f"https://www.youtube.com/results?search\_query={query}"

browser.get(url)

return

elif 'google' in input.lower():

assistant\_speaks("Opening in google")

indx = input.lower().split().index('google')

query = input.split()[indx + 1:]

url = f"https://google.com/search?q={query}"

browser.get(url)

return

**4.10 How does it open application**

**open\_application**is just a function uses *os package* to open the application present in the system.

It make use of the **os.startfile()** method which allows us to “start” a file with its associated program. In other words, we can open a file with it’s associated program, just like when you double-click a PDF and it opens in Adobe Reader.

if "chrome" in input:

assistant\_speaks("Google Chrome")

os.startfile("C:\Program Files (x86)\Google\Chrome\Application\chrome.exe")

return

Here, we have passed a fully qualified path to **os.startfile** that tells it to open the google application. Same way, we can do for any application which we are using frequently.

**4.11 How does the desktop background changes and a sound is heard when application is run**

When we run the application, the background wallpaper of the system changes and a sound is heard. Whenever we give any command, the background continues changing for few seconds. When we exit from the application, the background is restored to the previous background and the sound will again.

import ctypes

ctypes.windll.user32.SystemParametersInfoW(20, 0,r"your\path\location\image.jpg", 0)

music\_dir = r" your\path\location\sound.mp3"

winsound.PlaySound("music\_dir", winsound.SND\_ASYNC |winsound.SND\_ALIAS )

**4.12 How does it tell joke**

Import the pyjokes module in a Python file and use the get\_joke function to easily drop a random joke into your application.

Import pyjokes

Assistant\_speaks(pyjokes.get\_joke())

**4.13 How does it play songs**

Pygame module is used to load and play the sounds. Import the pygame.mixer module in the python file. The pygame.mixer is used to play music/audio files(like .mp3 file).

The song can be paused, un-paused or stopped according to users’ wish.

from pygame import mixer

mixer.init()

mixer.music.set\_volume(0.50)

filename = random.choice(L)

mixer.music.load( filename )

mixer.music.play( )

Here, L is the list of frequent songs that one plays. And random.choice() method selects any one of the song present in the list and plays it

**4.14 How does it Search for queries which are not specifically coded**

If the user asks any query which does not match with any of the above instances, then the application would ask to search that query on the web. If said yes, then browser will open

assistant\_speaks("I can search the web for you, Do you want to continue?")

ans = get\_audio()

input="search"+input

if 'yes' in str(ans) or 'yeah' in str(ans):

search\_web(input)

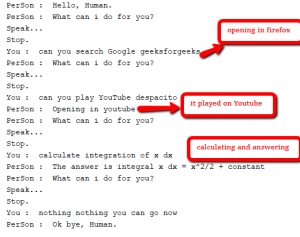
else:

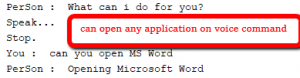
return

**4.14 It can answer some specific Queries**

It can answer some specific queries like define yourself, who created you, your functionality, tell something about my college, say any jokes, say my branch, tell today’s date, what’s the time.

**5. WORKING**

****

****