

Virtual Voice Assistant

Project submitted to the
SRM-University – AP, Andhra Pradesh
For the partial fulfilment of the requirements to award the degree of

Bachelor of technology
In
Computer science engineering
School of engineering and sciences

Submitted by
Ritika Khande (AP21110010142)
Adi Vishnu Avula (AP21110010186)
Krishna Varshita Borra (AP21110010187)
Sai Deepika Daram (AP21110010188)



Under the guidance of
(Poonam Yadav)
SRM University – AP
Neerukonda, Mangalagiri, Guntur
Andhra Pradesh – 522 240
[December, 2022]

Certificate

Date: 06-Dec-22

This is to certify that the work present in this Project entitled “**VIRTUAL VOICE ASSISTANT**” has been carried out by **Group 6 (Ritika Khande, Adi Vishnu Avula, Krishna Varshita Borra, Sai Deepika Daram)** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor

(Signature)

Prof. / Dr. Poonam Yadav

Acknowledgements

We (Group 6) would like to express my special thanks of gratitude to our professor Poonam Yadav, who gave us the golden opportunity to do this project of “Virtual Voice assistant” as a part of the course Hands on Using Python (CSE 106L). I got to learn a lot from this project about different core concepts of Python Programming. I am very grateful to ma’am for her support and guidance in completing this project. Secondly, I would also like to thank my friends and who helped me a lot in finalising this project within the limited time frame.

Group 6

CSE-C

Table of Contents

Contents

Certificate.....	3
Acknowledgments	5
Table of Contents	7
Abstract.....	9
Statement of Contributions.....	11
1. Introduction.....	13
2. Methodology.....	15
2.1 Design:	15
2.2 Implementation.....	15
3. Results.....	19
4. Concluding Remarks.....	22

Abstract

A voice assistant is a computer assistant that listens to specific vocal commands and responds with relevant details or carries out certain tasks as asked by the user using speech recognition, language processing algorithms, and voice synthesis.

Voice assistants can deliver important information based on the user's spoken orders, also known as intentions, by listening for certain keywords and removing background noise.

So, in this project, we develop a virtual voice assistant that can perform various tasks on your computer using commands given through voice. This application is similar to google assistant, Siri, using python.

Statement of Contributions:

This was the work contributed by the team members:

Adi Vishnu Avula - Designing the functions, Coding

Krishna Varshita Borra - Debugging and Report

Sai Deepika Daram - Listing out Commands and Report

Ritika Khande - Listing out Commands and Report

1. Introduction:

Even while applications like Siri have only been available for ten years, voice assistants have a very lengthy history that really dates back more than 100 years.

Radio Rex, the first voice-activated product, was introduced in 1922. Simple instructions came with this toy, which had a toy dog that would stay within a dog house until the owner called out "Rex," at which time it would leap out.

Google has progressively been rolling out voice search for its Google mobile apps on several platforms in 2008, the year Android was originally published, and a specific Google Voice Search Application was released in 2011. This resulted in more sophisticated functionality, which ultimately gave rise to Google Voice Assistant.

Then, in 2010, Siri came after this. The original software was published in 2010 on the iOS App Store, and two months later Apple purchased it. Later, Siri was formally launched as an integrated voice assistant within iOS with the introduction of the iPhone 4s. Since then, Siri has been ported to every Apple product on the market, creating an ecosystem that connects all of the devices.

Voice recognition functions by converting the analogue signal produced by the user's voice into a digital signal. Following this, the computer attempts to discern the user's intent by matching the digital signal to words and phrases. The computer needs a database of existing words and syllables in a particular language to be able to closely match the digital signal with in order to accomplish this.

The ease of use of voice assistants is one of its main advantages. Using your voice and your natural conversational inclinations will suffice in place of needing to learn how to utilise specialised physical gadgets like mice and touch displays.

Today's environment has a wide variety of uses for voice assistants. When your hands are full and you can't use a touch screen or keyboard, for instance, or when you need to change the music while driving, you may just tell a voice assistant to "play music." As a result, driving is safer and the risk of distracted driving is reduced.

Functionalities of this project:

We have designed the project in a way that, users can perform their tasks very easily in a short time.

This application allows the user to:

- Know the time
- Get content from Wikipedia
- Open Popular Websites
- Search in popular search engines
- Get Directions to a place
- Create or Open Local Folders

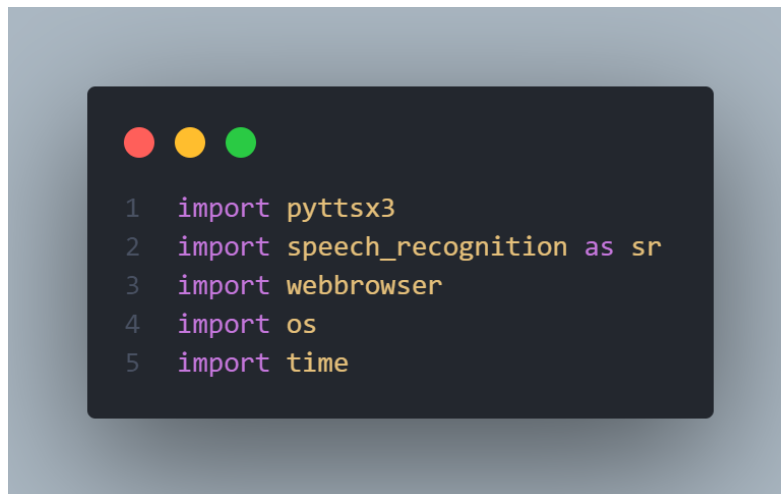
This software is platform independent and can be used in any system

2. Methodology

2.1 Design:

We designed this project using Python Programming Language, using some modules for speech recognition and taking in voice input. Then we have compared the voice input string with various conditions, and if a condition is matched with the user's command, the block of code under that condition will be executed.

2.2 Implementation:



These are the libraries that we have used in our project.

pytsx3 - This is a text-to-speech conversion library in Python which works offline

speech_recognition - Library for performing speech recognition, with support for several engines

webbrowser - The webbrowser module provides a high-level interface to allow displaying web-based documents to users

os - The OS module in Python provides functions for interacting with the operating system, like working with files

time - This module provides various time-related functions

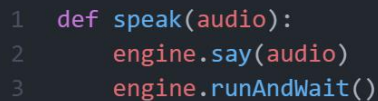


```

1 engine = pyttsx3.init("sapi5")
2 voices = engine.getProperty("voices")
3
4 engine.setProperty('voice', voices[1].id) # 1 for female and 0 for male voice

```

First, we will start by setting a variable engine, with the sapi5 API (An API from Microsoft for Text to Speech). Using the setProperty function, we will initialize the voice to a female voice

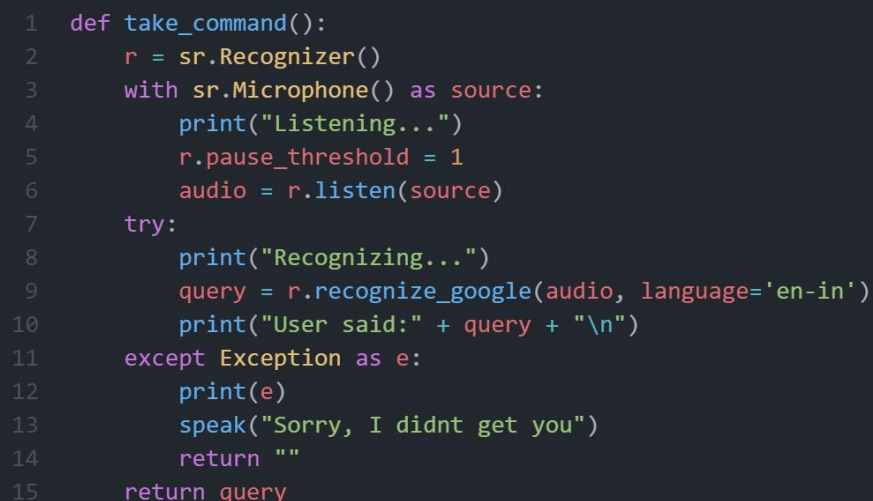


```

1 def speak(audio):
2     engine.say(audio)
3     engine.runAndWait()

```

Now, we defined a user-defined function speak(audio), where audio is a string argument, which is used to make the Text-to-speech engine speak out the given string to the user.



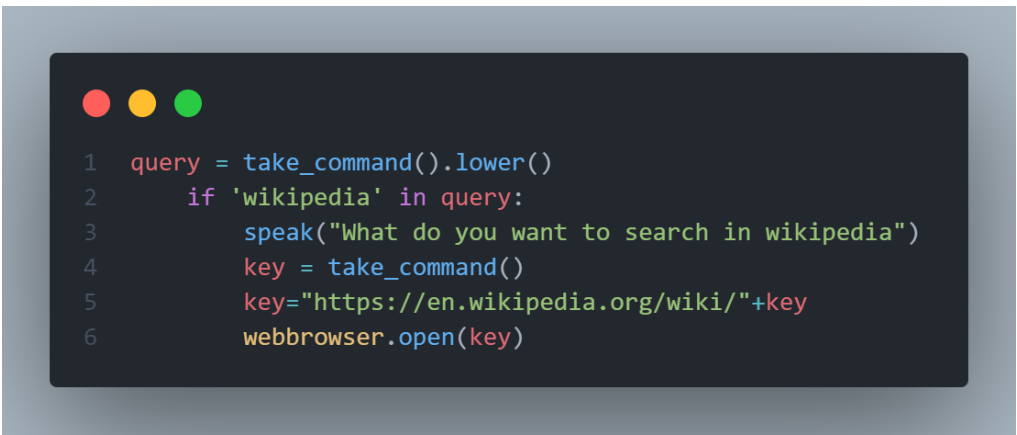
```

1 def take_command():
2     r = sr.Recognizer()
3     with sr.Microphone() as source:
4         print("Listening...")
5         r.pause_threshold = 1
6         audio = r.listen(source)
7     try:
8         print("Recognizing...")
9         query = r.recognize_google(audio, language='en-in')
10        print("User said:" + query + "\n")
11    except Exception as e:
12        print(e)
13        speak("Sorry, I didnt get you")
14        return ""
15    return query

```

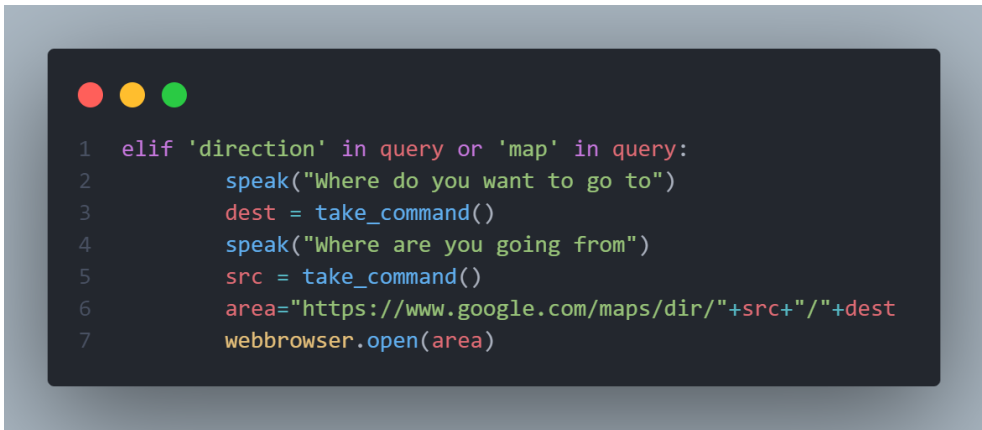

We have also defined another user defined function `take_command()` to get the `string(text)` value of user's speech input. In this function, we make use of the `speech_recognition` library that we imported earlier, and use the `Microphone()` function to set the default system microphone as the audio source, and then we will use the Speech Recognition API from Google to convert the audio to text.

Now, we will check the user's command with various conditions, and if a condition is matched, we will execute that operation. For example,



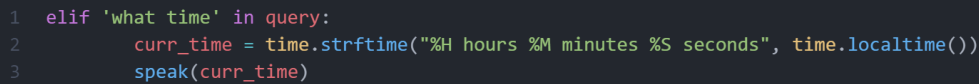
```
1 query = take_command().lower()
2 if 'wikipedia' in query:
3     speak("What do you want to search in wikipedia")
4     key = take_command()
5     key="https://en.wikipedia.org/wiki/"+key
6     webbrowser.open(key)
```

This is one of the pre-defined conditions to be checked with the user's command. If the user says "Wikipedia" in his command, then the program asks him what topic he wants to search in Wikipedia, and then searches for it and opens it in his webbrowser.



```
1 elif 'direction' in query or 'map' in query:
2     speak("Where do you want to go to")
3     dest = take_command()
4     speak("Where are you going from")
5     src = take_command()
6     area="https://www.google.com/maps/dir/"+src+"/"+dest
7     webbrowser.open(area)
```

Similarly, if the user wants to know the directions to any places and uses the word "Direction" in his command, then the program asks the user for the Starting point and Destination, and opens Google Maps in their web browser showing the required directions.

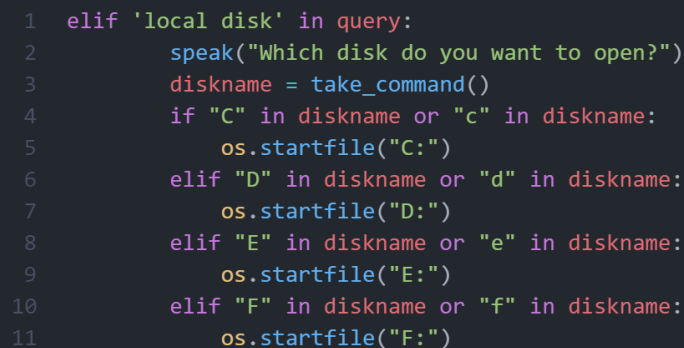


```

1 elif 'what time' in query:
2     curr_time = time.strftime("%H hours %M minutes %S seconds", time.localtime())
3     speak(curr_time)

```

If the user wants to know the current time, and if the program encounters the keywords “what time” in the command, then the program speaks out the current time for the user in 24-hour format Hours, Minutes and Seconds.

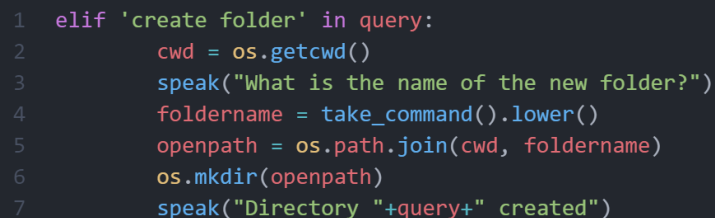


```

1 elif 'local disk' in query:
2     speak("Which disk do you want to open?")
3     diskname = take_command()
4     if "C" in diskname or "c" in diskname:
5         os.startfile("C:")
6     elif "D" in diskname or "d" in diskname:
7         os.startfile("D:")
8     elif "E" in diskname or "e" in diskname:
9         os.startfile("E:")
10    elif "F" in diskname or "f" in diskname:
11        os.startfile("F:")

```

Now, if the user wants to open a local storage drive, the program checks for the keywords “local disk” in the user’s command, and opens the corresponding disk drive as per the choice of the user. Since the drive letter is common for almost all PC’s, this code can run independent of the platform.



```

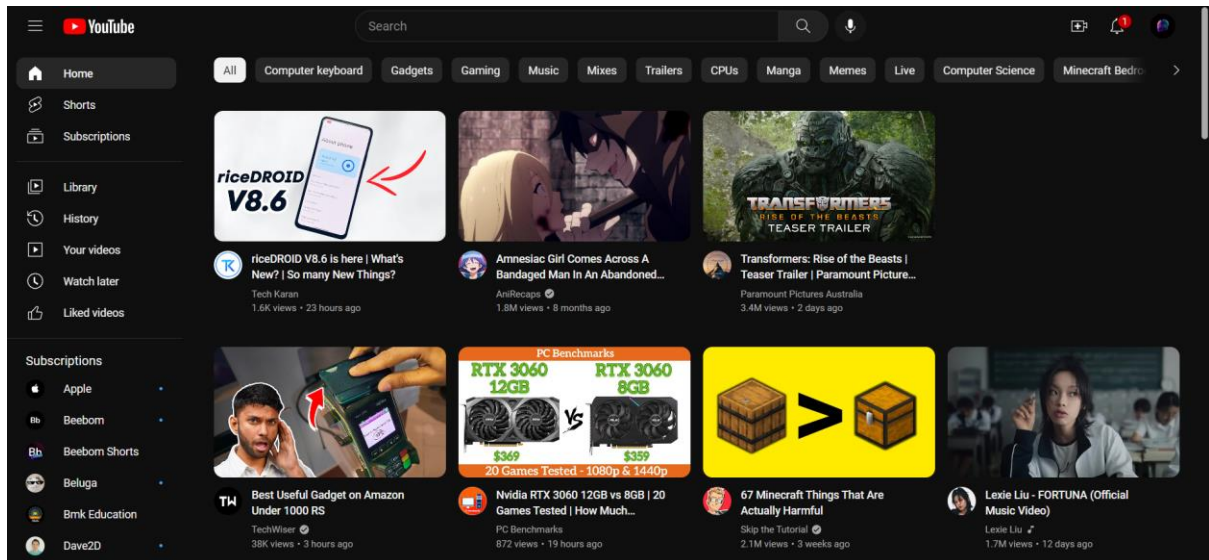
1 elif 'create folder' in query:
2     cwd = os.getcwd()
3     speak("What is the name of the new folder?")
4     foldername = take_command().lower()
5     openpath = os.path.join(cwd, foldername)
6     os.mkdir(openpath)
7     speak("Directory "+query+" created")

```

The program can also create folders in the present working directory by taking the name of the folder as an input from the user. This is triggered by the “create folder” keywords in the user’s command.

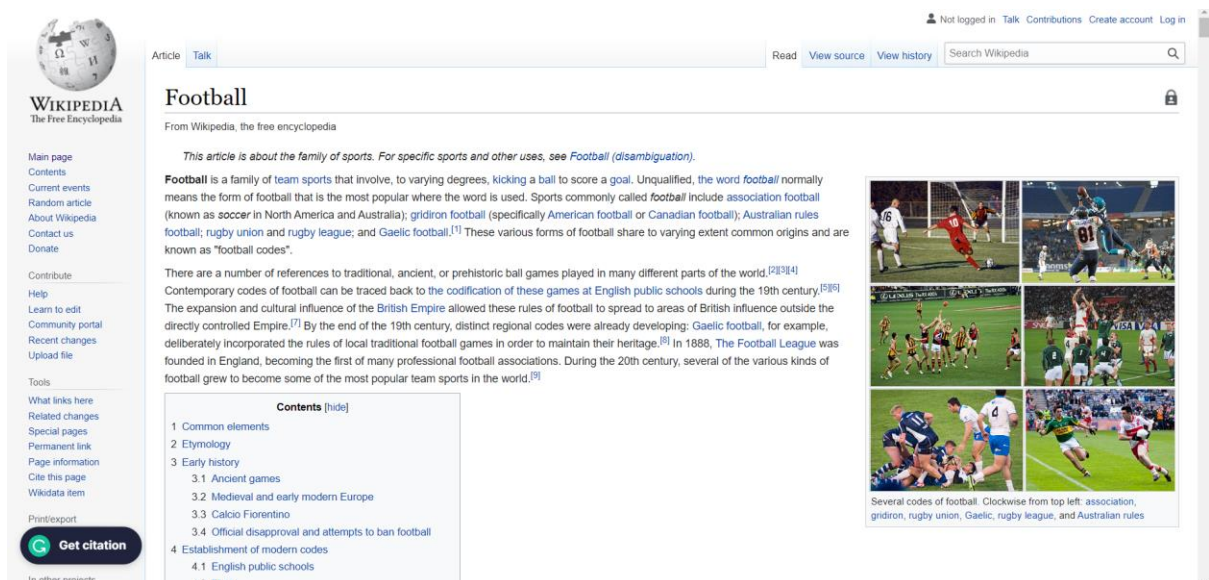
3. Results

```
Listening...
Recognizing...
User said:open YouTube
```



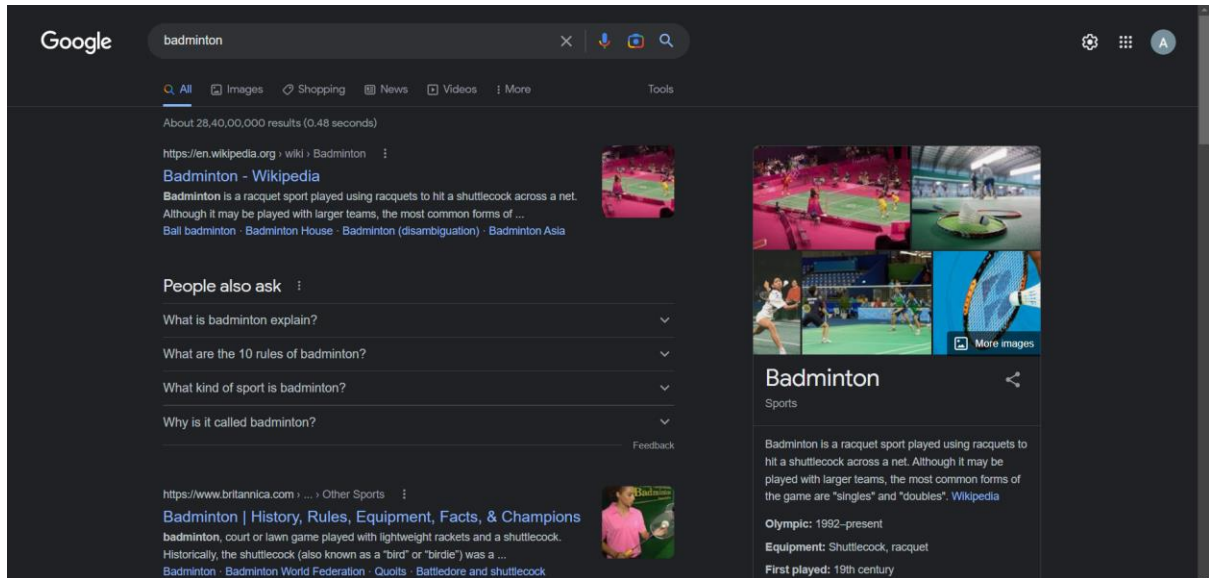
```
Listening...
Recognizing...
User said:Wikipedia

Listening...
Recognizing...
User said:football
```

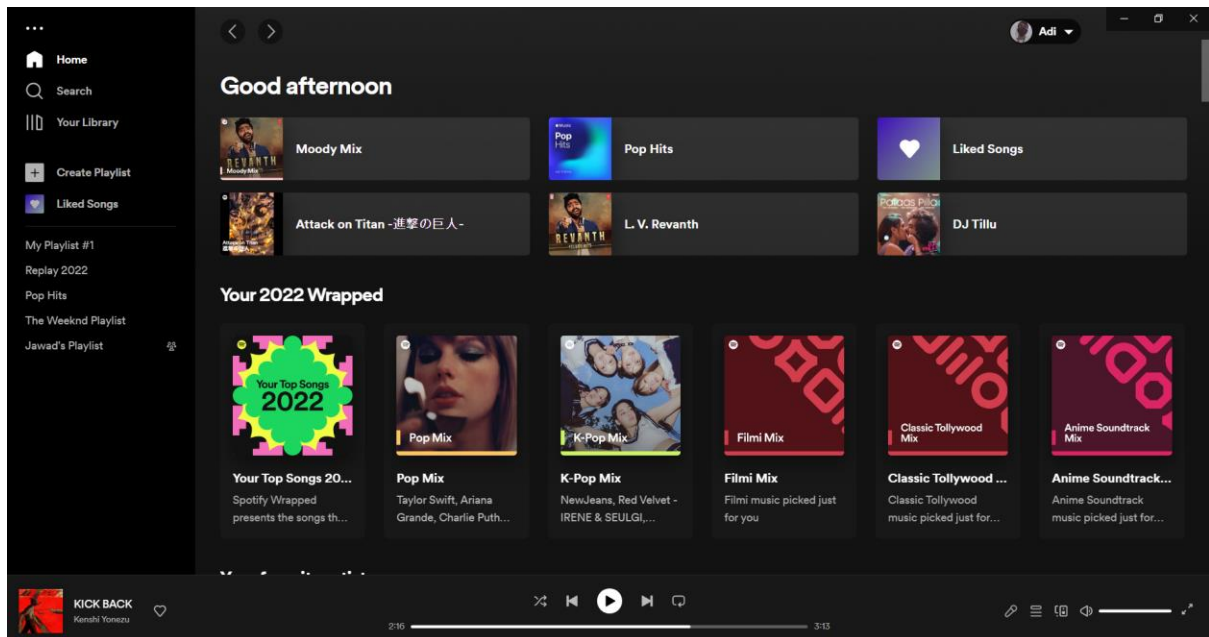


Listening...
Recognizing...
User said:search

Listening...
Recognizing...
User said:badminton

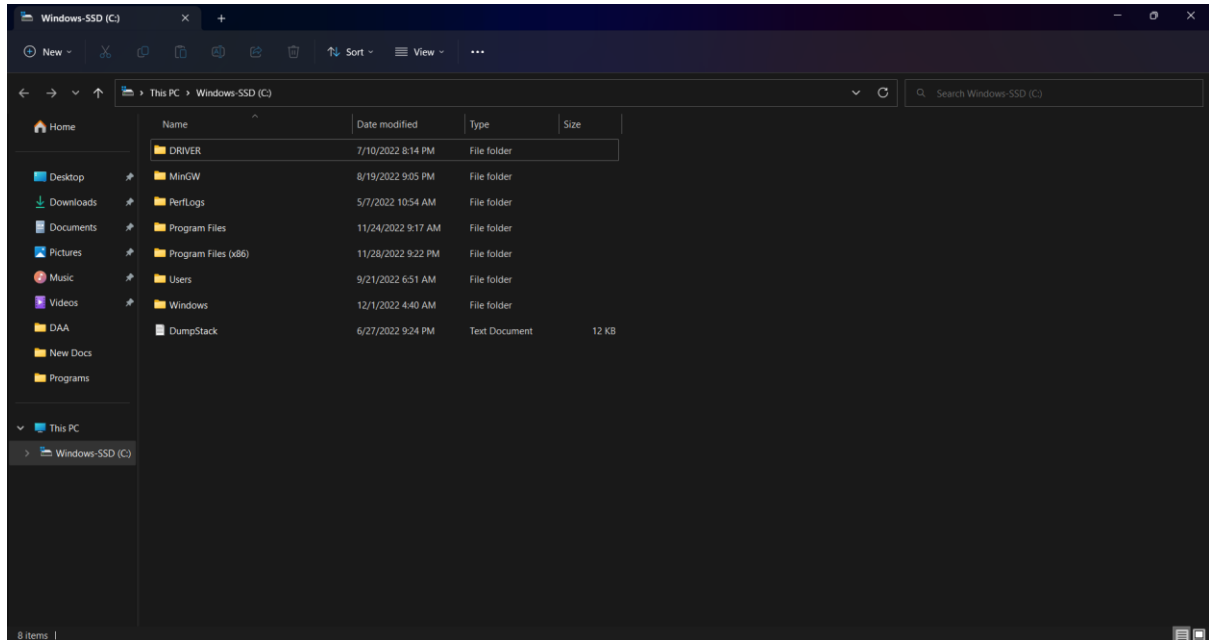


Listening...
Recognizing...
User said:play music



```
Listening...
Recognizing...
User said:local disk

Listening...
Recognizing...
User said:c
```



```
Listening...
Recognizing...
User said:exit
```

```
PS C:\Users\adivi\Documents\Programs>
```

Program Exited

4. Concluding Remarks:

In the process of making this project, we have understood the concepts of python and how to implement them. This project can be used for performing continuous operations in a computer with just voice commands.