

MINOR PROJECT REPORT

ON

VOICE ASSISTANT

Submitted in the partial fulfilment of the requirement for the
award of the degree of

MASTER OF COMPUTER APPLICATION



SUMMITTED TO
DR. RAJAN GUPTA
ASSOCIATE PROFESSOR
VSIT, VIPS

SUBMITTED BY
MUHAMMED RAIHAN P. S. (13017704423)
MANOJ JAISWAL (12517704423)
MCA 1C

(BATCH: 2023-2025)

Vivekananda Institute of Professional Studies - Technical Campus
(Affiliated to Guru Gobind Singh Indraprastha University)

DECLARATION

This is to certify Muhammed Raihan P. S. (13017704423) and Manoj Jaiswal (12517704423) of MCA 1st Semester from Vivekananda Institute of Professional Studies, Delhi has presented this project work entitled “CYPHER voice assistant” in partial fulfilment of the requirements for the award of the degree of Master of Computer Application under our supervision and guidance.

Dr. Rajan Gupta
(Project Coordinator)

Dr. Supriya Madan
(Dean)

ACKNOWLEDGEMENT

It is our proud privilege to express our profound gratitude to the entire management of Vivekananda Institute of Professional Studies and teachers of the institute for providing us with the opportunity to avail the excellent facilities and infrastructure. The knowledge and values inculcated have proved to be of immense help at the very start of my career. Special thanks to Hon'ble Founder, Vivekananda Institute of Professional Studies, Delhi for having provided us an excellent infrastructure at VSIT.

I am grateful to Dr. Supriya Madan (Dean, VSIT), and “project guide” for their astute guidance, constant encouragement and sincere support for this project work.

Sincere thanks to all my family members, seniors and friends for their support and assistance throughout the project.

MUHAMMED RAIHAN P. S. (13017704423)

MANOJ JAISWAL (12517704423)

TABLE OF CONTENT

Declaration	2
Acknowledgement	3
Table of Content	4
Table of Figures	5
1 INTRODUCTION	6
1.1 What is an AI	7
1.2 What is an AI voice assistant	8
1.3 Advantages of the system	8
1.4 Present system and how are they used	9
1.5 Use cases	10
1.6 Introduction	11
1.7 Problem statement	11
1.8 Objective of the project.....	12
2 DESIGN OF THE SYSTEM	13
2.1 Flow chart	14
2.2 Hardware requirements	15
2.3 Software requirements	16
3 IMPLEMENTATION AND CODING	17
3.1 Languages	18
3.2 Libraries	19
3.3 Coding	21
4 TESTING AND TEST RESULT	39
4.1 Software testing and objective of testing	40
4.2 Sample test data / Output screen printout	42
5 CONCLUSION	45
5.1 Conclusion	46
5.2 Future scope	46
5.3 Limitation of the system	47
6 REFERENCES	48

TABLE OF FIGURES

Fig 1.1:	AI Evolution	7
Fig 2.1:	Flow chart of voice assistant	14
Fig 2.3.1:	IDE Jupyter Notebook	16
Fig 2.3.2:	QT5 Designer	16
Fig 3.3.1:	Categories of functions	21
Fig 3.3.2:	List of functions	22
Fig 4.1.1:	Video recorder input/output screen	40
Fig 4.2.1:	Wikipedia search input and output screen	42
Fig 4.2.2:	Google search input and output screen	42
Fig 4.2.3:	Weather forecast input and output screen	42
Fig 4.2.4:	News input and output screen	43
Fig 4.2.5:	Social media app/web input and output screen .	44
Fig 4.2.6:	Open camera input and output screen	44

CHAPTER 1

INTRODUCTION

1.1 WHAT IS AN AI

Artificial intelligence (AI) refers to the ability of a computer or machine to perform tasks that normally require human intelligence, such as learning, problem-solving, and decision-making. AI can be classified into two main categories: narrow or general. Narrow AI is designed to perform specific tasks, such as image or speech recognition, and is trained to perform these tasks using large amounts of data and algorithms. In contrast, general AI, also known as strong AI, is designed to be able to perform any intellectual task that a human can, and can learn and adapt to new situations.

One of the main goals of AI research is to create systems that can learn and adapt to new situations on their own, without being explicitly programmed to do so. This is known as machine learning, and it involves the use of algorithms and statistical models to enable computers to learn from data and improve their performance over time.

AI has the potential to revolutionize many different fields, from healthcare and education to transportation and manufacturing. It can help to improve the efficiency and accuracy of tasks, and can also enable the development of new technologies and applications that were previously not possible.

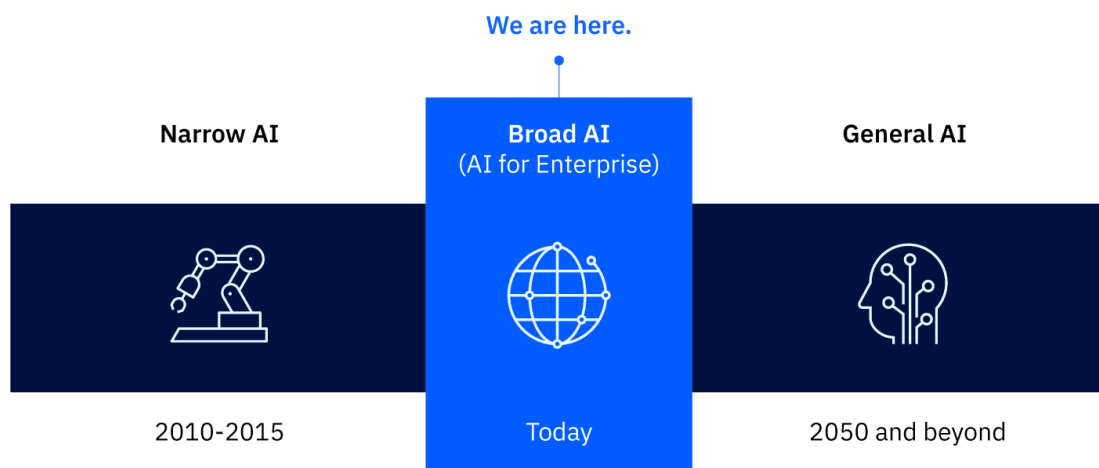


Fig 1.1: AI Evolution

1.2 WHAT IS AN AI VOICE ASSISTANT

Voice assistant is a type of software program that uses voice recognition technology to allow users to interact with a device or service using natural language voice commands. In the current scenario, advancement in technologies are such that they can perform any task with same effectiveness or can say more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time.

Some common examples of voice assistants include Apple's Siri, Amazon's Alexa, and Google Assistant. These assistants can be used to perform a wide range of tasks, such as setting reminders, answering questions, playing music, and controlling smart home devices.

Now the basic question arises in mind that how it is an AI? The virtual assistant that I have created is like if it is not an A.I, but it is the output of a bundle of the statement. But fundamentally, the main purpose of A.I machines is that it can perform human tasks with the same efficiency or even more efficiently than humans. It is a fact that my virtual assistant is not a very good example of A.I., but it is an A.I.

1.3 ADVANTAGES OF THE SYSTEM

- **Convenience:** Voice assistants allow users to perform tasks and retrieve information quickly and easily, without the need to type or use a touchscreen. This can be particularly useful when users are busy or have their hands full.
- **Time-saving:** Voice assistants can be used to perform tasks quickly and efficiently, saving users time and effort.
- **Increased efficiency / Multitasking:** By using a voice assistant, users can multitask and complete tasks more quickly, which can increase productivity and efficiency.
- **Improved accessibility:** Voice assistants can be particularly useful for people with mobility impairments or other disabilities that make using traditional input methods challenging.
- **Hands-free operation:** Voice assistants allow users to complete tasks and retrieve information hands-free, which can be especially useful while driving or when users are occupied with other tasks.
- **Improved accuracy:** With advances in natural language processing, many voice assistants are becoming increasingly accurate at understanding and responding to voice commands.
- **Integration with other devices:** Many voice assistants can be integrated with other devices, such as smart home systems, allowing users to control multiple devices using voice commands.

1.4 PRESENT SYSTEM HOW ARE THEY USED








We are familiar with many existing voice assistants like Alexa(Amazon), Siri(Apple), Google Assistant, Cortana(Microsoft) which uses concept of language processing, and voice recognition. They listens the command given by the user as per their requirements and performs that specific function in a very efficient and effective manner. As these voice assistants are using Artificial Intelligence hence the result that they are providing are highly accurate and efficient. These assistants can help to reduce human effort and consumes time while performing any task, they removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform task. These assistants are no less than a human assistant but we can say that they are more effective and efficient to perform any task. The algorithm used to make these assistant focuses on the time complexities and reduces time. But for using these assistants one should have an account (like Google account for Google assistant, Microsoft account for Cortana) and can use it with internet connection only because these assistants are going to work with internet connectivity. They are integrated with many devices like, phones, laptops, and speakers etc.

Voice assistants are activated by a trigger word or phrase, such as "Hey Siri" or "Ok Google." Once activated, the user can issue a command or ask a question, and the voice assistant will process the request and provide a response. Some voice assistants also have the ability to learn and adapt to a user's preferences and habits over time, making them more efficient and personalized.

Voice assistants are becoming increasingly popular and are being integrated into a wide range of devices and platforms. They are often used as a convenient way to interact with technology and perform tasks hands-free, making them particularly useful while driving or when the user's hands are occupied.



1.5 USE CASES

	BANKING Open an account Check balance and transfer money Schedule an appointment Report a lost/stolen card
	HEALTH CARE Schedule an Appointment Get care team Medication list Read change an Appointment
	RETAIL Request return/exchange Place an order Cancel an order Check order status Modify an order
	MANUFACTURING Machine maintenance alerts Maintain safety checks Fetch equipment Information
	MEDIA & ENTERTAINMENT 24x7 user assistance Enhance customer engagement Large outreach
	TRAVEL Virtual help desk Booking assistance
	SALES AND MARKETING Forecast report E-commerce assistance Personalized shopping experiences Provide information about products to the customers

1.6 INTRODUCTION

Artificial intelligence has the potential to significantly improve the quality of life for individuals. In recent years, the rapid advancement of technology has ushered in a new era of human-machine interaction, with voice assistants playing a pivotal role in shaping this landscape. Voice assistants, powered by artificial intelligence and natural language processing, have become integral parts of our daily lives, offering convenience and efficiency in accessing information, controlling smart devices, and performing various tasks. Additionally, AI-powered virtual assistants such as Siri, Alexa, and Google Assistant can assist with tasks such as reading and sending messages, making phone calls, and setting reminders. Overall, AI-based solutions have the potential to significantly improve the quality of life for visually impaired individuals and should be further developed and integrated into daily life.

1.7 PROBLEM STATEMENT

- **Inadequate Contextual Understanding:** Existing voice assistants often struggle to accurately interpret complex or nuanced queries, leading to misinterpretations and inefficient responses.
- **Privacy and Security Concerns:** The storage and processing of sensitive user data by voice assistants raise significant privacy and security issues, demanding robust measures to protect user information from unauthorized access and misuse.
- **Inclusivity Challenges:** The lack of consistent recognition of accents, dialects, and linguistic variations hinders an inclusive user experience, creating disparities in accessibility and usability.
- **Industry-Specific Adaptation:** The seamless integration of voice assistants into various industries, such as healthcare, education, and business, requires tailored solutions to meet specific needs and ensure compatibility across diverse platforms.
- **Compatibility and Integration:** Ensuring voice assistants seamlessly integrate with a wide range of devices and platforms is crucial for their widespread adoption and effectiveness in diverse environments.

1.8 OBJECTIVE OF THE PROJECT

The main objective of an AI voice assistant is to access information, and interact with technology using voice.

Technologies used in this system

Voice Assistant

- Text to speech conversion (TTS)
- Speech to text conversion
- User query translation (Hindi to English)
- Widgets
- Web Browser
- Operating system
- Camera
- Others

CHAPTER 2

DESIGN OF THE SYSTEM

2.1

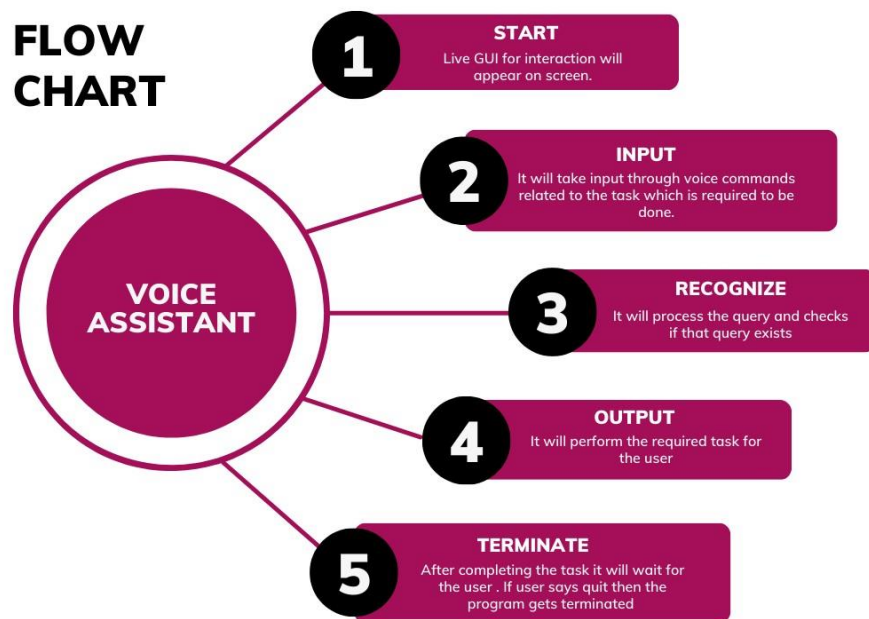


Fig 2.1: Flow chart of voice assistant

There are several reasons why Python might be a good choice for creating a voice assistant:

- Python is a widely used, general-purpose programming language with a large, active community of developers. This makes it easier to find resources, documentation, and support when working with Python.
- Python has a number of libraries and frameworks specifically designed for working with audio and speech data. For example, the **speech_recognition** library makes it easy to process audio input and perform speech recognition, and the **pyttsx3** library can be used to generate speech output.
- Python is a dynamically-typed, interpreted language, which means it can be easier to write and debug compared to compiled languages. This can make it more suitable for rapid prototyping and iterative development.
- Python has a large ecosystem of third-party libraries and frameworks that can be used to build additional functionality into your voice assistant, such as natural language processing, machine learning, and text-to-speech synthesis.

2.2 HARDWARE REQUIREMENTS

The hardware requirements for creating a voice assistant will depend on the specific capabilities and features you want to include in your assistant. Here are a few hardware considerations to keep in mind:

- **Operating system:** You will need a computer with an operating system (e.g. Windows, macOS, Linux) to run the software tools and libraries required to build your voice assistant.
- **Processor and memory:** Depending on the complexity and scale of your voice assistant, you may need a relatively powerful processor and sufficient memory to support the processing and storage requirements of your assistant.
- **Audio input and output:** To process audio input and generate speech output, you will need a microphone and speakers (or a headset with a built-in microphone and earphones). If you want to build a stand alone device (e.g. a smart speaker), you may also need additional hardware components such as a processor, memory, and connectivity options (e.g. Wi-Fi, Bluetooth).
- **Internet connectivity:** Depending on the features you want to include, you may need an internet connection to access online resources or services (e.g. natural language processing APIs, cloud storage, etc.).
- **Additional hardware:** Depending on the capabilities you want to include, you may need additional hardware components such as a camera, touch screen, or other sensors. For example, if you want to build a voice assistant that can recognize objects in photos or videos, you may need a camera and image recognition software.

2.3 SOFTWARE REQUIREMENTS

IDE: JUPYTER NOTEBOOK

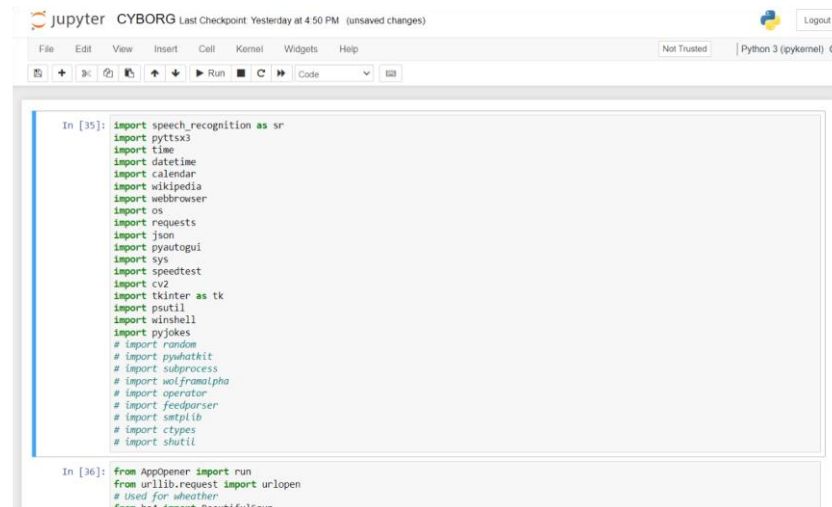


Fig 2.3.1: IDE Jupyter Notebook

The IDE used in this project is Jupyter Notebook. All the python files were created in Jupyter Notebook and all the necessary packages were easily installable in this IDE. It is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality. The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

CHAPTER 3

IMPLEMENTATION AND CODING

3.1 LANGUAGE

LANGUAGE: PYTHON



Python is a high-level programming language created by Guido Van Rossum fondly known as Benevolent Dictator for Life. Python was first released in 1991. Today Python interpreters are available for many operating systems including Windows and Linux.

- **Free:** Python is free to use and distribute and is supported by community. Python interpreter is available for every major platform.
- **Simplicity and readability:** Python has a simple, easy-to-learn syntax that emphasizes readability and reduces the cost of program maintenance.
- **Large standard library:** Python comes with a large standard library that supports many common programming tasks, such as connecting to web servers, reading and writing files, and working with data.
- **Third-party libraries:** There is a large ecosystem of third-party libraries and frameworks available for Python, including libraries for scientific computing, machine learning, and natural language processing. This makes it easy to add additional functionality to your Python programs.
- **Cross-platform:** Python is available on a wide range of platforms, including Windows, macOS, Linux, and many others. This makes it a good choice for developing programs that need to run on multiple platforms.
- **Strong community:** Python has a large, active community of developers, which makes it easier to find support and resources when you're working with the language.

3.2 LIBRARIES

Speech Recognition - Speech recognition is a machine's ability to listen to spoken words and identify them. Speech recognition is a technology that allows computers to transcribe spoken words into written text or to execute commands based on spoken input.

Pytttsx3 - pyttsx3 is a Python library that allows you to convert text to speech. It is a cross-platform library that can be used to generate synthesized speech on a wide range of platforms, including Windows, Linux, and macOS.

Datetime - The datetime module in Python provides classes for working with dates, times, and durations. It is useful for tasks such as:

1. Displaying the current date and time
2. Measuring the duration of an event
3. Parsing and formatting dates and times
4. Performing arithmetic with dates and times (e.g., finding the difference between two dates)
5. Converting between time zones

Calendar - The calendar module in Python provides functions for working with calendars and calendar-related data. It is useful for tasks such as:

1. Displaying a calendar for a given month or year
2. Finding the day of the week for a given date
3. Calculating the number of days in a month or year
4. Determining whether a year is a leap year

Wikipedia - The Wikipedia module is a Python library that allows you to access the Wikipedia API and scrape data from Wikipedia pages. The Wikipedia module is not a standard library module in Python, so it needs to be installed separately. It can be useful for building applications that need to access or work with data from Wikipedia. For example, you might use the Wikipedia module to build a tool that searches Wikipedia for information on a given topic, or to build a bot that automatically updates Wikipedia pages with new data.

Webbrowser - The webbrowser module in Python is used to open webpages in your default web browser. It provides a high-level interface to various web browsers, such as Google Chrome, Mozilla Firefox, and Safari, so you can easily open webpages from your Python scripts.

You can also use the webbrowser module to open webpages in specific web browsers.

For example: `webbrowser.get('firefox').open('http://google.com')`

OS - The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc. You first need to import the os module to interact with the underlying operating system.

Requests - Make a request to a web page, and print the response text. The requests module is a popular Python library for making HTTP requests. It allows you to send HTTP requests using Python, and get the HTTP response from the server.

Json - The json module is a built-in Python module for working with JSON data. JSON (JavaScript Object Notation) is a popular data format for storing structured data. It is commonly used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).

Pyautogui - The pyautogui module is a Python library for automating mouse and keyboard actions. It allows you to write scripts that can automatically interact with the GUI of your computer, such as clicking buttons, typing text, and moving the mouse.

SYS - The sys module is a built-in Python module that provides functions and variables related to the Python interpreter. It can be used to access command-line arguments, modify the interpreter's behaviour, and interact with the operating system.

Cv2 - Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand. The cv2 module is a Python wrapper for the OpenCV (Open Computer Vision) library. OpenCV is a popular library for computer vision tasks, such as object detection, face recognition, and image processing.

Tkinter - Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Import the Tkinter module. The tkinter module provides a number of classes and functions for creating and manipulating GUI widgets, such as buttons, labels, and text boxes. It also provides support for event handling, layout management, and drawing graphics.

Psutil - Psutil is a Python cross-platform library used to access system details and process utilities. It is used to keep track of various resources utilization in the system. Usage of resources like CPU, memory, disks, network, sensors can be monitored.

Winshell - The winshell module is a Python library for working with the Windows shell. It provides a number of functions for interacting with the Windows operating system, such as creating and deleting files and directories, starting programs, and managing the clipboard.

Pyjokes – The pyjokes module is not included in the Python standard library, so you will need to install it before you can use it in your Python scripts. You can install the pyjokes module using pip, the Python package manager.

3.3 CODING

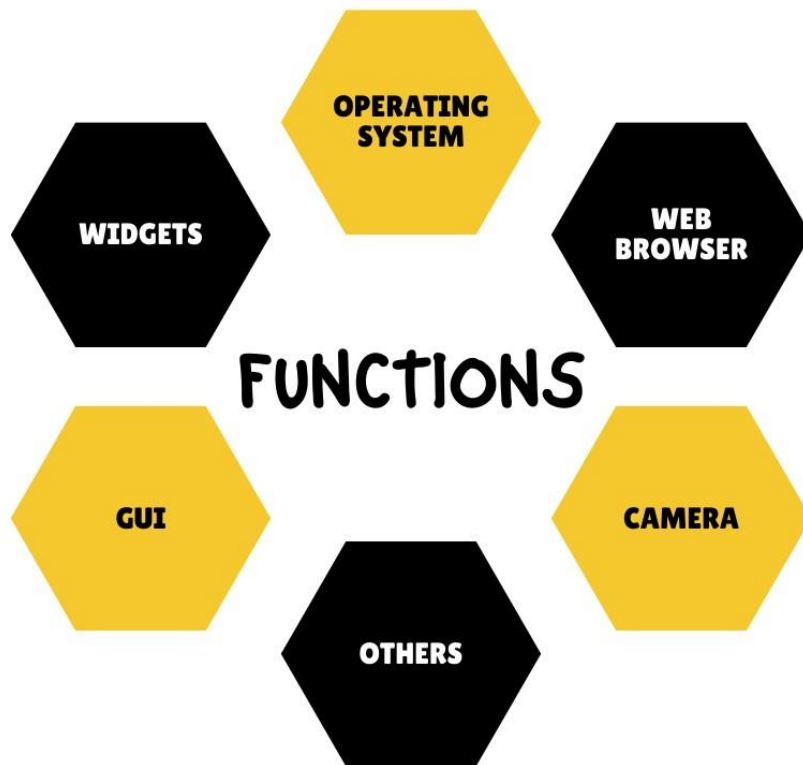


Fig 3.3.1 Categories of functions

This whole project is divided into six categories :-

- **Widgets:** Functions related to date, time, location, weather comes under this category.
- **Operating system:** Functions related to system power, operating system (shutdown, restart, sleep), system volume, screenshot, recycle bin comes under this category.
- **Web browser:** Functions related to open chrome, open google/search on google / Wikipedia, news, social media app opener, internet speed comes under this category.
- **Camera:** Functions related to open camera, photo capture, video recorder, close camera comes under this category.
- **Others:** Functions related to email sender, sms sender, set alarm, music player, personal questions come under this category.
- **GUI:** To interact with the user is an interface which is created with the help of qt5 designer.

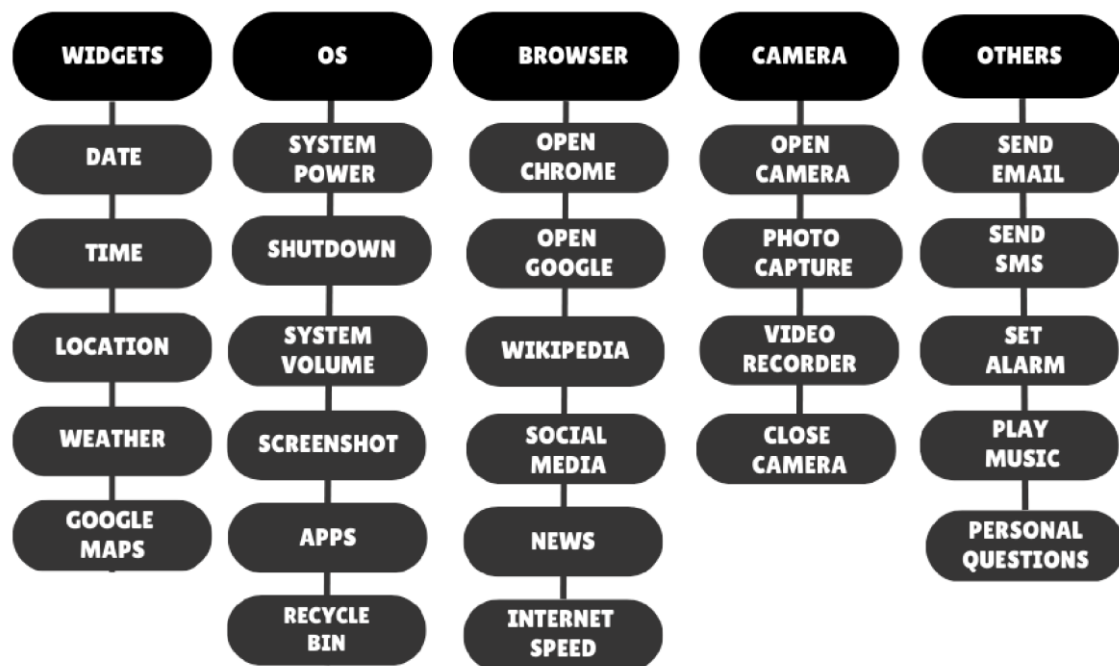


Fig 3.3.2 List of functions

SOURCE CODE

LIBRARIES

```
import speech_recognition as sr
import pytsx3
import time
import datetime
import calendar
import wikipedia
import webbrowser
import os
import requests
import json
import pyautogui
import sys
import tkinter as tk
import psutil
import winshell
import pyjokes
import openai
import cv2
import numpy as np
import math
from AppOpener import run
from urllib.request import urlopen
# Used for wheather
from bs4 import BeautifulSoup
# Used to take screenshot
from tkinter.filedialog import *
```

VOICE

```
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
# engine.setProperty('rate', 250)
# print(voices[0].id)
# print(voices[1].id)
```

SPEAK FUNCTION

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




ENGINE LISTEN

```
def engine_listen():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source)
        r.dynamic_energy_threshold = True
        r.pause_threshold = 1
        r.operation_timeout = 1
        print('\n')
        print("Listening...")
        audio = r.listen(source)
    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language = "en-US").lower()
        print(f"User: {query}")
        return query
    except:
        print("Please repeat that one")
        engine_listen()
```

WISH FUNCTION

```
def wish_master():
    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 Cypher. Sir, how can I help you")
```


FUNCTIONS RELATED TO WIDGETS

<p>DATE</p> <pre>def date(): now = datetime.datetime.now() date = datetime.datetime.today() day = calendar.day_name[date.weekday()] date = now.day month = now.month year = now.year months = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December",] ordinals = ["1st", "2nd", "3rd", "4th", "5th", "6th", "7th", "8th", "9th", "10th", "11th", "12th", "13th", "14th", "15th", "16th", "17th", "18th", "19th", "20th", "21st", "22nd", "23rd", "24th", "25th", "26th", "27th", "28th", "29th", "30th", "31st",] print(f"Cypher: {day}-{ordinals[date - 1]}-{months[month - 1]}-{year}") speak("The current date is") speak(day) speak(ordinals[date - 1]) speak(months[month - 1]) speak(year)</pre>	
<p>TIME</p> <pre>def time(): time = datetime.datetime.now().strftime("%H:%M:%S") speak("The current time is\n") print(f"Cypher: {time}") speak(time)</pre>	
<p>LOCATION FINDER</p> <pre>def location_finder(): url = "http://ipinfo.io/json" response = urlopen(url) data = json.load(response) print(f"Cypher: Sir, you are currently in {data['city']} city and country {data['country']}") speak(f'Sir, you are currently in {data['city']} city and country {data['country']}')</pre>	

<p>WEATHER FORECAST</p> <pre> def weather_forecast(): url = "http://ipinfo.io/json" response = urlopen(url) data = json.load(response) city = data["city"] # city = "Delhi" # city = "New York" # creating url and requests instance url = "https://www.google.com/search?q="+ "weather"+city html = requests.get(url).content # getting raw data soup = BeautifulSoup(html, 'html.parser') temp = soup.find('div', attrs={'class': 'BNeawe iBp4i AP7Wnd'}).text str = soup.find('div', attrs={'class': 'BNeawe tAd8D AP7Wnd'}).text # formatting data data = str.split('\n') time = data[0] sky = data[1] # printing all data print(f"Cypher: Sir you are currently in {city}") speak(f"Sir you are currently in {city}") print(f"Cypher: Temperature is {temp}") speak(f"Here Temperature is {temp}") print(f"Cypher: Sky Description is {sky}") speak(f"And Sky Description: is {sky}") </pre>	
<p>GOOGLE MAPS</p>	

FUNCTIONS RELATED TO WEBBROWSER

<p>CHROME OPENER</p> <pre>def open_chrome(): chrome_path = "CHROME PATH" os.startfile(chrome_path)</pre>	
<p>GOOGLE SEARCH</p> <pre>def open_google(): chrome_path = "CHROME PATH" tabUrl = "http://google.com/search?q=" print("Cypher: Sir, what should i search on google ?") speak("Sir, what should i search on google ?") search = engine_listen() webbrowser.get(chrome_path).open(tabUrl+search)</pre>	
<p>WIKIPEDIA SEARCH</p> <pre>def search_in_wikipedia(): query = engine_listen() speak('Searching Wikipedia...') query = query.replace("wikipedia", "") results = wikipedia.summary(query, sentences=2) print("Cypher: According to Wikipedia\n") speak("According to Wikipedia") print(f"Cypher: {results}") speak(results)</pre>	
<p>NEWS</p> <pre>def news(): api_key = "API KEY" speak("Sir please select a country from the list given below") country = ["India", "United States", "Japan", "China", "United Kingdom", "australia"] for i in range(len(country)): print(country[i]) query = str(engine_listen())</pre>	



<pre> if "india" in query: main_url = "https://newsapi.org/v2/top- headlines?country=in&category=business&apiKey="+api_ke y elif "united states" in query or "u s" in query: main_url = "https://newsapi.org/v2/top- headlines?country=us&category=business&apiKey="+api_ke y elif "japan" in query: main_url = "https://newsapi.org/v2/top- headlines?country=jp&category=business&apiKey="+api_ke y elif "china" in query: main_url = "https://newsapi.org/v2/top- headlines?country=cn&category=business&apiKey="+api_ke y elif "united kingdom" in query or "u k" in query: main_url = "https://newsapi.org/v2/top- headlines?country=uk&category=business&apiKey="+api_ke y elif "australia" in query: main_url = "https://newsapi.org/v2/top- headlines?country=au&category=business&apiKey="+api_ke y else: chrome_path = "C:/Program Files/Google/Chrome/Application/chrome.exe %s" tabUrl = "http://google.com/search?q=" search = "top 10 treding news from around the world" webbrowser.get(chrome_path).open(tabUrl+search) news = requests.get(main_url).json() article = news["articles"] news_article = [] for arti in article: news_article.append(arti["title"]) for i in range(10): print(i+1,news_article[i]) </pre>	
---	--

SOCIAL MEDIA APP OPENER

```
def social_media_app_opener():
    query = engine_listen()
    chrome_path = "CHROME PATH"
    if query is not None:
        if 'whatsapp' in query:
            speak("Opening Whatsapp")
            run("whatsapp")



webbrowser.get(chrome_path).open("web.whatsapp.com")
elif 'youtube' in query:
    speak("Opening Youtube")
    run("youtube")
    webbrowser.get(chrome_path).open("youtube.com")
elif 'facebook' in query:
    speak("Opening Facebook")
    run("facebook")
    webbrowser.get(chrome_path).open("facebook.com")
elif 'instagram' in query:
    speak("Opening Instagram")
    run("instagram")
    webbrowser.get(chrome_path).open("instagram.com")
elif 'twitter' in query:
    speak("Opening Twitter")
    run("twitter")
    webbrowser.get(chrome_path).open("twitter.com")
elif 'snapchat' in query:
    speak("Opening Snapchat")
    run("snapchat")
    webbrowser.get(chrome_path).open("snapchat.com")
elif 'linkedin' in query:
    speak("Opening LinkedIn")
    run("linkedin")
    webbrowser.get(chrome_path).open("linkedin.com")
elif 'telegram' in query:
    speak("Opening Telegram")
    run("telegram")
webbrowser.get(chrome_path).open("web.telegram.org")
elif 'reddit' in query:
    speak("Opening Reddit")
    run("reddit")
    webbrowser.get(chrome_path).open("reddit.com")
elif 'discord' in query:
    speak("Opening Discord")
    run("discord")
    webbrowser.get(chrome_path).open("discord.com")
elif 'gmail' in query or 'email' in query:
```





<pre> speak("Opening Email") run("email") webbrowser.get(chrome_path).open("mail.google.com") </pre>	
<p>INTERNET SPEED</p> <pre> def internet_speed(): # Make a request to a popular website start = time.time() r = requests.get('http://www.google.com') end = time.time() elapsed = end - start # Calculate the download speed in megabits per second download_speed = len(r.content) / elapsed / (1024 * 1024) # Make a request to a file hosting service start = time.time() r = requests.post('http://example.com/upload', data=b'x' * (1024 * 1024)) end = time.time() elapsed = end - start # Calculate the upload speed in megabits per second upload_speed = len(r.content) / elapsed / (1024 * 1024) # Measure the internet speed download_speed, upload_speed = measure_internet_speed() # Print the speeds print(f'Cypher: Download speed: {download_speed:.2f} Mbps') speak(f'Download speed: {download_speed:.2f} M b p s') print(f'Cypher: Upload speed: {upload_speed:.2f} Mbps') speak(f'Upload speed: {upload_speed:.2f} M b p s') </pre>	
<p>CHATGPT API</p> <pre> def chatgpt(): openai.api_key = 'API KEY' model_engine = 'text-davinci-003' query = engine_listen() prompt = str(query) completion = openai.Completion.create(engine = model_engine, prompt = prompt, max_tokens = 1024, </pre>	

<pre> n = 1, stop = None, temperature = 0.5,) response = completion.choices[0].text print(response) speak(response) </pre>	
---	--

FUNCTIONS RELATED TO OPERATING SYSTEM

<p>SYSTEM POWER</p> <pre> def system_power(): battery = psutil.sensors_battery() percentage = battery.percent print(f"Cypher: Sir we have {percentage} percent battery") speak(f'Sir we have {percentage} percent battery') if percentage >=75: speak("We have enough power to continue our work.") elif percentage >=40 and percentage <75: speak("We should connect our system to the charger.") elif percentage >=15 and percentage <40: speak("We don't have enough power to work please connect to the charger.") elif percentage <=15: speak("We don't have enough power to run this system .Please connect to the charger. The system will shut down very soon.") </pre>	
<p>OPERATING SYSTEM</p> <pre> def operating_system(): if "shutdown" in query: print("Sir, Do you wish to shutdown your computer say yes or no ?") speak("Sir, Do you wish to shutdown your computer say yes or no ?") if "yes" in query: speak("Shutting down your computer") os.system("shutdown /s /t 1") else: exit() elif "restart" in query: os.system("shutdown /r /t 1") elif "sleep" in query: os.system("shutdown -l") </pre>	

<p>SCREEN SHOT</p> <pre> def screenshot(): root = tk.Tk() window = tk.Canvas(root, width = 300, height = 50) window.pack() def take_ss(): screen_shot = pyautogui.screenshot() save_path = asksaveasfilename() screen_shot.save(save_path+".jpg") speak("Sir, dont forget to close the window after saving your file") button = tk.Button(root,text="Screenshot", font=('Aerial 11 bold'), background="black", foreground="white", command=take_ss) button.pack(pady=20) window.create_window(150,25,window=button) root.mainloop() </pre>	
<p>OPERATING SYSTEM APPS</p> <pre> def operating_system_app(): if "calculator" in query: speak("Opening Calculator") run("calculator") elif "excel" in query: speak("Opening Excel") run("excel") elif "vlc" in query: speak("Opening vlc media player") run("vlc media player") elif "notepad" in query: speak("Opening Notepad") run("notepad") elif "word" in query: speak("Opening word") run("word") elif "task manager" in query: speak("Opening Task manager") run("task manager") elif "command prompt" in query: speak("Opening command prompt") run("command prompt") elif "paint" in query: speak("Opening Paint") run("paint") </pre>	

EMPTY RECYCLE BIN

```
def empty_recycle_bin():  
    winshell.recycle_bin().empty(  
        confirm = True, show_progress = False, sound = True  
    )  
    speak("Deleted the files that are there in the recycle bin")
```



FUNCTIONS RELATED TO CAMERA

PHOTO AND VIDEO RECORDING

```
def camera():
    print("Cypher: Opening camera")
    speak("Opening camera")
    print("Cypher: Press space bar to take a picture")
    speak("Press space bar to take a picture")
    print("Cypher: Press r to record the video")
    speak("Press r to record the video")
    print("Cypher: Sir press q to close the camera")
    speak("Sir press q to close the camera")
    camera = cv2.VideoCapture(0)
    image_counter = 0

    camera.set(cv2.CAP_PROP_FRAME_WIDTH, 720)
    camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

    fourcc = cv2.VideoWriter_fourcc('m','p','4','v')
    writer = cv2.VideoWriter("recording.mp4", fourcc, 30.0,
(720,480))
    recording = False

    while True:
        ret, frame = camera.read()
        cv2.imshow('Camera', frame)

        if ret:
            if recording:
                writer.write(frame)

        key = cv2.waitKey(1)

        if key == 32:
            image_name =
"image_{}.png".format(image_counter)
            cv2.imwrite(image_name , frame)
            image_counter+=1

        elif key == ord("r"):
            recording = not recording
            print("Cypher: Recording has been started")
            speak("Recording has been started")

        elif key == ord("q"):
            break
```



<pre>camera.release() writer.release() cv2.destroyAllWindows() print("Cypher: Saving your file") speak("Saving your file") print("Cypher: Closing camera") speak("Closing camera")</pre>	
--	--

OTHERS

```
# def email_sender():  
# def music_player():  
# def set_alarm():
```

ALL THE TASKS THAT THIS ASSISTANT CAN PERFORM

```
def cypher():
    wish_master()
    while True:
        query = engine_listen()
        if query is not None:
            if "date" in query and "time" in query:
                date()
                time()
            elif "date" in query:
                date()
            elif "time" in query:
                time()
            elif "location" in query or "where are we" in query or "where we are" in query:
                location_finder()
            elif "weather" in query or "climate" in query or "temperature" in query:
                weather_forecast()
            elif "chrome" in query or "open chrome" in query:
                open_chrome()
            elif "search" in query or "google" in query or "open google" in query:
                open_google()
            elif "wikipedia" in query:
                search_in_wikipedia()
            elif "news" in query:
                news()
            elif "internet speed" in query:
                internet_speed()
            elif "open" in query:
                if "whatsapp" in query or "youtube" in query or "facebook" in query or "instagram" in query or "telegram" in query or "twitter" in query or "snapchat" in query or "linkedin" in query or "reddit" in query or "discord" in query or "email" in query or "gmail" in query:
                    social_media_app_opener()
                elif "calculator" in query or "excel" in query or "vlc" in query or "notepad" in query or "word" in query or "task manager" in query or "command prompt" in query or "paint" in query:
                    operating_system_app()
            elif "battery" in query or "power" in query:
                system_power()
            elif "shutdown" in query or "restart" in query or "sleep" in query:
                operating_system()
            elif "volume" in query or "mute" in query or "unmute" in query:
                if "up" in query or "increase" in query:
                    pyautogui.press("volumeup")
                elif "down" in query or "decrease" in query:
                    pyautogui.press("volumedown")
                elif "mute" in query:
                    pyautogui.press("volumemute")
```

```

        elif "unmute" in query:
            pyautogui.press("volumeunmute")
        elif "screenshot" in query:
            screenshot()
            speak("Thanks for closing the window")
        elif "recycle bin" in query:
            empty_recycle_bin()
        elif "camera" in query or "open camera" in query or "close camera" in
query or "photo" in query or "capture" in query or "picture" in query or "record" in
query or "video record" in query:
            camera()
        elif "open email" in query or "email" in query or "send email" in query or
"mail" in query or "gmail" in query:
            email_sender()
        elif "open spotify" in query or "play music" in query or "music" in query or
"songs" in query:
            music_player()
        elif "joke" in query or "jokes" in query:
            jokes()
        elif "who are you" in query or "what can you do" in query or "define
yourself" in query or "your name" in query:
            speak("Hi sir my name is cyborg and i am a virtual sidekick")
        elif "created you" in query or "made you" in query or "build you" in query:
            speak("I am a computer program built by a human being")
        elif "your purpose" in query or "why do you exist" in query:
            speak("To take over this world and to wipe out the entire human
civilization")
        elif "how are you" in query:
            speak("Sir i am good. what about you")
        elif "fine" in query or "good" in query:
            speak("It's good to know that you are fine")
        else:
            chatgpt()

```

```

if __name__ == "__main__":
    while True:
        permission = engine_listen()
        if permission is not None:
            if "wake up" in permission or "hey" in permission or "cypher" in
permission or "hello" in permission:
                cypher()
            elif "goodbye" in permission or "sleep" in permission or "bye" in
permission or "stop" in permission:
                print("Good bye sir, have a nice day .")
                speak("Good bye sir, have a nice day .")
                sys.exit()

```

CHAPTER 4

TESTING AND TEST RESULT

4.1 SOFTWARE TESTING AND OBJECTIVE OF TESTING

The system testing is done on fully integrated system to check whether the requirements are matching or not. The system testing for **Voice Assistant** desktop assistant focuses on the following four parameters.

FUNCTIONALITY

In this we check the functionality of the system whether the system performs the task which it was intended to do. To check the functionality each function was checked and run, if it is able to execute the required task correctly then the system passes in that particular functionality test. For example to check whether voice assistant can record a video using the system camera. As we can see in the figure 5.1 .

User: cyborg can you record a video

Cyborg: Press r to record the video

Cyborg: Press q to close the camera

User: pressed r

Cyborg: Recording has been started

User: pressed q

Cyborg: Saving your file

Cyborg: Closing camera

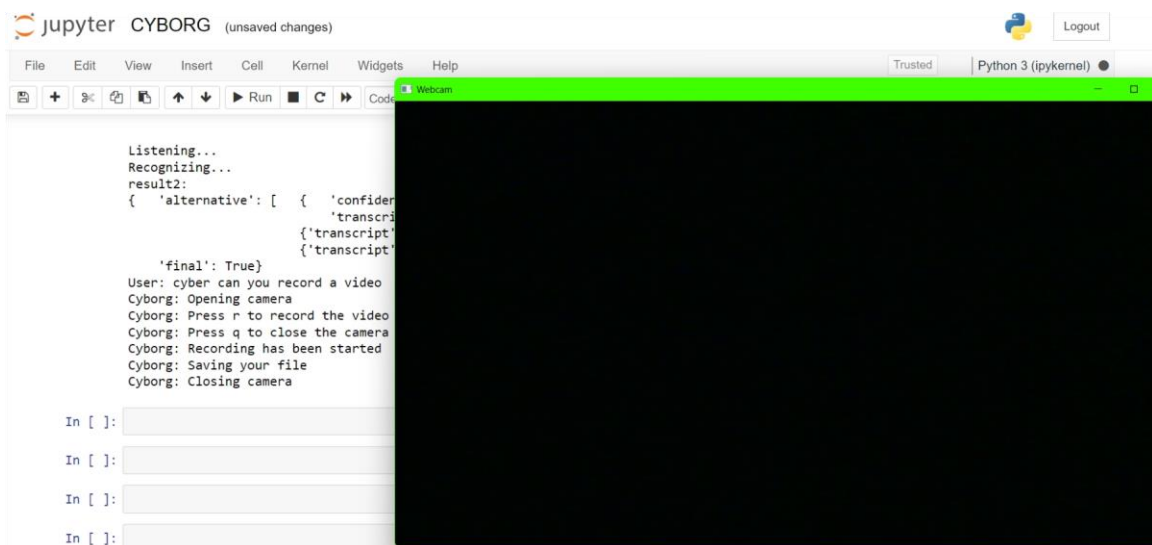


Fig 4.1.1: Video recorder input/output screen

USABILITY

Usability of a system is checked by measuring the easiness of the software and how user friendly it is for the user to use, how it responds to each query that is being asked by the user. It makes it easier to complete any task as it automatically do it by using the essential module or libraries of Python, in a conversational interaction way. Hence any user when instruct any task to it, they feel like giving task to a human assistant because of the conversational interaction for giving input and getting the desired output in the form of task done. The desktop assistant is reactive which means it know human language very well and understand the context that is provided by the user and gives

response in the same way, i.e. human understandable language, English. So user finds its reaction in an informed and smart way. The main application of it can be its multitasking ability. It can ask for continuous instruction one after other until the user “QUIT” it. It asks for the instruction and listen the response that is given by user without needing any trigger phase and then only executes the task.

SECURITY

The security testing mainly focuses on vulnerabilities and risks. As voice assistant is a local desktop application, hence there is no risk of data breaching through remote access. The software is dedicated to a specific system so when the user logs in, it will be activated.

STABILITY

Stability of a system depends upon the output of the system, if the output is bounded and specific to the bounded input then the system is said to be stable. If the system works on all the poles of functionality then it is stable.

4.2 SAMPLE TEST DATA / OUTPUT SCREEN

WIKIPEDIA SEARCH

```
Listening...
Recognizing...
result2:
{  'alternative': [  {  'confidence': 0.88687539,
                        'transcript': 'Elon Musk in Wikipedia'},
                      {'transcript': 'alone mask in Wikipedia'}],
  'final': True}
User: elon musk in wikipedia
Cyborg: According to Wikipedia

Cyborg: Elon Reeve Musk ( EE-lon; born June 28, 1971) is a business magnate and investor. He is the founder, CEO and chief engineer of SpaceX; angel investor, CEO and product architect of Tesla, Inc.; owner and CEO of Twitter, Inc.; founder of The Boring Company; co-founder of Neuralink and OpenAI; and president of the philanthropic Musk Foundation.
```

Fig 4.2.1: Wikipedia search input and output screen

GOOGLE SEARCH

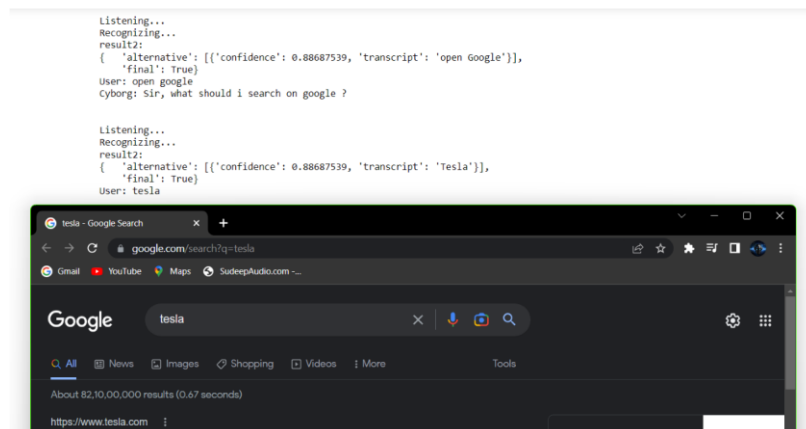


Fig 4.2.2: Google search input and output screen

WEATHER FORECAST

```
Listening...
Recognizing...
result2:
{  'alternative': [  {  'confidence': 0.88687527,
                        'transcript': 'can you tell me the current '
                        'temperature'}],
  'final': True}
User: can you tell me the current temperature
Cyborg: Sir you are currently in Delhi
Cyborg: Temperature is 18°C
Cyborg: Sky Description is Smoke
```

Fig 4.2.3: Weather forecast input and output screen

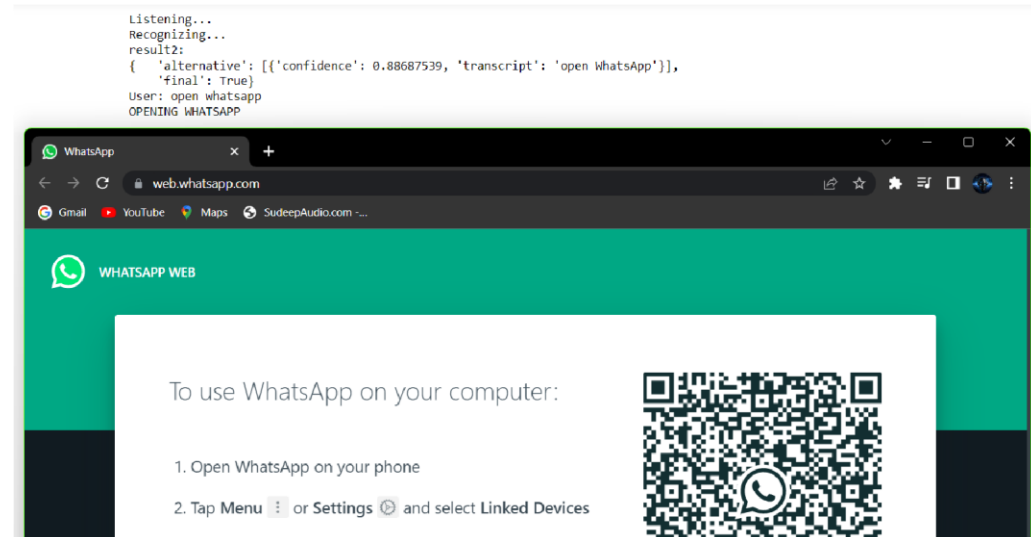
NEWS

```
Listening...
Recognizing...
result2:
{ 'alternative': [ { 'confidence': 0.88687527,
                    'transcript': 'can you tell me the current news'}],
  'final': True}
User: can you tell me the current news
India
United States
Japan
China
United Kingdom
australia

Listening...
Recognizing...
result2:
{ 'alternative': [{ 'confidence': 0.88687539, 'transcript': 'India'}],
  'final': True}
User: india
1 Aristo Bio-Tech IPO GMP, Date, Price, Review, Allotment - IPO Watch
2 Sponsored: Mahindra Thar RWD: The Ideal Everyday SUV - carandbike
3 Auto Expo 2023: Honda XRE 300 flex-fuel capable bike shown - Autocar India
4 IT Q3 Results LIVE: Wipro to report earnings today; Infosys raises, HCL Tech trims revenue guidance - Zee Business
5 Maruti Shows Off Fully Accessorised Jimmy At Auto Expo 2023 - CarDekho
6 Centrum Wealth's co-heads asked to quit due to 'loss of confidence' | Mint - Mint
7 Zerodha's Nithin Kamath: It has been a painful bull market for active traders - Moneycontrol
8 Jet Airways relaunch back on track as NCLT allows transfer of ownership to Jalan-Kalrock Consortium - Moneycontrol
9 Auto Expo 2023: SIAM hosts 2nd Edition of the Global Electrification Mobility Summit - The Financial Express
10 India alleges France's Pernod violated Delhi city rules to boost market share - Economic Times
```

Fig 4.2.4: News input and output screen

SOCIAL MEDIA APP/WEB OPENER



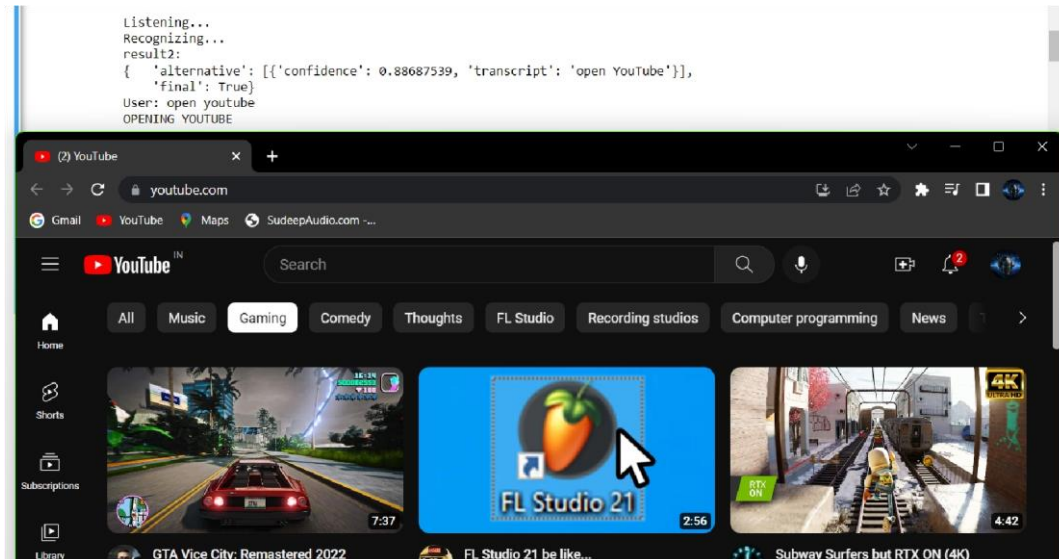


Fig 4.2.5: Social media app/web input and output screen

OPEN CAMERA

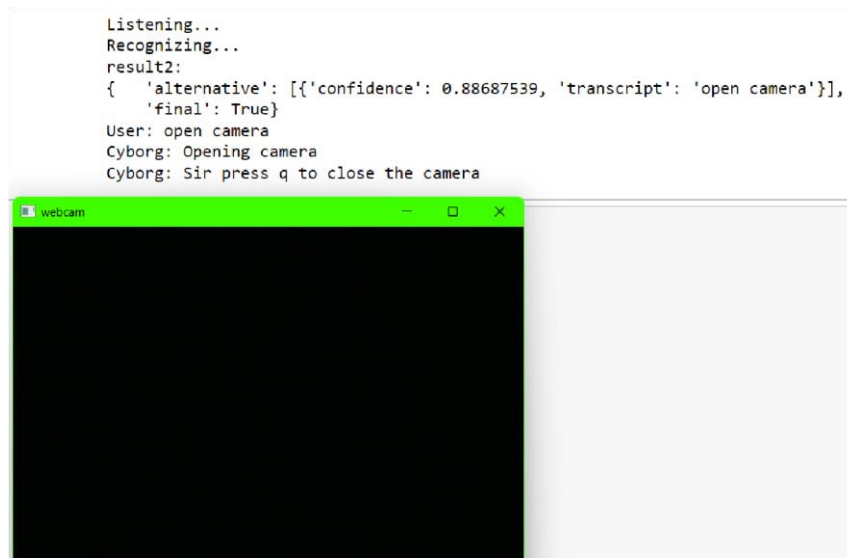


Fig 4.2.6: Open camera input and output screen

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

This paper presents a comprehensive overview of the design and development of a Voice enabled personal voice assistant for pc using python programming language. This voice enabled personal assistant, in today's life style will be more effective in case of saving time, compared to that of previous days. We have successfully created a working virtual assistant which can be which can be activated by the user using the wake keyword "CYBORG" and can perform various tasks based on the user query or commands. In future, we hope to incorporate more Artificial intelligence in to this project, such as Machine learning, Neural networks, Natural language processing, and so on, as well as Internet of things. With the addition of these elements, we will be able to improve this virtual voice assistant by adding more functionality to it.

5.2 FUTURE SCOPE

- The future scope of voice assistants is quite broad and continues to evolve. Some potential areas of growth include:
- Increased integration with smart home devices: Voice assistants are becoming increasingly integrated with smart home devices such as lights, thermostats, and security systems. This allows users to control these devices using their voice, making their homes more convenient and efficient.
- Greater personalization and customization: Voice assistants are becoming more personalized, with the ability to learn users' preferences and habits over time. This will allow them to provide more relevant and useful information and recommendations.
- More advanced natural language processing: Voice assistants are becoming more sophisticated in their ability to understand and respond to natural language. This will make them more accurate and useful for a wide range of tasks.
- Increased use in businesses and industries: Voice assistants are being used in a variety of industries, such as healthcare, retail, and banking, to automate customer service and other tasks.
- Advancements in AI and machine learning will make them more intelligent, accurate, and efficient.
- Voice commands can be encrypted to maintain security.
- Android app can also be developed.

5.3 LIMITATION OF THE SYSTEM

- Limited understanding of context and meaning: Voice assistants may have difficulty understanding the context and meaning of certain phrases or questions, leading to inaccuracies or misunderstandings.
- Limited functionality in noisy environments: Voice assistants may have difficulty understanding commands in noisy environments, such as crowded public spaces or busy offices.
- Limited ability to understand different accents or dialects: Voice assistants may have difficulty understanding users with thick accents or those speaking in dialects.
- Limited ability to recognize and respond to multiple users: Some voice assistants may have difficulty recognizing and responding to multiple users, leading to confusion and inaccuracies.
- Privacy and security concerns: Personal voice assistants often collect and store large amounts of personal data, which can raise concerns about privacy and security.
- Dependence on internet connection: Some voice assistants require an internet connection to work, which can be limiting in certain situations.
- Some people may be uncomfortable with the idea of having an always-on listening device in their home.

REFERENCES

www.geeksforgeeks.com

www.stackoverflow.com

www.python.org

<https://pypi.org/project/opencv-python/>

<https://opencv.org/>

<https://platform.openai.com/docs/overview>