Desktop Assistant

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE & ENGINEERING

Submitted to:
Snigdha Luthra

Submitted By:
Student Group
NAME: Kishan Kumar
UID: 19BCS1421
NAME: Arun Gosain
UID: 19BCS1413
NAME: Vansh Gupta
UID: 19BCS1509

Mentor Signature
(Sanjeev Kumar & e8752)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Chandigarh University, Gharuan

June 2021

ACKNOWLEDGEMENT

This project report on a project titled VIRTUAL ASSISTANT, I had to take the help and guideline of a few respected people, who deserve my greatest gratitude.

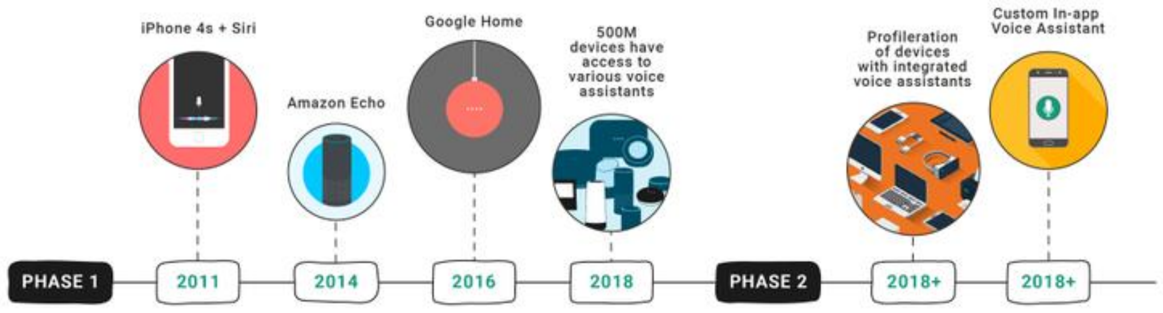
We express our deep sense of gratitude to our respected and learned guide, **Er. Sanjeev Kumar** for their valuable help and guidance. We are thankful to her for the encouragement. We are also thankful to all the other faculty & staff members of our department for the kind cooperation and help.

Many people, especially my classmates and friends themselves, have made valuable comments and suggestions on this project which gave me inspiration to improve my project. Here I thank all the people for their help directly and indirectly to complete this project report.

TABLE OF CONTENTS

ABSTRACT

With the increasing workload in day-to-day life, the traditional method to keep track of all the tasks manually was an inconvenience. We see every day some new technological inventions and innovations are taking place to provide more ease to humans to automate manual tasks. We have seen self-driving cars, industrial robots being used for manufacturing in factories, calculators to solve complex mathematical calculations, chatbots on online sites to interact with the customer. Our Project is Personal Desktop Assistant which is designed to easily automate computer tasks with the voice command only. The purpose of this application is to increase the productivity of the user.


iPhone 4s + Siri
Amazon Echo
Google Home
500M devices have access to various voice assistants
Profileration of devices with integrated voice assistants
Custom In-app Voice Assistant
PHASE 1
2011
2014
2016
2018
PHASE 2
2018+
2018+

INTRODUCTION

What is Desktop Voice Assistant?

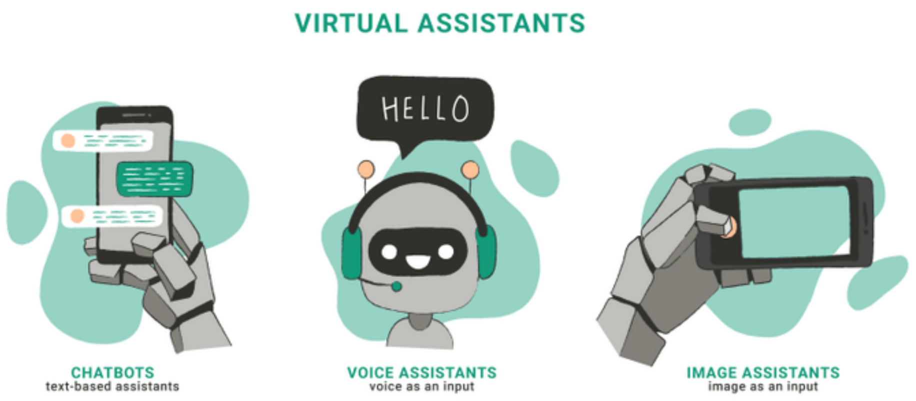
A Desktop Voice Assistant is software that works on a speech recognition system. It takes voice command and produces the output as per command, it is capable of performing various tasks or services for an individual based on voice commands. It helps ease our day-to-day tasks, such as showing weather reports, creating reminders, making shopping lists, making phone calls, etc.

Personal assistant software can also improve user productivity by managing routine tasks of the user and by providing information from online sources to the user. The most popular virtual personal assistant we all use commonly nowadays include Ok Google, Siri, Cortana, and Alexa.

It provides users a wealth of knowledge at their fingertips and can automate many time-consuming activities with just a sentence or two. Voice assistants can deliver an experience that includes tone of voice the main ingredient in building effective communication and emotional connection. Voice assistant breaks down barriers for people with disabilities, whether sensory, physical, or cognitive. Voice interaction is especially useful for those with visual, impairments. Voice assistants can increase productivity as one can easily create meetings reminders, and to-do items, make phone calls/send texts, and can get a quick overview of your day.

Features of NEMO (The Desktop Voice Assistant):

- It can send emails on your behalf.
- It can play music and movies for you.
- It can perform Wikipedia searches.
- It is capable of opening websites like Google, YouTube, etc. with just one voice command.
- It can take a screenshot of the screen.
- It can tell the current time to the user.
- It can start or stop screen recording.
- It can open applications like Chrome, Notepad, and Games.



CHARACTERISTICS OF THE PROJECT

- **Convenience:** Provides users a wealth of knowledge at their fingertips and can automate many time-consuming activities with just a sentence or two.
- **Accessible and Inclusive:** Voice assistant breaks down barriers for people with disabilities, whether sensory, physical, or cognitive. Voice interaction is especially useful for those with visual impairments.
- **More Human Touch:** Voice assistants can deliver an experience that includes tone of voice the main ingredient in building effective communication and emotional connection.
- **Increase Productivity:** Voice assistants can increase productivity as one can easily create meetings reminders, and to-do items, make phone calls/send texts, and can get a quick overview of your day.
- **Enjoyment:** People – particularly younger people – genuinely enjoy speaking to home assistants, showing that a human-to-machine bond can be created through voice.

TECHNOLOGY USED

- For creating the desktop voice assistant software we have used the Python programming language.
- Python helps in the easy writing and execution of codes. Also, it is a robust, highly useful language focused on rapid application development (RAD).
- Python is highly flexible and is an open-source programming language.
- It is also a portable language. So, we can run the same code on any platform such as Linux, Unix, Windows, and Mac.
- Provides a library that contains built-in modules that provide access to system functionality such as file I/O that otherwise would be inaccessible.
- Provides standard GUI library Tkinter for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications.

METHODOLOGY/ MODULES USED

To create the desktop voice assistant following Python modules are used:

- **Pyttsx:** *Pyttsx* stands for Python Text to Speech. It is a cross-platform Python wrapper for text-to-speech synthesis. It is a Python package supporting common text-to-speech engines on Mac OS X, Windows, and Linux. It works for both Python2.x and 3.x versions. Its main advantage is that it works offline.

- **Speech Recognition:** This is a library for performing speech recognition, with support for several engines and APIs, online and offline. It supports APIs like Google Cloud Speech API, IBM Speech to Text, Microsoft Bing Voice Recognition etc.

- **Webbrowser:** The *webbrowser* module provides a high-level interface to allow displaying Web-based documents to users. Under most circumstances, simply calling the *open()* function from this module will do the right thing.

- **Os:** The *OS* module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.

- **Pyautogui:** *Pyautogui* is a library that allows our Python scripts control the mouse and keyboard to automate interactions with other applications.

- **Datetime:** *Datetime* module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals.

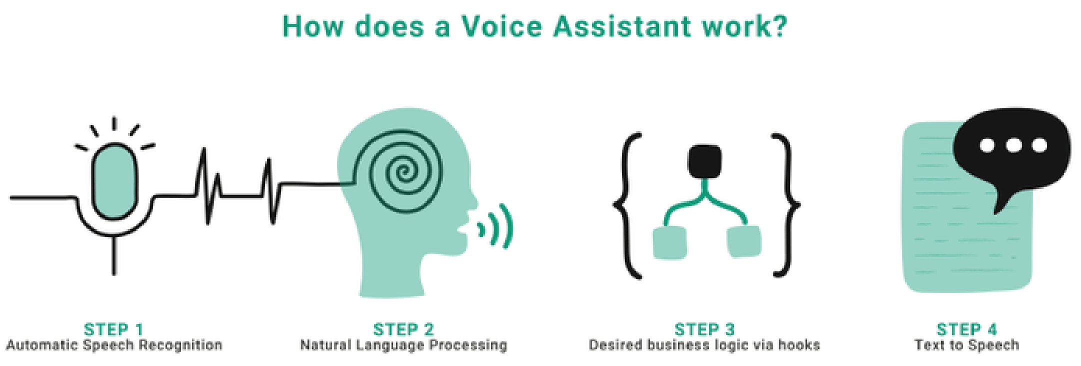
- **Wikipedia:** *Wikipedia* is a Python library that makes it easy to access and parse data from Wikipedia. Search Wikipedia, get article summaries, get data like links and images from a page, and more.

- **PIL:** *Python Imaging Library (PIL),* is one of the important modules for image processing in Python.

- **Random:** The *random* module is used to generate the pseudo-random variables. It can be used to *perform some action randomly* such as selecting a random elements from a list

WORKING OF DESKTOP VOICE ASSISTANT

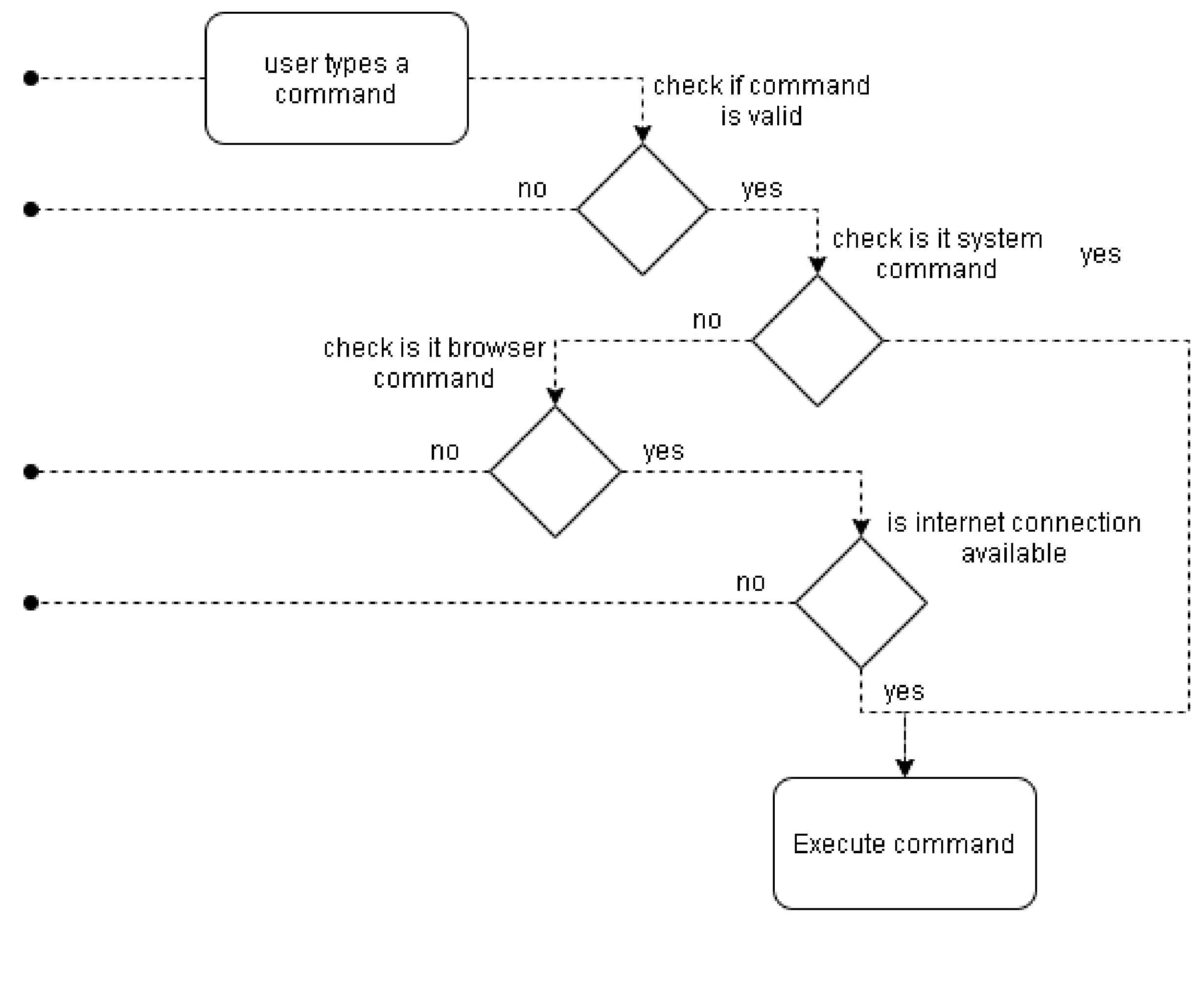
- **Speech-to-Text:** It takes voice command from user and then convert Speech-to-Text using pyttsx3 module.
- **Text Analysis:** Converted text is just a sequence of letters for computer. By using the conditional statements, program search for keywords that might our in that text.
- **Fetch result:** Based on a keyword that might occur in the text converted from speech, it fetches the command and provides the most suitable result.
- **Text-to-Speech:** After fetching the result it provides the user an acknowledgment which is usually converted from Text-to-Speech so that acknowledgment delivered to the user includes tone of voice to give a human touch.


How does a Voice Assistant work?
STEP 1
Automatic Speech Recognition
STEP 2
Natural Language Processing
STEP 3
Desired business logic via hooks
STEP 4
Text to Speech

PROJECT MODELLING AND DESIGINING

FLOWCHART

A flowchart is simply a graphical representation of steps. It shows steps in sequential order and is widely used in presenting the flow of algorithms, workflow or processes.

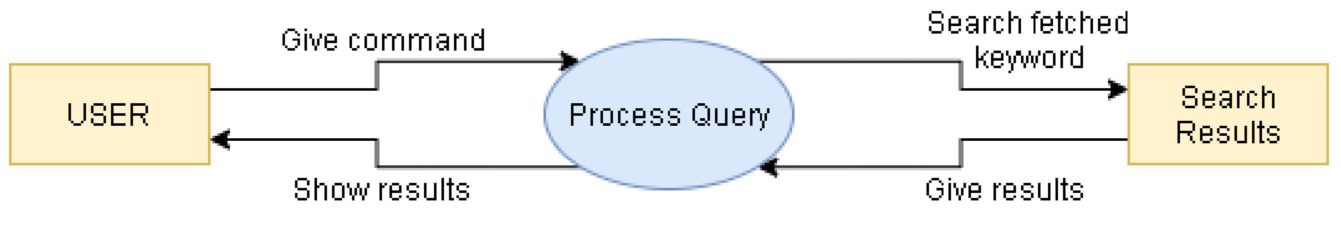

user types a command
check if command is valid
no
yes
check is it system command
yes
no
check is it browser command
no
yes
is internet connection available
no
yes
Execute command

FLOWCHART FOR DESKTOP ASSISTANT

DFD

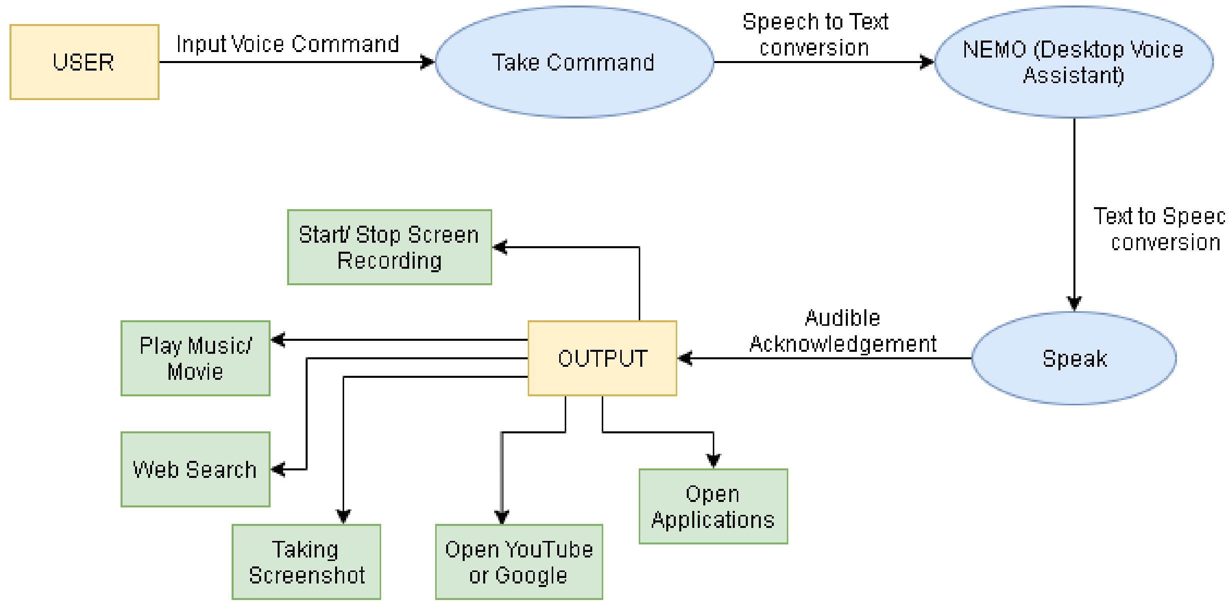
The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself.

LEVEL 0 DFD:



DFD DIAGRAM OF DESKTOP ASSISTANT

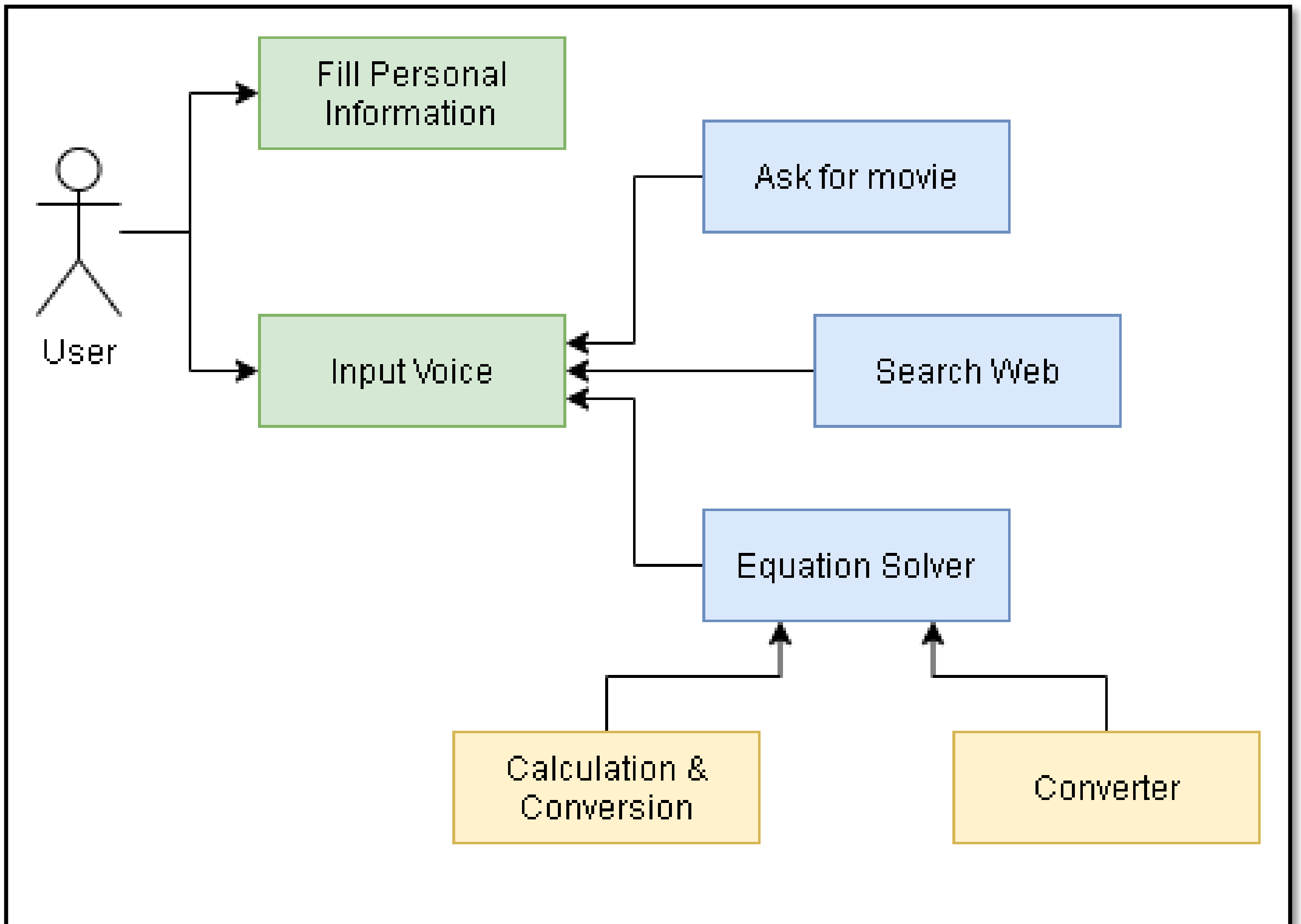
LEVEL 1 DFD:



LEVEL 1 DFD OF NEMO (DESKTOP VOICE ASSISTANT)

UML

UML stands for *Unified Modeling Language*, is a way to visually represent the architecture, design, and implementation of complex software systems. It is intended to provide a standard way to visualize the design of a system.



UML DIAGRAM FOR DESKTOP ASSISTANT

CODE IMPLEMENTATION

```python
import pyttsx3
import speech_recognition as sr
import datetime
import webbrowser
import os
import random
from PIL import ImageGrab
import pyautogui
import wikipedia

engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')  # getting details of current voice
engine.setProperty('voice', voices[1].id)

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

def wishme():
    hour = int(datetime.datetime.now().hour)
    if hour >= 0 and hour < 12:
        p = "Good morning!"
    elif hour >= 12 and hour < 18:
        p = "Good afternoon!"
    else:
        p = "Good evening!"

    p = p + " I am Nemo. Please tell me how I may help you."
    print(f"Nemo : {p}")
    speak(p)



def takeCommand():
    # It takes microphone input from the user and returns string output

    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 0.7
        audio = r.listen(source)
    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language='en-in')  # Using google
for voice recognition.
        print(f"User : {query}\n")  # User query will be printed.

    except Exception as e:
        # print(e)
        print("Say that again please...")  # Say that again will be printed
in case of improper voice
        return "None"  # None string will be returned

    return query

if __name__ == "__main__":

    wishme()
    while True:
        query = takeCommand().lower()

        if 'search' in query:
            try:
                speak('Searching Wikipedia..')
                query = query.replace("search", "")
                results = wikipedia.summary(query, sentences=2)
                speak("According to Wikipedia")
                print(f"Nemo : According to Wikipedia, {results}")
                speak(results)
            except Exception as e:
                p = "Sorry no results found"
                print(f"Nemo : {p}")
                speak(p)

        elif 'open youtube' in query:
```

```python
            webbrowser.open("Youtube.com")

        elif 'open google' in query:
            webbrowser.open("Google.com")

        elif 'time' in query:
            strTime = datetime.datetime.now().strftime("%H:%M")
            p = f"The time is {strTime}"
            print(f"Nemo : {p}")
            speak(p)

        elif 'play music' in query:
            music_dir = 'C:\\Users\\gosai\\Music'
            songs = os.listdir(music_dir)
            i = random.randint(0, len(songs) - 1)
            os.startfile(os.path.join(music_dir, songs[i]))

        elif 'play movie' in query:
            movie_dir = 'C:\\Users\\gosai\\Videos\\Movies'
            movie = os.listdir(movie_dir)
            i = random.randint(0, len(movie) - 1)
            os.startfile(os.path.join(movie_dir, movie[i]))

        elif 'screenshot' in query:
            image = ImageGrab.grab()
            image.show()

        elif 'record screen' in query:
            pyautogui.hotkey('win', 'alt', 'r')

        elif 'stop screen' in query:
            pyautogui.hotkey('win', 'alt', 'r')

        elif 'open chrome' in query:
            path = r'C:\Program Files
(x86)\Google\Chrome\Application\chrome.exe'
            os.startfile(os.path.join(path))
        elif 'open game' in query:
            path = r'C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Riot Games\VALORANT.lnk'
            os.startfile(os.path.join(path))

        elif 'open notepad' in query:
            path = r'C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Accessories\Notepad.lnk'
            os.startfile(os.path.join(path))
```

ADVANTAGES

- **Increase Productivity:** Voice assistants can increase productivity as one can easily create meetings reminders, and to-do items, make phone calls/send texts, and can get a quick overview of your day.

- **Aids Hand-free Operation**: Voice talking gives consumers hands-free access to many functions because you only need the voice to activate them. So it makes it easier and faster to do certain things.

- **Helpful for differently abled:** Voice assistant breaks down barriers for people with disabilities, whether sensory, physical, or cognitive. Voice interaction is especially useful for those with visual impairments.

- **Instant Knowledge:** You can get instant brief knowledge about any topic. It can provide you instant result fetched from Wikipedia to save your time.

DISADVANTAGES

- **Listening problem:** VPA faces a problem in processing wrong pronounced words and miscellaneous words.
- **Silent mode support:** VPA gives a response to voice input thus it doesn't work properly in silent mode.
- **Navigation languages:** Most of VPAs generally understand the only English language.
- **Internet access:** VPA needs an active internet connection to give the desired output.

CONCLUSION

We may conclude by saying that the task of creating a desktop voice assistant from scratch using Python and its modules helped us a lot in understanding the implementation of Python. We chose Python for the development of desktop voice assistant as it provides modules that make it easy to interact with the system functionality, also Python provides a module for GUI development of the application. Our Desktop Voice Assistant named *"NEMO"* will enable the user to perform or automate various tasks by just giving voice commands. For example, the user will be able to perform a web search, open applications, play music, and many more tasks. Talking about GUI we have tried to make it user-friendly and as simple as possible to understand.

BIBLIOGRAPHY

Website Reference:

- *https://www.w3schools.com/python/*
- *https://en.wikipedia.org/wiki/Python_(programming_language)*
- *https://www.coursera.org/specializations/data-science-python*
- *https://www.geeksforgeeks.org/personal-voice-assistant-in-python/*

Book Reference:

- *Python Programming – Kiran Gurbani*
- *Learning Python – Mark Lutz*

YouTube Channel Reference:

- *CS Dojo*
- *edureka!*