

## **SMART VOICE ASSISTANT**

*A project submitted in partial fulfillment of the  
requirements for the award of the degree of*

### **Bachelor of Technology in INFORMATION TECHNOLOGY**



Submitted by:  
**Dheeraj**  
**Roll No.:11912020**  
**Group No- 214**

Supervised by:  
**Dr. Mukesh Mann**  
Assistant Professor

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,  
SONEPAT -131201, HARYANA, INDIA**

## ACKNOWLEDGEMENTS

First of all, I am immensely indebted to Almighty God for his blessings and grace without which I could not have undertaken this task and my efforts would never have been a success.

I humbly consider it a privilege and honor to express my heartiest and profound gratitude to Prof **(Dr) Mukesh Mann**, Assistant Professor -IT, and IIIT Sonepat, for his appropriate direction, valuable suggestion, under judging assistance so generously extended to me.

This guidance and support received from my entire classmates who contributed and who are contributing to this project, is vital for the success of this project. I am grateful for their constant support and help.

I also owe a sense of gratitude to my parents for encouragement and support throughout the project.

Dheeraj

## **SELF DECLARATION**

I hereby declare that the work contained in the project titled “**SMART VOICE ASSISTANT**” is original. I have followed the standards of project ethics to the best of my abilities. I have acknowledged all sources of information that I have used in the project.

Name: Dheeraj

Roll No.: 11912020

Department of Information Technology,  
Indian Institute of Information Technology,  
Sonepat-131201, Haryana, India.

## CERTIFICATE

This is to certify that **Dheeraj** has worked on the project entitled “**SMART VOICE ASSISTANT**” under my supervision and guidance.

The contents of the project, being submitted to the **Information Technology, IIIT Sonepat**, for the award of the degree of **B.Tech in Information Technology**, are original and have been carried out by the candidate himself. This project has not been submitted in full or part for the award of any other degree or diploma to this or any other university.

Dr. Mukesh Mann  
Supervisor

Department of Information Technology,  
Indian Institute of Information Technology,  
Sonepat-131201, Haryana, India

## **ABSTRACT**

**Name of the student - Dheeraj, Roll No.11912020, Degree for which submitted  
B.Tech (IT), Department of Information Technology, IIIT, Sonepat.**

**Project Title: SMART VOICE ASSISTANT**

**Name of the thesis supervisor: Dr. Mukesh Mann**

**Month and year of the project submission: September 2020**

The project aims to develop a personal voice-assistant for Windows-based systems. SMART VOICE ASSISTANT draws its inspiration from virtual assistants like Cortana for Windows, and Siri for iOS. As there is already Cortana present for Windows but it is not that much powerful. It can only perform small tasks, Whereas our SMART VOICE ASSISTANT has been designed to provide a user-friendly interface for carrying out a variety of tasks by employing certain well-defined commands.

Users can interact with the assistant either through voice commands or using keyboard input. As a personal assistant, SMART VOICE ASSISTANT assists the end-user with day-to-day activities like a general human conversation, searching queries in google, searching for videos, retrieving images, live weather conditions, word meanings, searching for medicine details, health recommendations based on symptoms, and reminding the user about the scheduled events and tasks.

## LIST OF ABBREVIATIONS

---

## LIST OF FIGURES

<b>Figure's No.</b>	<b>Name of Figures</b>	<b>Page Nos.</b>
Fig 1.1	THE SASHIMI MODEL	04
Fig 2.1	GENERAL STRUCTURE OF SVA	07
Fig 2.2	DATA FLOW SEQUENCE	08
Fig 2.3	CLASS DIAGRAM	09
Fig 2.4	USE CASE DIAGRAM	10
Fig 2.5	STAR UML	11
Fig 2.6	PYTHON	11
Fig 2.7	PYCHARM	12
Fig 2.8	GITHUB	12
Fig 3.1	USING WHATSAPP WITH HELP OF SVA	15
Fig 3.2	OUTPUT OF WHATSAPP COMMAND	15
Fig 3.3	PLAYING YOUTUBE VIDEO WITH HELP OF SVA	16
Fig 3.4	OUTPUT OF SVA PLAYING YOUTUBE VIDEO	16
Fig 3.5	SENDING A MAIL WITH HELP OF SVA	17
Fig 3.6	SVA SENDING A MAIL	17
Fig 3.7	WRITING NOTES IN MS WORD USING SVA	18
Fig 3.8	SVA WRITING NOTES	18
Fig 3.9	SVA ANSWERING BASIC MATHEMATICS CALCULATION	19
Fig 3.10	SVA RESPONDING TO BASIC QUERIES	19
Fig 4.1	BLACK-BOX TESTING	21
Fig 4.2	WHITE-BOX TESTING	22

## **LIST OF TABLES**

## Table of Contents

---

Acknowledgments.....	i
Self Declaration.....	ii
Certificate.....	iii
Abstract.....	iv
List of Abbreviations.....	v
List of Figures.....	vi-vii
List of Tables.....	viii-ix
List of Publications.....	x

TABLE OF CONTENTS		
<b>CHAPTER 1</b>	<b>INTRODUCTION</b>	<b>01-06</b>
	1.1	Introduction.....
	1.2	Problem Outline.....
	1.3	Project Objectives.....
	1.4	Project Methodology .....
	1.5	Scope of Project Work.....
	1.6	Organization of project.....
	1.7	Summary.....
<b>CHAPTER 2</b>	<b>DESIGN</b>	<b>07-13</b>
	2.1	Introduction.....
	2.2	General Components of SVA.....
		2.2.1 Front-End Interfaces .....
		2.2.2 End Users .....

## Table of Contents

		<b>2.2.3 Voice Recognition System.....</b>	<b>07</b>
	2.3	<b>System Architecture.....</b>	<b>08</b>
	2.4	<b>Data Flow Sequence.....</b>	<b>08</b>
	2.5	<b>Class Diagram.....</b>	<b>09</b>
	2.6	<b>Use Case Diagram.....</b>	<b>10</b>
	2.8	<b>Development Platform Tools .....</b>	<b>10</b>
	2.9	<b>Summary.....</b>	<b>12</b>
<b>CHAPTER 3</b>	<b>IMPLEMENTATION</b>		<b>14-19</b>
	3.1	<b>Introduction.....</b>	<b>14</b>
	3.2	<b>List of Libraries Used.....</b>	<b>14</b>
	3.3	<b>Some Hot features of SVA.....</b>	<b>15</b>
	3.4	<b>Summary.....</b>	<b>19</b>
<b>CHAPTER 4</b>	<b>SOFTWARE TESTING AND VALIDATION</b>		<b>20-25</b>
	4.1	<b>Introduction.....</b>	<b>20</b>
	4.2	<b>Various Testing Techniques.....</b>	<b>20</b>
	4.3	<b>Technique Used for SVA .....</b>	<b>25</b>
	4.4	<b>Fail Case.....</b>	<b>25</b>
	4.5	<b>Summary.....</b>	<b>25</b>
<b>CHAPTER 5</b>	<b>CONCLUSION AND FUTURE SCOPE</b>		<b>26-26</b>
	5.1	<b>Conclusions and Significance .....</b>	<b>26</b>
	5.2	<b>Limitations and Future Directions.....</b>	<b>26</b>

## Table of Contents

---

	<i>References</i> .....	27
	<i>Appendix</i> .....	28

## **CHAPTER 1**

### **INTRODUCTION**

#### **➤ INTRODUCTION**

Today everyone is doing work from home. Keeping this scenario in mind, majorly everyone is using a Windows PC. And they perform mostly major tasks every day like starting a meeting on zoom, opening notes and ppts, switching cameras in between the meetings, mute/unmute their microphones to speak while in a meeting. Even I do lunch while attending a meeting. So in that case, it would be easy for me with the help of **SVA** to unmute myself without touching my mouse and keyboard. And all of the tasks which are listed above and even many more can be performed easily with the help of SVA just by giving the voice commands to it.

With the help of **SVA**, the user will be able to access the services of Windows PC with their voice command. Users can easily send an email to any recipient. In this Project, a physically disabled person or the person having less knowledge about windows PC, how to access the Windows PC, can easily access the functionalities of pc with their voice or speech command. This application includes the functions and services such as: send an email, send messages on Whatsapp and make a video call on Whatsapp, Google any information, Wikipedia Search, Run Videos on YouTube, music player service, checking weather, Wikipedia search, camera, we can also write a note by giving a command. **SVA** and also include some system tasks like CPU information, shutdown, restart, taking screenshots, Turning On Bluetooth, Wi-Fi, Location Services, Battery Saver Airplane Mode, Hotspot, etc.

Some features of SVA are as follows:

1. **Dictate in any S/W or Website** – Yes, you can convert your speech into text in any S/W or website. You can dictate into word processing S/W like MS Word, Notepad, Wordpad, etc. This feature can be even helpful to disable and visually impaired people. It has very accurate speech recognition.
2. **Play/Search Songs and Videos** – You can play songs and videos present in your Directory or folder.
3. **Custom Voice Commands** – It is very easy to make custom voice commands in SVA. Using this feature you can create your commands and decide what should SVA do when you say something. For example, you can create a custom voice command such as open my sites which opens multiple websites that you frequently use.
4. **Teach Custom Replies** – You can teach the reply that your computer should give back to you when you speak something.
5. **Voice Music/Media Control** – You can control music and video playback by voice commands. Say Volume Up or Volume Down to change the volume.
6. **Mathematics** – You can do arithmetic calculations quickly. The S/W also supports mathematical functions such as Power and Roots, prime numbers, percentage, divisors, trigonometry, and much more.
7. **Search Online Information** – SVA can create a report on any subject by retrieving information from the Internet. It can even search for you on search

engines like Google and other online resources like Wikipedia. For example ask, “Who is Mark Zuckerberg?”

8. **Dictionary** – This S/W does much more than a paper dictionary or thesaurus. It is a speech recognition enabled dictionary with an audio reply. This means just voice command it to define a word and it will speak back the definition to you.
9. **Search Files and Folders** – Search files and folders on your computer.
10. **Open Programs, Files, and Websites** – Quickly open files and websites. You can even create custom commands for the same.
11. **Weather Information** – Get current weather condition reports for any city/town in the world.
12. To search anything on Google and Wikipedia by giving a voice command.
13. To run or open any application with the help of a voice command.
14. To do System Tasks easily by voice command like screenshot, shutdown, restart, log off, CPU and battery information, date and time information, Turning On Bluetooth, Wi-Fi, Location Services, Battery Saver Airplane Mode, Hotspot, etc.
15. To send messages, make video calls and audio calls on Whatsapp.

## ➤ PROBLEM OUTLINE -

We are all well aware of Cortana, Siri, Google Assistant, and many other virtual assistants that are designed to aid the tasks of users in Windows, Android, and iOS platforms. But Cortana is not that much powerful. Cortana can not perform all the tasks of a user. It can only do some small things. But SVA can perform almost every task for a user. A user performs majorly the same tasks every day. These tasks can be easily performed with the help of SVA just by giving voice commands to it.

## PURPOSE

This S/W aims at developing a personal assistant for Windows-based operating systems. The main purpose of the S/W is to perform the tasks of the user at certain commands, provided in either of the ways, speech or text. It will ease most of the work of the user as a complete task can be done on a single command. SMART VOICE ASSISTANT draws its inspiration from Virtual assistants like Cortana for Windows and Siri for iOS. Users can interact with the assistant either through voice commands or keyboard input.

## PROJECT GOALS -

Currently, the project aims to provide the Windows Users with a Virtual Assistant that would not only aid in their daily routine tasks like searching the web, extracting weather data, vocabulary help, and many others but also help in the automation of various activities. In the long run, we aim to develop a complete assistant

## PROJECT DESCRIPTION

As a personal assistant, SVA assists the end-user with day-to-day activities like a general human conversation, searching queries in various search engines like Google, Bing or Yahoo, searching for videos, retrieving images, live weather conditions, word meanings, searching for medicine details, health recommendations based on symptoms and reminding the user about the scheduled events and tasks.

## **PROJECT OBJECTIVES -**

### **1. TO HAVE INTERACTION BETWEEN USERS AND SVA-**

Interact with the user with a voice command to perform some operations.

### **2. TO PERFORM OPERATIONS OVER VOICE COMMANDS-**

Infinite tasks can be performed with the help of SVA. Just think of any task and it can be performed just by giving a voice command to the SVA.

### **3. TO AUTOMATE THE SVA –**

Using SVA to automate the user's everyday tasks that he/she performs on his Windows PC.

**ALL THE ABOVE LISTED OBJECTIVES WILL BE ACHIEVED BY USING SASHIMI MODEL.**

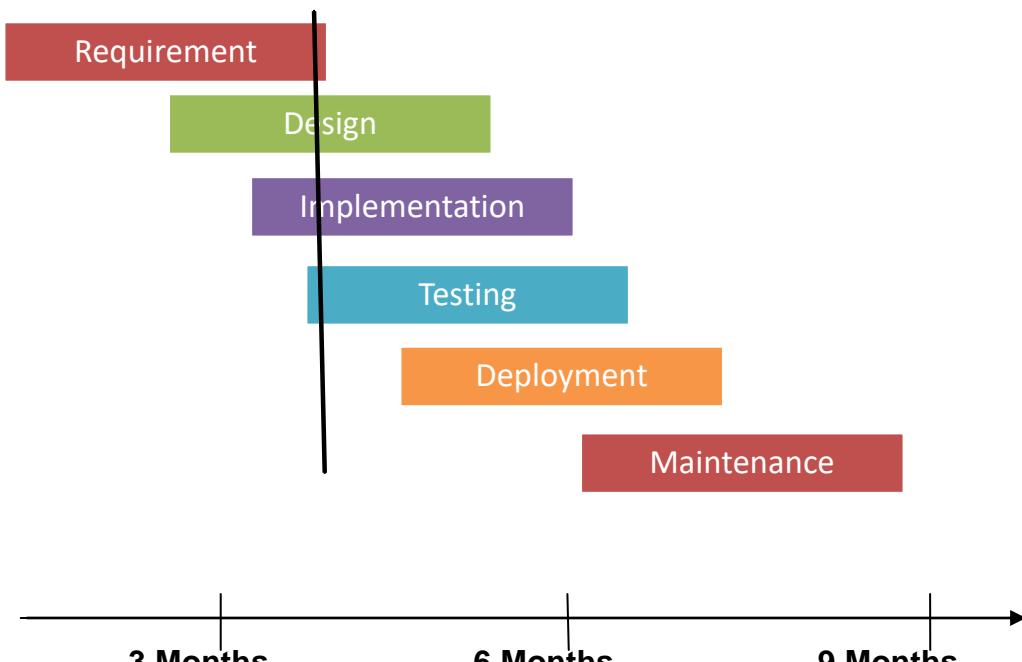
## **➤ PROJECT METHODOLOGY –**

All the objectives mentioned above are going to be achieved with the help of the **SASHIMI** model.

The first objective i.e **USER INTERACTION WITH VOICE COMMANDS**, We are going to interact with SVA with some of the commands and when it will start recognizing the voice then alternately we will start working on the next objective i.e **PERFORMING OPERATIONS OVER VOICE COMMANDS**.

When both of these objectives will get fulfilled, we will try to **AUTOMATE the SVA** that the user can set the commands according to his/her choice and can perform any specific task of his/her choice on that command. In this way, he can automate the SVA so that the major tasks that he performs every day on his system can be performed easily by using SVA.

In this way, we are going to overlap the different S/W phases over each other. All the resources for the project are available to us and we wanted the project to get started soon, moreover, we wanted to shorten our time scale that's why we used this model.

**Fig 1.1 THE SASHIMI MODEL****Advantages of using the SASHIMI Model in this project-**

- Shortens the development time, as different S/W layers are overlapped over each other.
- A 12-month project could be done in 9 months.
- People with different skills can start working on the project without waiting for the work done for the previous phase.
- The architect can start working on the project before even the analyst or who is doing the requirements are done.

**Disadvantages of using the SASHIMI Model in this project-**

- It may result in some rework because the design phase is started before all the requirements were done.
- It means that if something is found later, during the requirement phase, the design phase is then again needed to be adjusted.
- If the coding phase is already started, then that requirement change may also result in some rework.

## ➤ SCOPE OF PROJECT -

**UTILISATIONS** - Presently, SMART VOICE ASSISTANT (SVA) is being developed as an automation tool and virtual assistant. Among the various roles played by SVA are

1. Search Engine with voice interactions
2. Medical diagnosis with Medicine aid.
3. Vocabulary App to show meanings and correct spelling errors.
4. Weather Forecasting Application.

**LIMITATIONS** - Having so many advantages, everything comes with some limitations too, SVA will have the following limitations currently -

1. **SVA cannot understand two different commands having the same meaning** - Users need to set each command before using it. If somehow the user forgets or doesn't set the command before using it then SVA will not be able to perform any task based on that specific command.
2. **SVA may have a problem in recognizing the voice while having noise in the background** – SVA will face difficulty in recognizing the voice of the user if he/she speaks while having noise in the background.
3. **Unable to recognize images** – SVA can't search anything on the web with the help of an image as it can not recognize any image.
4. **Difficult to type punctuations**- SVA may face difficulty while typing punctuations like if you speak comma then instead of putting a ',' it may type it down as comma.

## ➤ THE ORGANISATION OF PROJECT -

- In Chapter 1, a brief Intro, problem outline, project objectives, methodology, the scope of SVA are discussed.
- In Chapter 2, the Study and review of literature – computational intelligence and important aspects of S/W (SVA) will be discussed.
- In Chapter 3, Optimization – the respective methodology, framework, experimental skills, performance analysis will be discussed in brief for SVA.
- In Chapter 4, Conclusion and Future Scope, limitations, significances, future directions for SVA will be discussed.

## ➤ SUMMARY –

In this chapter, A brief introduction to SVA is given. Through this SMART VOICE ASSISTANT (SVA), various services for users will be automated by using a single line command. It will ease most of the tasks of the user like searching the web, retrieving weather forecast details, vocabulary help, and medical-related queries. More features are given in the introduction part of this chapter.

After that, the problem outline is discussed with the purpose of the project, its goals, and its description which tells about why we selected this project.

Project Objectives are discussed afterward which are user interaction with voice commands, performing operations over voice commands, using the tool for automation.

Project Methodologies are also discussed to achieve the above-mentioned objectives in the future. THE SASHIMI MODEL will be used in the project methodology.

The further future scopes for SVA are also discussed in this chapter. The aim is to make this project a complete assistant and make it smart enough to automate the major tasks of the user. Its utilizations and limitations are discussed in the future scope section of this chapter.

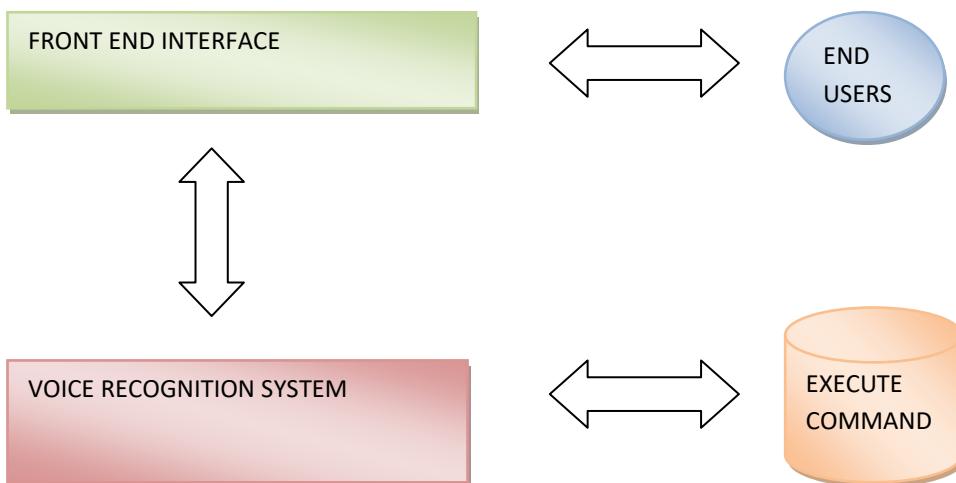
The organization of the project is discussed at the end in which a brief intro of all the chapters has been given.

**CHAPTER - 2****DESIGN****➤ Introduction –**

SVA will be used in the following three ways: firstly, a command to the computer whereas secondly, SVA records the voice and matches with the available commands. If it is available then finally a proper response is provided and proper action is taken. In this chapter, we will be discussing general components for SVA. As seen in Fig-2.1, the voice will be divide into four different parts: front-end interface, end-users, voice recognition System finally executes the command. Each section is explained as follows:

**➤ Front-End Interfaces -**

In the front-end-interface, the user will be having direct access to the interface and users will be communicating with SVA by providing Input and receiving the Output. It receives user prompt input voice and in return delivers a proper response and proper action is taken with the associated command.



**FIGURE 2.1 GENERAL STRUCTURE OF SVA**

**➤ End Users -**

End-users refers to device users. They will be using these devices for communication with the use of the application, and moreover, end-users are those who will be using this application with their devices like computer and laptop users.

**➤ Voice Recognition Systems -**

It is the heart of a Voice application system, which has the ability to understand voice input given by the user and make the application work in an efficient way. This system is an important component for the user as a gateway to use his or her voice as an input component. In a Nutshell, for clearly understanding user voice command, we should consider a voice recognition system that contains all the processes by which

the application system directs for building speech signals to text data and a few forms of the important meaning of speech.

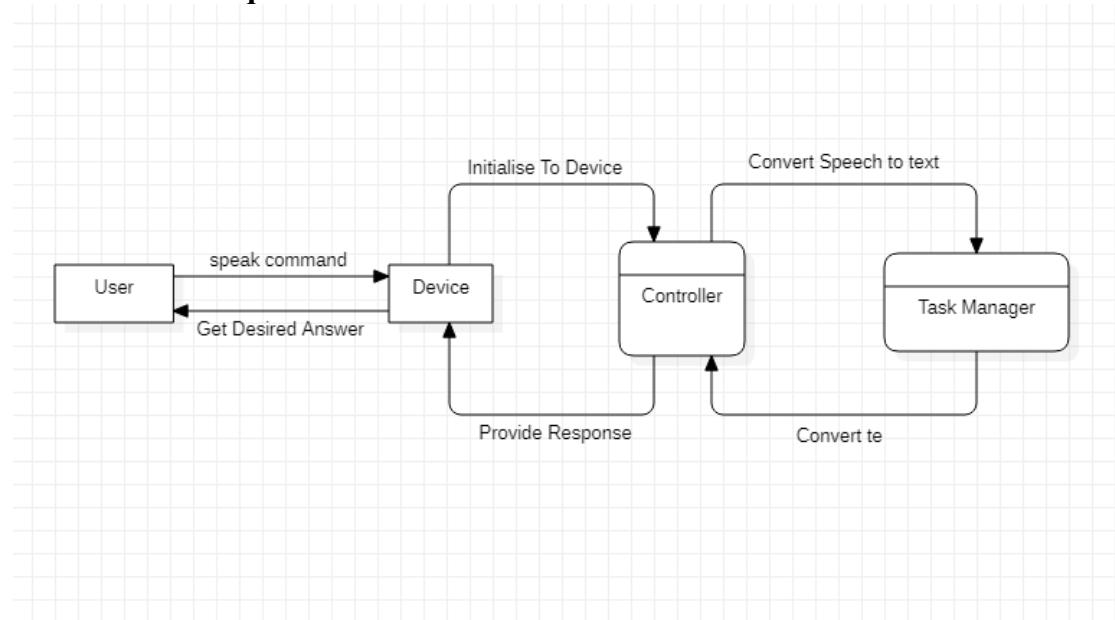
#### ➤ System Architecture -

The total design consists of these phases:

- 1) Collection of data that is in speech format.
- 2) Analyse the voice and convert it to text.
- 3) storing the data and processing it.
- 4) Speech generation from the text output that is processed.

The data that is collected in the speech form is stored and used as input for the next phase of the process. In the next phase, the input which is given in the form of voice is processed continuously and is converted into text. In the third phase, the text which is converted is analyzed by Python Script which processes it and identifies the action to be taken for the command. In the last phase, after the action to be taken is identified, the output will be obtained from text to speech conversion.

#### ➤ Data Flow Sequence -

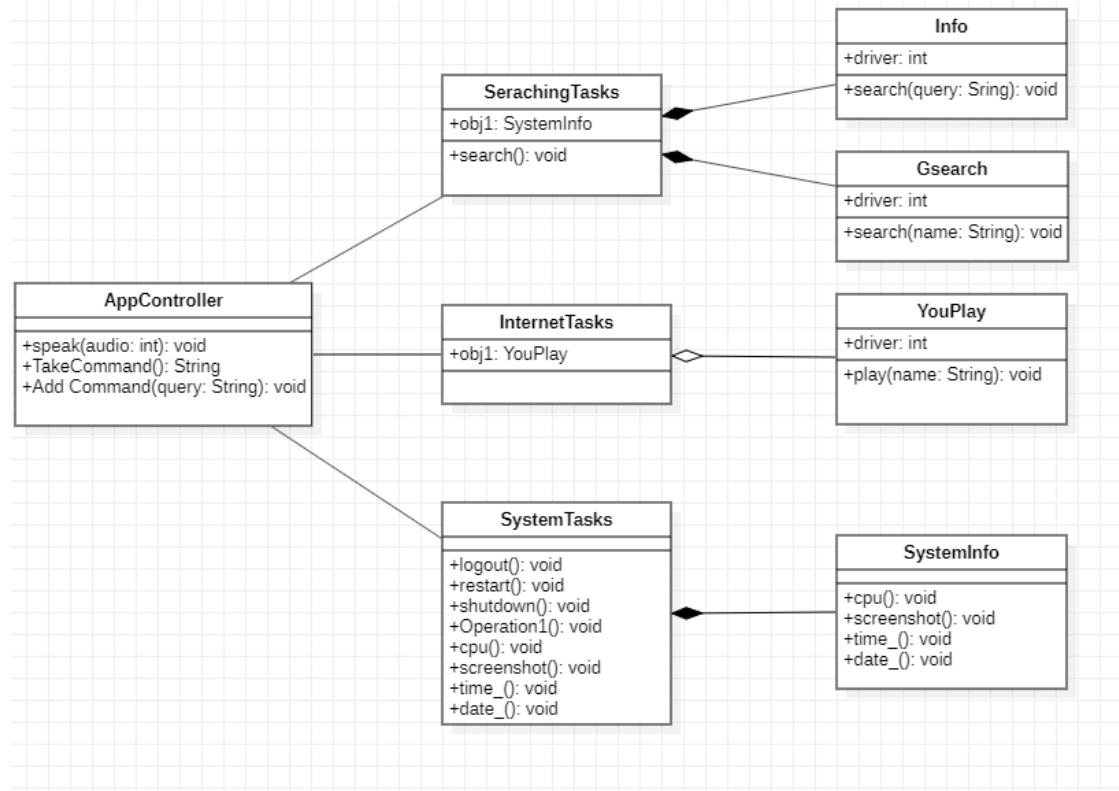


**FIG 2.2 DATA FLOW SEQUENCE OF SVA**

- **Initialize Device**:- Device initialization does whatever steps are necessary to get a system into a working state. It sets the unit in motion by calling its name. The process is specific for every device, there are no magic values that would initialize any device that you come across.
- **Task Manager**:- The Speech-to-Text and Text-to-Speech transfers are done by the task manager. It provides a semantic description of what was spoken, which is passed into a service manager.
- **Device**:- For speech output, a natural language (NLG) component and a text-to-speech (TTS) component is used.
- **Execute the command**: Run the Respective python script after you have found the match for the given order.

## ➤ CLASS DIAGRAM FOR SMART VOICE ASSISTANT-

The Class diagram of SVA consists of the main classes that are used in it.



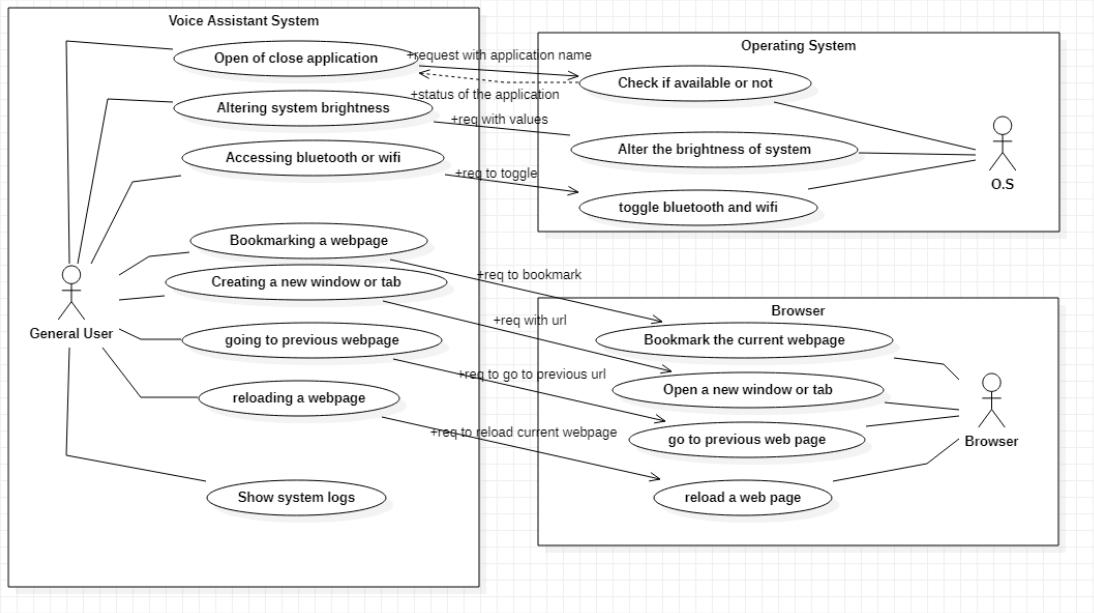
**FIG 2.3 CLASS DIAGRAM**

## ➤ MAIN CLASSES FOR SMART VOICE ASSISTANT-

The main classes of SVA are listed below.

- APP CONTROLLER CLASS – To control the apps which are being used.
- SEARCHING TASKS CLASS - Performs searching over the tasks.
- INTERNET TASKS CLASS - Performs Internet tasks.
- SYSTEM TASKS CLASS - Performs System tasks.
- INFO CLASS - Provides Info to the query.
- SEARCH CLASS - Searches any query.
- YOUPLAY CLASS - Playing any video on Youtube.
- SYSTEM INFO CLASS – Provides CPU stats, Time, date, etc.

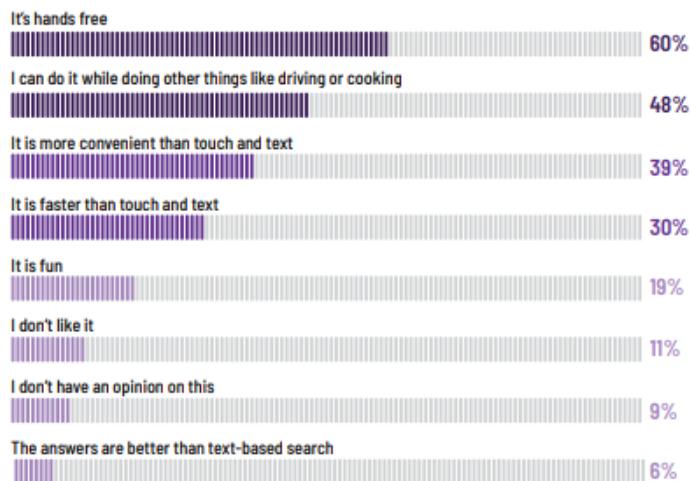
➤ USE CASE DIAGRAM FOR SMART VOICE ASSISTANT-



**FIGURE 2.4 USE CASE DIAGRAM**

➤ WHAT USERS WILL LIKE ABOUT SMART VOICE ASSISTANT-  
There are various benefits of using SVA which are mentioned below in table 2.1.

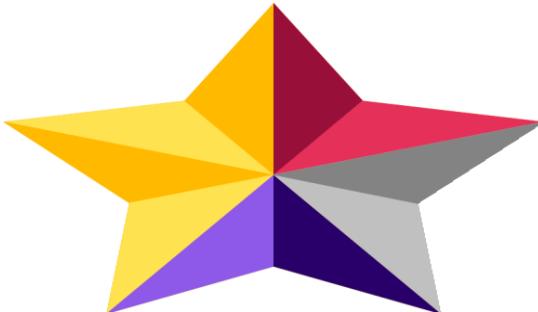
**TABLE 2.1 BENEFITS OF USING SVA**



➤ DEVELOPMENT PLATFORM TOOLS –

The following development tools are needed for the development of this project

1. Star Uml For Developing Various Diagrams related to this project.
2. Python, as the programming language.
3. PyCharm IDE for Coding purpose.
4. GitHub for collaboration purpose.

**1) STAR UML****FIG 2.5 STAR UML**

StarUML is a sophisticated software modeler for agile and concise modeling. It is compatible with various UML diagrams: Class, Object, Use Case, Component, Deployment, Composite Structure, Sequence, Communication, Statechart, Activity, Timing, Interaction Overflow, Information Flow, and Profile Diagram. It is a UML tool by MKLab. The software was licensed under a modified version of GNU GPL until 2014 when a rewritten version 2.0.0 was released for beta testing under a proprietary license.

**2) PYTHON LANGUAGE****FIG 2.6 PYTHON**

Python is an interpreted, high-level, and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

**3) PYCHARM****FIG 2.7 PYCHARM**

PyCharm is an integrated development environment used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains.

#### 4) GITHUB



**FIG 2.8 GITHUB**

GitHub, Inc. is a subsidiary of [Microsoft](#) which provides [hosting](#) for [software development](#) and [version control](#) using [Git](#). It offers the [distributed version control](#) and [source code management](#) (SCM) functionality of Git, plus its own features. It provides [access control](#) and several collaboration features such as [bug tracking](#), [feature requests](#), [task management](#), [continuous integration](#), and [wikis](#) for every project.[\[3\]](#) Headquartered in [California](#), it has been a subsidiary of [Microsoft](#) since 2018.

#### ➤ SUMMARY –

This chapter talks about the design part of the Smart Voice Assistant. The working of SVA is also discussed in this chapter.

General components of SVA i.e End Users, Front-end-interface, Voice recognition systems, command executions are discussed in brief.

This chapter also talks about the system architecture of SVA in which a brief discussion is there for the total design of the SVA.

The total design consists of these phases:

- 1) Collection of data that is in speech format
- 2) Analyse the voice and convert it to text
- 3) Storing the data and processing it
- 4) Speech generation from the text output that is processed.

Then a brief discussion of Data Flow Sequence is also there which basically tells the depth working of SVA.

In the end, the CLASS Diagram which consists of all the main classes of SVA, and the USE CASE Diagram which shows some of the use cases of the SVA are discussed.

## CHAPTER - 3

### IMPLEMENTATION

#### ➤ **INTRODUCTION -**

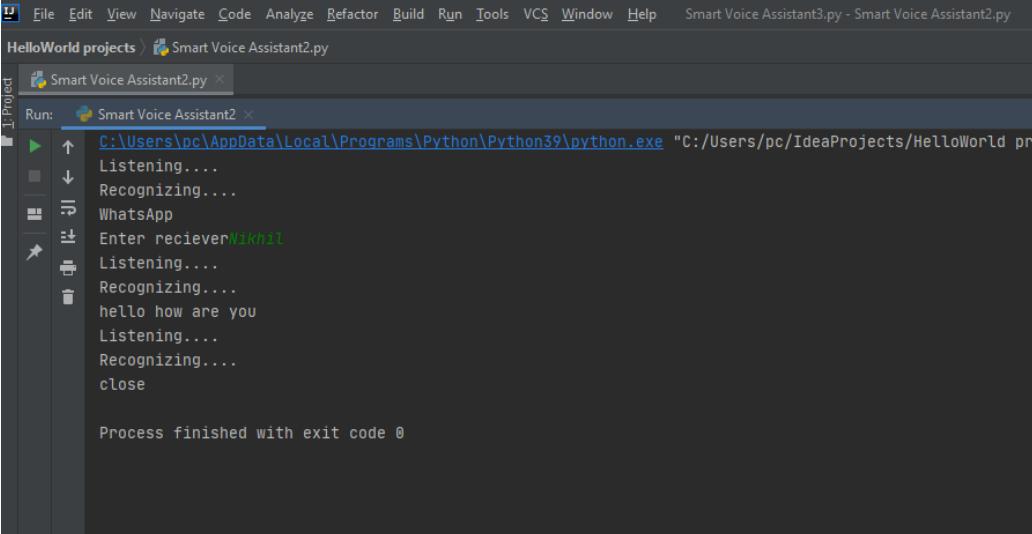
Implementation i.e how SVA is implemented is discussed in this chapter.

#### ➤ **List of Libraries Used:**

- **Pyttsx3:** pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible with both Python 2 and 3.
- **Datetime:** The datetime module supplies classes for manipulating dates and times.
- **Speech\_recognition:** Library for performing speech recognition, with support for several engines and APIs, online and offline.
- **Smtplib:** The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP
- **Webbrowser:** The webbrowser module provides a high-level interface to allow displaying Web-based documents to users.
- **Psutil:** Psutil (process and system utilities) is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors) in Python. It is useful mainly for system monitoring, profiling and limiting process resources and management of running processes.
- **Pyjokes:** One line jokes for programmers (jokes as a service)
- **Os:** This module provides a portable way of using operating system dependent functionality
- **Pyautogui:** PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse & keyboard.
- **Random:** This module implements pseudo-random number generators for various distributions.
- **Docx:** python-docx is a Python library for creating and updating Microsoft Word (.docx) files.
- **Wolframalpha:** Basic usage is pretty simple. Create the client with your App ID (request from Wolfram Alpha) used to ask queries.
- **Time:** This module provides various time-related functions.
- **Selenium:** The selenium package is used to automate web browser interaction from Python.
- **Keyboard:** Take full control of your keyboard with this small Python library. Hook global events, register hotkeys, simulate key presses and much more.

## ➤ Some Hot Features of Smart voice Assistant:-

- 1) Sending a Message on Whatsapp using Smart Voice Assistant – SVA sends a message on Whatsapp to the receiver just with help of a voice command.



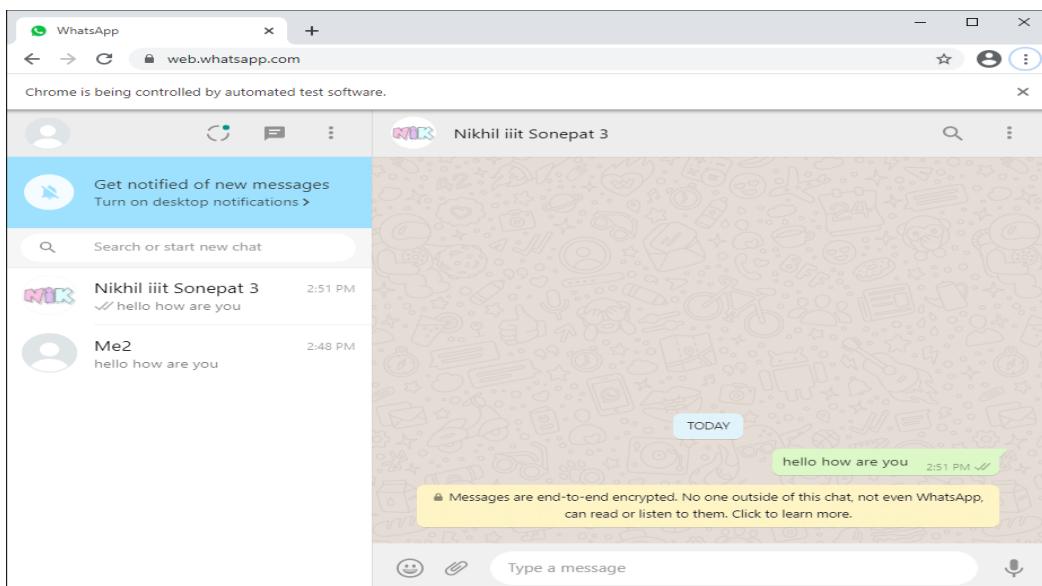
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Smart Voice Assistant3.py - Smart Voice Assistant2.py
HelloWorld projects Smart Voice Assistant2.py
Smart Voice Assistant2.py
Run: Smart Voice Assistant2.py
C:\Users\pc\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/pc/IdeaProjects/HelloWorld pro
Listening....
Recognizing....
WhatsApp
Enter receiverNIKHIL
Listening....
Recognizing....
hello how are you
Listening....
Recognizing....
close

Process finished with exit code 0

```

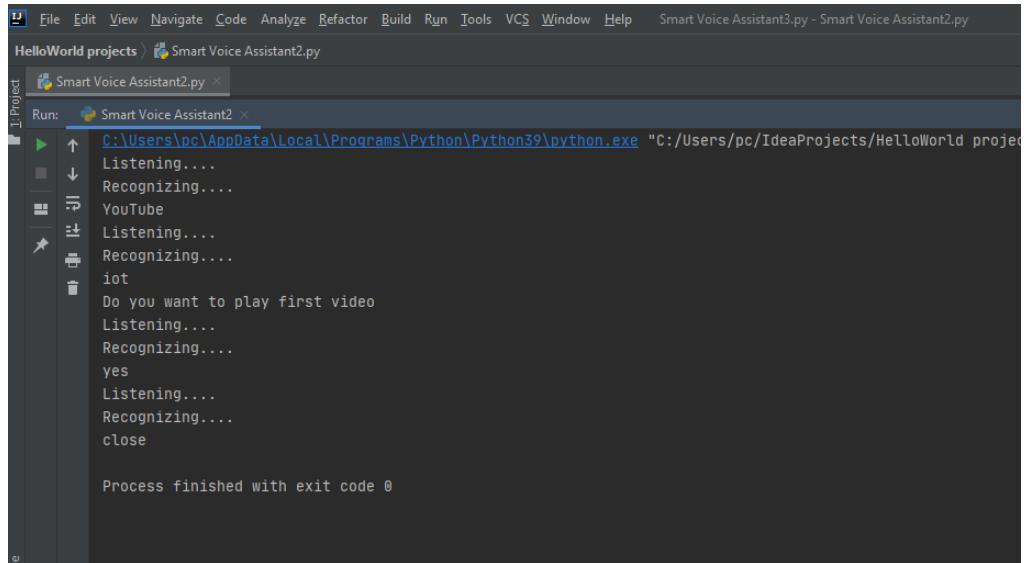
**Fig 3.1 Using Whatsapp with help of SVA**



**Fig 3.2 Output of Whatsapp Command**

- In Fig 3.1, When the command “Whatsapp” is given to SVA, it asks for the receiver's name and then asks for the message which is to be sent. Fig 3.2 is the proper action of the SVA. It opens WhatsApp and messages the receiver.

2) Searching and Playing Video on Youtube Using Smart voice Assistant – It will open YouTube and search on youtube and give the option to play the first video. It will be a completely hands-free experience.



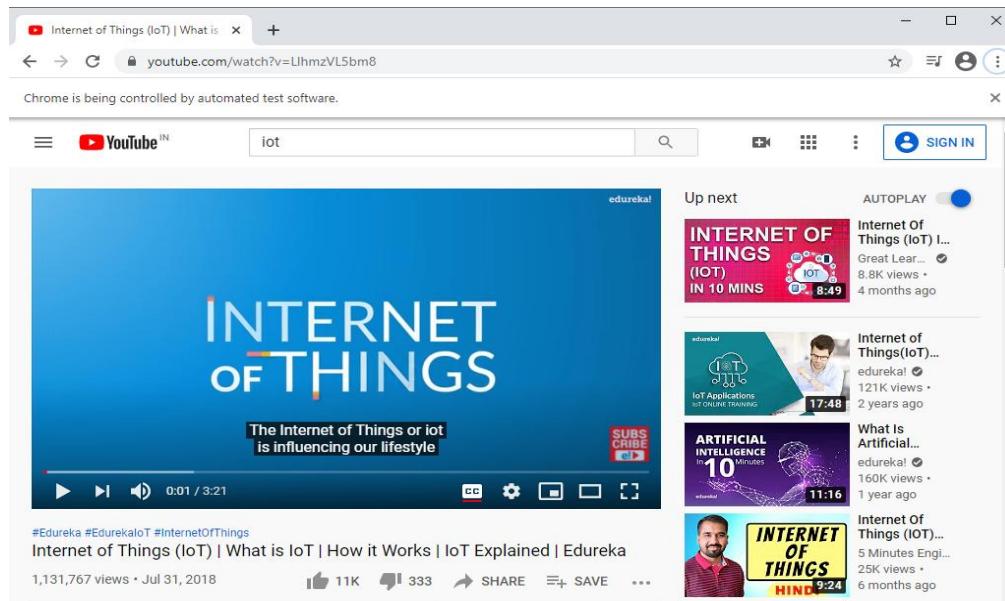
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Smart Voice Assistant3.py - Smart Voice Assistant2.py
HelloWorld projects > Smart Voice Assistant2.py
Smart Voice Assistant2.py
Run: Smart Voice Assistant2
Project: HelloWorld projects
C:\Users\pc\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/pc/IdeaProjects>HelloWorld project
Listening....
Recognizing....
YouTube
Listening....
Recognizing....
iot
Do you want to play first video
Listening....
Recognizing....
yes
Listening....
Recognizing....
close

Process finished with exit code 0

```

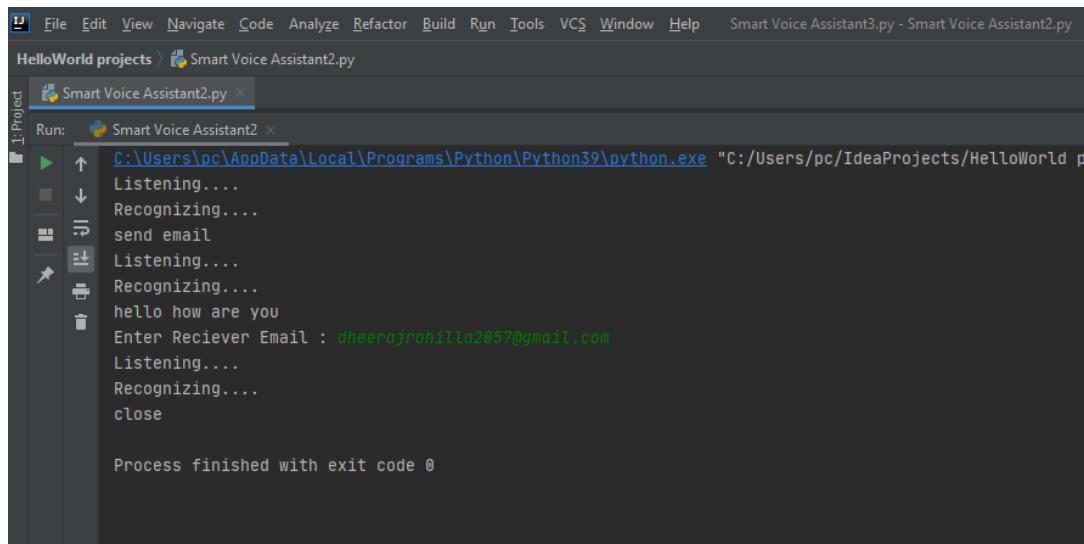
**Fig 3.3 Playing a Youtube video with the help of SVA**



**Fig 3.4 SVA playing Youtube Video**

- In Fig 3.3, SVA asks for the topic to be searched on youtube. The respected video is then played on Youtube with just the help of voice command in Fig 3.4.

3) Sending a Mail Using Smart Voice Assistant - It sends email using an SMTP server to send an email. It will be a completely hands-free experience only you have to write an email to the receiver.



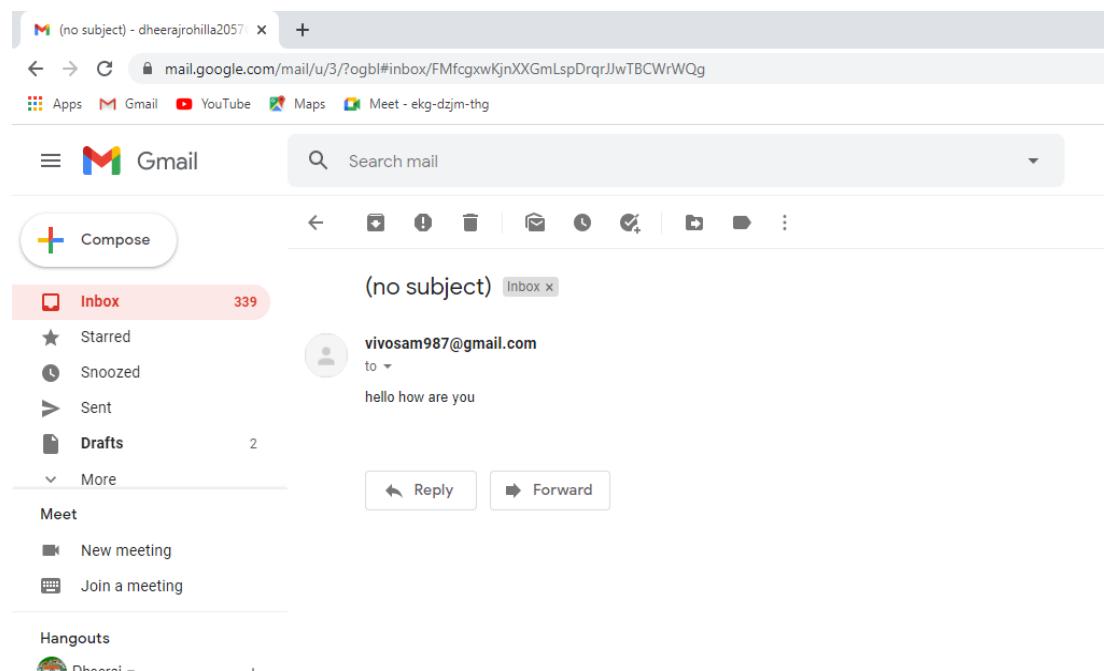
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Smart Voice Assistant3.py - Smart Voice Assistant2.py
HelloWorld projects > Smart Voice Assistant2.py
Smart Voice Assistant2.py
Project Run: Smart Voice Assistant2
Run: C:\Users\pc\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/pc/IdeaProjects/HelloWorld p
Listening....
Recognizing....
send email
Listening....
Recognizing....
hello how are you
Enter Reciever Email : dheerajröhilla2057@gmail.com
Listening....
Recognizing....
close

Process finished with exit code 0

```

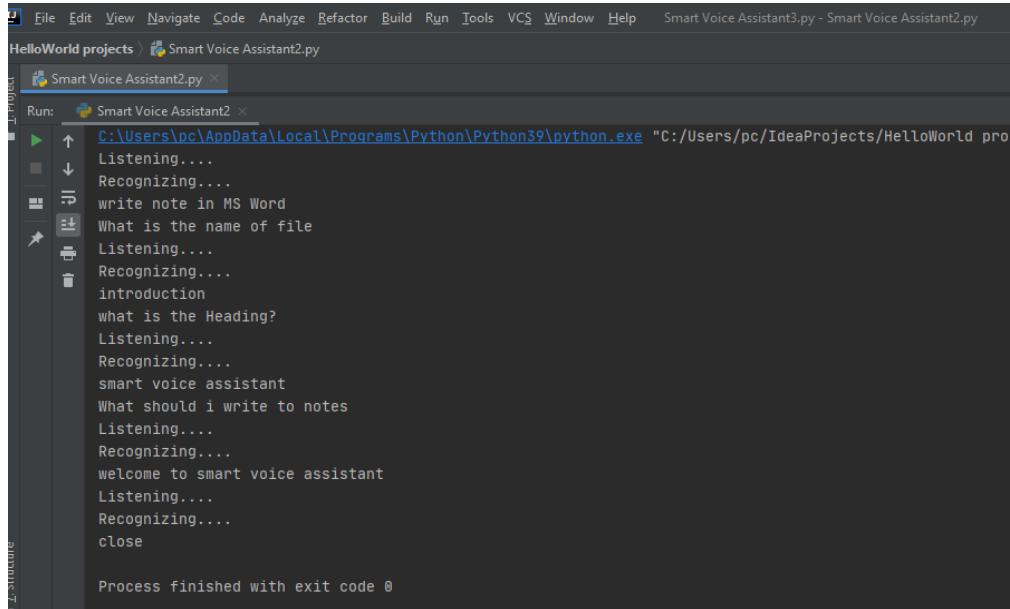
**Fig 3.5 Sending a mail with the help of SVA**



**Fig 3.6 SVA sending a mail**

- In Fig 3.5, SVA is asking for the receiver's mail and the content of the mail. In Fig 3.6, SVA sends the mail to the receiver.

4) Write Notes in MS Word Using Smart Voice Assistant - It makes notes simply by giving a voice command. It will be a completely hands-free experience.



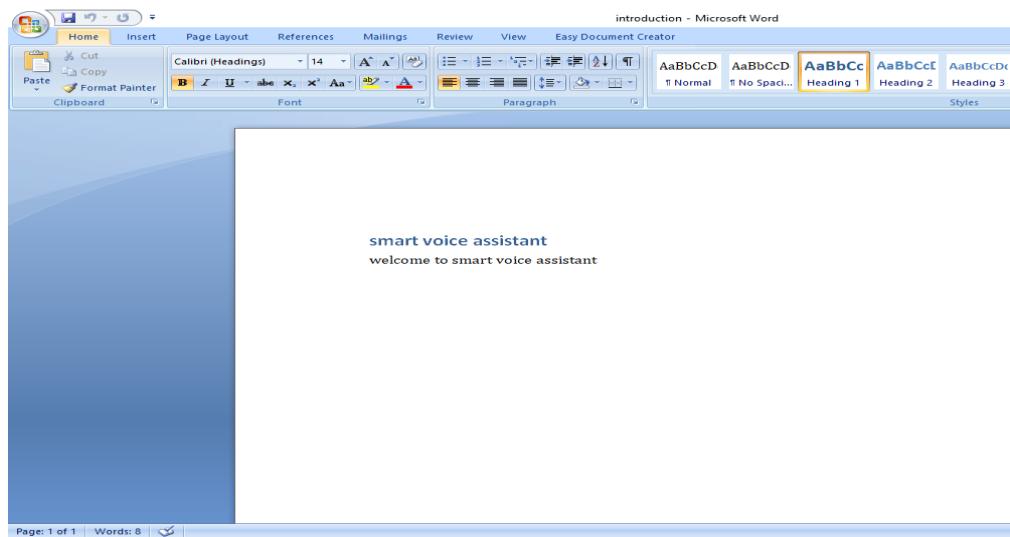
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Smart Voice Assistant3.py - Smart Voice Assistant2.py
HelloWorld projects / Smart Voice Assistant2.py
Smart Voice Assistant2.py
Run Smart Voice Assistant2
C:\Users\pc\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/pc/IdeaProjects/HelloWorld pro
Listening....
Recognizing....
write note in MS Word
What is the name of file
Listening....
Recognizing....
introduction
what is the Heading?
Listening....
Recognizing....
smart voice assistant
What should i write to notes
Listening....
Recognizing....
welcome to smart voice assistant
Listening....
Recognizing....
close

Process finished with exit code 0

```

**Fig 3.7 Using SVA to write notes in MS Word**



**Fig 3.8 SVA writing notes in MS Word**

- In Fig 3.7, SVA is Listening to the notes that are given by the user in the form of a voice command. And Fig 3.8, SVA is writing notes in MS WORD.

5) Basic Mathematical Calculation Using Smart Voice Assistant - We can do the basic mathematical calculation by just giving a voice command. It will be a completely hands-free experience.

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Smart Voice Assistant3.py - Smart Voice Assistant3.py
HelloWorld projects > Smart Voice Assistant3.py
Smart Voice Assistant2.py Smart Voice Assistant3.py
Run: Smart Voice Assistant2
C:\Users\pc\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/pc/IdeaProjects/HelloWorld projects/Smart Voice Assistant3.py"
Listening....
Recognizing....
calculate sin 45
The answer is : 1/sqrt(2)
Listening....
Recognizing....
calculate 11 x 2
The answer is : 22
Listening....
Recognizing....
close

Process finished with exit code 0

```

**Fig 3.9 SVA doing a basic mathematical calculation**

6) Getting Answers to basic questions Using Smart Voice Assistant – We can ask a basic question and the answer will be given by SVA. It will be a completely hands-free experience.

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Smart Voice Assistant3.py - Smart Voice Assistant2.py
HelloWorld projects > Smart Voice Assistant2.py Smart Voice Assistant3.py
Run: Smart Voice Assistant2
C:\Users\pc\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/pc/IdeaProjects/HelloWorld projects/Smart Voice Assistant3.py"
Listening....
Recognizing....
who is the CEO of Microsoft
Satya Nadella (Director and Chief Executive Officer)
Listening....
Recognizing....
who made Taj Mahal
Ustad Ahmad Lahori
Listening....
Recognizing....
exit

Process finished with exit code 0

```

**Fig 3.10 SVA responding to basic questions**

➤ **SUMMARY** – In this chapter, the implementation of SVA is discussed. Some main features are also discussed.

## **CHAPTER – 4**

### **TESTING AND VALIDATION**

#### ➤ Software Testing - Validation Testing -

Validation is the process of examining whether or not the software satisfies user requirements. It is carried out at the end of the SDLC. If the software matches the requirements for which it was made, it is validated.

- Validation ensures the product under development is as per the user requirements.
- Validation answers the question – "Are we developing the product which attempts all that user needs from this software ?".
- Validation emphasizes user requirements.

#### ➤ Software Verification

Verification is the process of confirming if the software is meeting the business requirements, and is developed adhering to the proper specifications and methodologies.

- Verification ensures the product being developed is according to design specifications.
- Verification answers the question– "Are we developing this product by firmly following all design specifications ?"
- Verifications concentrate on the design and system specifications.

The target of the test is -

- **Errors** - These are actual coding mistakes made by developers. In addition, there is a difference in the output of software and the desired output is considered as an error.
- **Fault** - When an error exists fault occurs. A fault, also known as a bug, is a result of an error that can cause the system to fail.
- **Failure** - failure is said to be the inability of the system to perform the desired task. Failure occurs when a fault exists in the system.

#### ➤ Manual Vs Automated Testing

Testing can either be done manually or using an automated testing tool:

- **Manual** - This testing is performed without taking the help of automated testing tools. The software tester prepares test cases for different sections and levels of the code, executes the tests, and reports the result to the manager.

Manual testing is time and resource consuming. The tester needs to confirm whether or not the right test cases are used. A major portion of testing involves manual testing.

- **Automated** This testing is a testing procedure done with aid of automated testing tools. The limitations of manual testing can be overcome using automated test tools.

A test needs to check if a webpage can be opened in Internet Explorer. This can be easily done with manual testing. But to check if the web-server can take the load of 1 million users, it is quite impossible to test manually.

There are software and hardware tools that help tester in conducting load testing, stress testing, regression testing.

### ➤ Testing Approaches

Tests can be conducted based on two approaches –

- Functionality testing
- Implementation testing

When functionality is being tested without taking the actual implementation into concern it is known as black-box testing. The other side is known as white-box testing where not only functionality is tested but the way it is implemented is also analyzed.

Exhaustive tests are the best-desired method for perfect testing. Every single possible value in the range of the input and output values is tested. It is not possible to test each and every value in a real-world scenario if the range of values is large.

### ➤ Black-box testing

It is carried out to test the functionality of the program. It is also called ‘Behavioral’ testing. The tester, in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested ‘ok’, and problematic otherwise.



**Fig 4.1 Black-Box Testing**

In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end-users conduct this test on the software.

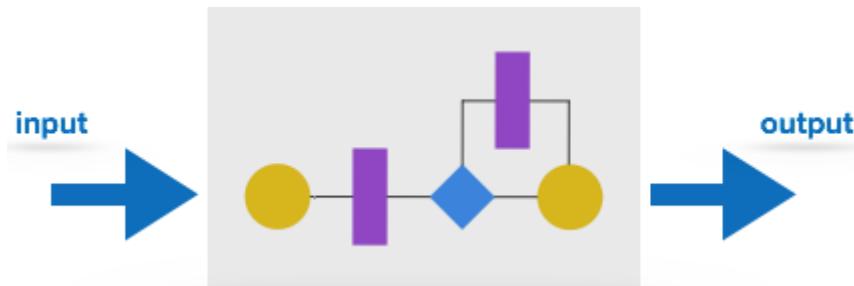
Black-box testing techniques:

- **Equivalence class** - The input is divided into similar classes. If one element of a class passes the test, it is assumed that all the class is passed.
- **Boundary values** - The input is divided into higher and lower-end values. If these values pass the test, it is assumed that all values in between may pass too.

- **Cause-effect graphing** - In both previous methods, only one input value at a time is tested. Cause (input) – Effect (output) is a testing technique where combinations of input values are tested in a systematic way.
- **Pair-wise Testing** - The behavior of software depends on multiple parameters. In pairwise testing, the multiple parameters are tested pair-wise for their different values.
- **State-based testing** - The system changes state on the provision of input. These systems are tested based on their states and input.

### ➤ White-box testing

It is conducted to test the program and its implementation, in order to improve code efficiency or structure. It is also known as ‘Structural’ testing.



**Fig 4.2 White-Box testing**

In this testing method, the design and structure of the code are known to the tester. Programmers of the code conduct this test on the code.

Below are some White-box testing techniques:

- **Control-flow testing** - The purpose of control-flow testing is to set up test cases that cover all statements and branch conditions. The branch conditions are tested for both being true and false so that all statements can be covered.
- **Data-flow testing** - This testing technique emphasizes to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

### ➤ Testing Levels

Testing itself may be defined at various levels of SDLC. The testing process runs parallel to software development. Before jumping on the next stage, a stage is tested, validated, and verified.

Testing separately is done just to make sure that there are no hidden bugs or issues left in the software. Software is tested on various levels -

### ➤ Unit Testing

While coding, the programmer performs some tests on that unit of the program to know if it is error-free. Testing is performed under the white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error-free.

## ➤ **Integration Testing**

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passing and data updation, etc.

## ➤ **System Testing**

The software is compiled as a product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- **Functionality testing** - Tests all functionalities of the software against the requirement.
- **Performance testing** - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do the desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environmental conditions.
- **Security & Portability** - These tests are done when the software is meant to work on various platforms and accessed by a number of persons.

## ➤ **Acceptance Testing**

When the software is ready to hand over to the customer it has to go through the last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if the user does not like the way it appears or works, it may be rejected.

- **Alpha testing** - The team of developers themselves performs alpha testing by using the system as if it is being used in the work environment. They try to find out how the user would react to some action in software and how the system should respond to inputs.
- **Beta testing** - After the software is tested internally, it is handed over to the users to use under their production environment only for testing purposes. This is not as yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

## ➤ **Regression Testing**

Whenever a software product is updated with new code, features, or functionality, it is tested thoroughly to detect if there is any negative impact of the added code. This is known as regression testing.

## ➤ **Testing Documentation**

Testing documents are prepared at different stages -

### **Before Testing**

Testing starts with test case generation. Following documents are needed for reference –

- **SRS document** - Functional Requirements document

- **Test Policy document** - This describes how far testing should take place before releasing the product.
- **Test Strategy document** - This mentions detailed aspects of the test team, responsibility matrix, and rights/responsibility of the test manager and test engineer.
- **Traceability Matrix document** - This is an SDLC document, which is related to the requirement gathering process. As new requirements come, they are added to this matrix. These matrices help testers know the source of requirements. They can be traced forward and backward.

### ➤ While Being Tested

The following documents may be required while testing is started and is being done:

- **Test Case document** - This document contains the list of tests required to be conducted. It includes the Unit test plan, Integration test plan, System test plan, and Acceptance test plan.
- **Test Description** - This document is a detailed description of all test cases and procedures to execute them.
- **Test case report** - This document contains a test case report as a result of the test.
- **Test logs** - This document contains test logs for every test case report.

### After Testing

The following documents may be generated after testing :

- **Test summary** - This test summary is the collective analysis of all test reports and logs. It summarizes and concludes if the software is ready to be launched. The software is released under a version control system if it is ready to launch.

### ➤ Testing vs. Quality Control, Quality Assurance, and Audit

We need to understand that software testing is different from software quality assurance, software quality control, and software auditing.

- **Software quality assurance** - These are software development process monitoring means, by which it is assured that all the measures are taken as per the standards of the organization. This monitoring is done to make sure that proper software development methods were followed.
- **Software quality control** - This is a system to maintain the quality of software products. It may include functional and non-functional aspects of the software product, which enhance the goodwill of the organization. This system makes sure that the customer is receiving a quality product for their requirement and the product certified as ‘fit for use’.
- **Software audit** - This is a review of the procedure used by the organization to develop the software. A team of auditors, independent of the development team examines the software process, procedure, requirements, and other aspects of SDLC. The purpose of a software audit is to check that software

and its development process, both conform to standards, rules, and regulations.

➤ **TESTING TECHNIQUE USED FOR SVA:-**

Manual testing is done for this project as it was not possible to automate the testing for this project. All the commands which are once set in SVA are working fine. Testing is done for many features like

- 1) Texting someone on Whatsapp
- 2) Playing any video on Youtube
- 3) Sending a mail to someone
- 4) Writing Notes in MS Word
- 5) Doing some basic calculation in SVA like calculate Sin45
- 6) Searching any query over the internet

➤ **FAIL CASE:-**

The one failure of sva is it can not work on the commands that are not available in the existing commands which were set by the user.

➤ **SUMMARY –**

Various software testing techniques are described in this chapter. Manual Testing is done for this project as automation for SVA was not possible.

## **CHAPTER - 5**

### **CONCLUSION AND FUTURE SCOPE**

#### ➤ **CONCLUSION AND SIGNIFICANCE –**

SVA uses Natural Language Processing to process the language spoken by the human and understand the query and process the query and respond to the human with the result. It is designed to minimize human efforts and control the PC with just a human Voice. The accuracy of the devices can be increased using machine learning and categorizing the queries in particular result sets and using them in further queries. The accuracy of the devices is increasing exponentially in the last decade. The devices can also be designed to accept commands in bilingual language and respond in the same language queried by the user. The device can also be designed to help visually impaired people.

It will have a high impact on society as it is very useful for disabled persons who cannot type or read.

#### ➤ **LIMITATIONS OF SVA –**

- Only supports the English language.
- Cannot work in a noisy environment.
- Cannot understand two different commands having the same meaning.
- SVA uses the human voice for interacting, using single commands as input for the device. These commands usually consist of single phrases. When commands become ambiguous, the resulting actions can be misunderstood by the devices.
- There is only one-way communication between the user and the SVA because the SVA cannot talk back for clarification.

#### ➤ **FUTURE SCOPE –**

- Not only small or easy tasks can be performed but SVA can also be automated with the help of artificial intelligence and machine learning. There are many tasks that the user performs every day. If SVA would be able to automate then the user just need to turn on the pc and all the daily repetitive tasks like opening zoom and connecting to a meeting can be performed by SVA very easily.
- In the future, SVA should be able to allow the user to train the software to understand their voice so that it can translate speech to text more precisely.
- The understanding of the SVA means Artificial Intelligence needs to be integrated with the device so that the SVA can work in a smart way and can also control IoT applications and devices and can also respond to query which will search the web for results and process it.

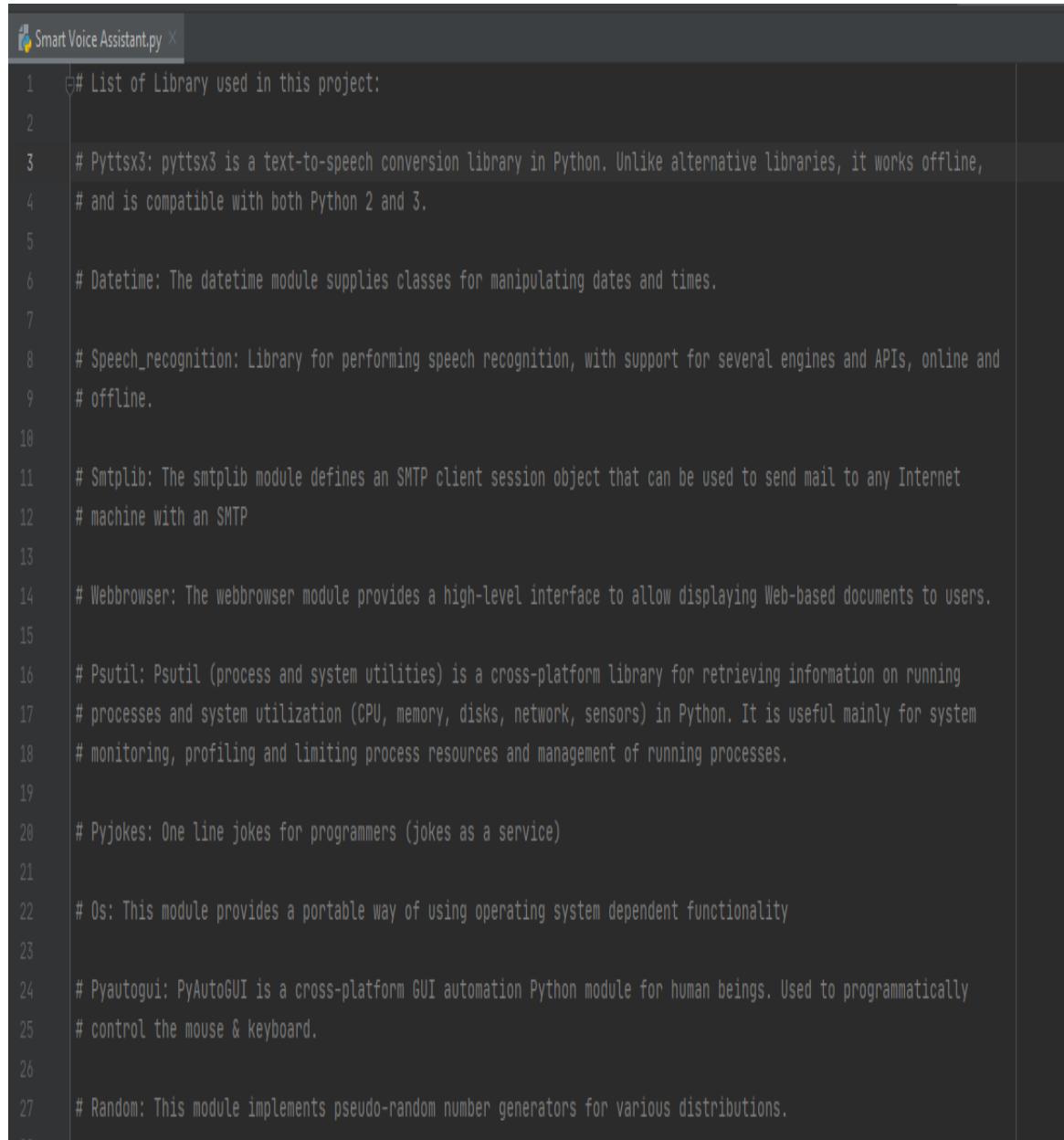
## **REFERENCES**

- 2.1) *BENEFITS OF VOICE ASSISTANT*, accessed 7 December 2020,  
<https://www.smartsheet.com/voice-assistants-artificial-intelligence/>
- 2.5) *Python*, accessed 6 December 2020, <https://www.python.org/>
- 2.6) *Github*, Built for developers, accessed 6 December 2020, <https://github.com/>
- 2.7) *ALL THE PYTHON TOOLS IN ONE PLACE*, accessed 6 December 2020,  
<https://www.jetbrains.com/pycharm/>
- 2.8) *Star UML*, accessed 6 December 2020, <https://staruml.io/>
- 4.1) Black-Box Testing, accessed 7 December 2020,  
[https://www.tutorialspoint.com/software\\_engineering/software\\_testing\\_overview.htm](https://www.tutorialspoint.com/software_engineering/software_testing_overview.htm)  
/
- 4.2) White-Box Testing, accessed 7 December 2020,  
[https://www.tutorialspoint.com/software\\_engineering/software\\_testing\\_overview.htm](https://www.tutorialspoint.com/software_engineering/software_testing_overview.htm)  
/

## **APPENDIX**

**GITHUB LINK-** <https://github.com/Dheeraj2000/Smart-Voice-Assistant-SVA-/>

### ➤ CODE –



The screenshot shows a code editor window titled "Smart Voice Assistant.py". The code is a list of comments explaining various Python modules used in the project. The modules listed include Pyttsx3, Datetime, Speech\_recognition, Smtplib, Webbrowser, Psutil, Pyjokes, Os, and Pyautogui. Each module is described in a single-line comment, such as "# Pyttsx3: pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, # and is compatible with both Python 2 and 3." The code is numbered from 1 to 27.

```
1  # List of Library used in this project:  
2  
3  # Pyttsx3: pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline,  
4  # and is compatible with both Python 2 and 3.  
5  
6  # Datetime: The datetime module supplies classes for manipulating dates and times.  
7  
8  # Speech_recognition: Library for performing speech recognition, with support for several engines and APIs, online and  
9  # offline.  
10  
11 # Smtp: The smtplib module defines an SMTP client session object that can be used to send mail to any Internet  
12 # machine with an SMTP  
13  
14 # Webbrowser: The webbrowser module provides a high-level interface to allow displaying Web-based documents to users.  
15  
16 # Psutil: Psutil (process and system utilities) is a cross-platform library for retrieving information on running  
17 # processes and system utilization (CPU, memory, disks, network, sensors) in Python. It is useful mainly for system  
18 # monitoring, profiling and limiting process resources and management of running processes.  
19  
20 # Pyjokes: One line jokes for programmers (jokes as a service)  
21  
22 # Os: This module provides a portable way of using operating system dependent functionality  
23  
24 # Pyautogui: PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically  
25 # control the mouse & keyboard.  
26  
27 # Random: This module implements pseudo-random number generators for various distributions.
```

```
27 # Random: This module implements pseudo-random number generators for various distributions.
28
29 # Docx: Python-docx is a Python library for creating and updating Microsoft Word (.docx) files.
30
31 # Wolframalpha: Basic usage is pretty simple. Create the client with your App ID (request from Wolfram Alpha) used to
32 # ask queries.
33
34 # Time: This module provides various time-related functions.
35
36 # Selenium: The selenium package is used to automate web browser interaction from Python.
37
38 # Keyboard: Take full control of your keyboard with this small Python library. Hook global events, register hotkeys,
39 # simulate key presses and much more.
40
41 # whatsapp youtube wikipedia notes in ms word
42
43
44 import pyttsx3 # pip install pyttsx3 # to convert text to speech
45 import datetime # to use datetime facility
46 import speech_recognition as sr # pip install speech_recognition for speech recognition
47 import wikipedia # pip install wikipedia
48 import smtplib # pip install smtplib# To send the mail you use smtpObj to connect to the SMTP server on the local machine.
49 import webbrowser as wb # to do web browser tasks
50 import psutil # pip install psutil
51 import pyjokes # pip install pyjokes
52 import os
53 import pyautogui # pip install pyautogui # gui related tasks like screen shot
54 import random
```



```
Smart Voice Assistant.py
34 import random
35 import docx # pip install docx
36 import requests
37 from urllib.request import urlopen
38 import wolframalpha # pip install wolframalpha
39 import time
40 from selenium import webdriver # pip install selenium
41 from keyboard import press # pip install keyboard
42
43
44 engine = pyttsx3.init()
45 wolframalpha_app_id = 'Y8Q8JU-3LH8Y8XQ85' #'Y8Q8JU-K52G4REV2A' #Y8Q8JU-3LH8Y8X085
46
47
48 class SystemInfo():
49
50     def cpu(self):
51         usage = str(psutil.cpu_percent()) # which return current value of cpu usage as a percentage
52         speak('CPU is at' + usage)
53         battery = psutil.sensors_battery()
54         speak("Battery is at")
55         speak(battery.percent)
56
57     def screenshot(self):
58         img = pyautogui.screenshot()
59         img.save('C:\\\\Users\\\\pc\\\\Documents\\\\scrnshot\\\\screenshot.jpg')
60
61     def time_(self):
```

```
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82     Time = datetime.datetime.now().strftime("%H:%M:%S") # for 12 hour clock use %I for 24 hour time
83     speak("The current time is")
84     speak(Time)
85
86     def date_(self):
87         year = datetime.datetime.now().year
88         month = datetime.datetime.now().month
89         date = datetime.datetime.now().day
90         speak("The current date is")
91         speak(date)
92         speak(month)
93         speak(year)
94
95
96     class SystemTasks():
97
98         def logout(self):
99             os.system("shutdown -1")
100
101        def restart(self):
102            os.system("shutdown /r /t 1")
103
104        def shutdown(self):
105            os.system("shutdown /s /t 1")
106
107        def cpu(self):
108            sysinfo = SystemInfo()
```

```
100     sysinfo = SystemInfo()
101     sysinfo.cpu()
102
103     def screenshot(self):
104         sysinfo = SystemInfo()
105         sysinfo.screenshot()
106
107     def time_(self):
108         sysinfo = SystemInfo()
109         sysinfo.time_()
110
111     def date_(self):
112         sysinfo = SystemInfo()
113         sysinfo.date_()
114
115
116
117
118
119
120
121
122
123
124     class SearchingTasks():
125         def __init__(self, obj1):
126             self.obj1 = obj1
127
128         def search(self):
129             self.obj1.search()
130
131
132     class InternetTasks():
133         def __init__(self, obj1):
134             self.obj1 = obj1
135
```

```
136
137 def sendEmail(to, content):
138     server = smtplib.SMTP('smtp.gmail.com', 587) #mailing service to use
139     server.ehlo() #help in identifying ourselves to smtp server
140     server.starttls() # which will helps us in putting connection to the smtp server into the TLS model
141     # for this function, you must enable low security in your gmail which you are going to use as sender
142
143     server.login('vivosam987@gmail.com', '9871960713')
144     server.sendmail('vivosam987@gmail.com', to, content)
145     server.close()
146
147
148 class Info():
149
150     def __init__(self):
151         self.driver = webdriver.Chrome(executable_path='C:\\\\Users\\\\pc\\\\Drivers\\\\chromedriver.exe')
152
153     def search(self, query):
154         self.query = query
155         self.driver.get(url="https://www.wikipedia.org/")
156         search = self.driver.find_element_by_xpath('//*[@id="searchInput"]')
157         search.click()
158         search.send_keys(query)
159
160         enter = self.driver.find_element_by_xpath('//*[@id="search-form"]/fieldset/button/i')
161         enter.click()
162
```

```
162
163     # the definition the bot can read
164     info = self.driver.find_element_by_xpath('//*[@id="mw-content-text"]/div[1]/p[2]')
165     readable_text = info.text
166     speak(readable_text)
167
168
169     class Gsearch():
170
171         def __init__(self):
172             self.driver = webdriver.Chrome(executable_path='C:\\\\Users\\\\pc\\\\Drivers\\\\chromedriver.exe')
173
174         def search(self, name):
175             self.driver.get(url="https://www.google.com")
176             search = self.driver.find_element_by_xpath('//*[@id="tsf"]/div[2]/div[1]/div/div[2]/input')
177             search.click()
178             search.send_keys(name)
179
180             submit = self.driver.find_element_by_name('btnK')
181             submit.click()
182
183
184         class YouPlay():
185
186             def __init__(self):
187                 self.driver = webdriver.Chrome(executable_path='C:\\\\Users\\\\pc\\\\Drivers\\\\chromedriver.exe')
188
189             def play(self, name):
190
191                 YouPlay
```

```
189     self.name = name
190     self.driver.get(url="https://www.youtube.com/results?search_query=" + name)
191     speak("Do you want to play first video")
192     print("Do you want to play first video")
193     response = TakeCommand().lower()
194     if 'yes' in response or 'sure' in response:
195         video = self.driver.find_element_by_xpath('//*[@id="video-title"]/yt-formatted-string')
196         video.click()
197
198
199 class Whatsapp():
200
201     def __init__(self):
202         self.driver = webdriver.Chrome(executable_path='C:\\\\Users\\\\pc\\\\Drivers\\\\chromedriver.exe')
203
204     def send(self, reciever, message):
205         self.driver.get(url="https://web.whatsapp.com/")
206         self.reciever = reciever
207         time.sleep(12)
208         search = self.driver.find_element_by_xpath('//*[@id="side"]/div[1]/div/label/div/div[2]')
209         search.click()
210         search.send_keys(reciever)
211         press('enter')
212         text = self.driver.find_element_by_xpath('//*[@id="main"]/footer/div[1]/div[2]/div/div[2]')
213         text.click()
214         text.send_keys(message)
215         button1 = self.driver.find_element_by_xpath('//*[@id="main"]/footer/div[1]/div[3]')
```

```
215     button1 = self.driver.find_element_by_xpath("//input[@id='button']/div[1]/div[2]//div[3]")
216     button1.click()
217     time.sleep(7)
218     self.driver.close()
219
220
221 class StackOverflow():
222
223     def __init__(self):
224         self.driver = webdriver.Chrome(executable_path='C:\\Users\\pc\\Drivers\\chromedriver.exe')
225
226     def Search(self, query):
227         self.driver.get(url="https://stackoverflow.com/")
228         time.sleep(4)
229         search = self.driver.find_element_by_xpath('//*[@id="search"]')
230         search.click()
231         search.send_keys(query)
232         search.click()
233         press("enter")
234
235
236     def speak(audio):
237         engine.say(audio)
238         engine.runAndWait() # wait until function is completed
239
240
241     def TakeCommand():
242         r = sr.Recognizer()
243         TakeCommand()
```

```
242     r = sr.Recognizer()
243
244     with sr.Microphone() as source:
245         print("Listening....")
246         r.pause_threshold = 1 # wait 1 sec until user command
247         audio = r.listen(source)
248     try:
249         print("Recognizing....")
250         query = r.recognize_google(audio, language='en-US') # audio was recognized by the google
251         print(query)
252     except Exception as e:
253         print(e)
254         print("Say again please")
255         return "None"
256     return query
257
258
259 def wishme():
260     speak("Welcome back")
261
262     # greetings
263     hour = datetime.datetime.now().hour
264
265     if hour>=6 and hour<12:
266         speak("Good Morning")
267
268     elif hour>=12 and hour<18:
269         speak("Good Afternoon sir")
```

```
267     speak("Good Morning sir")
268
269
270     elif hour>=18 and hour<24:
271         speak("Good Evening sir")
272
273
274     else:
275         speak("Good Night sir")
276
277     speak("Hello I am Smart Voice Assitant. Please tell me how can I help you today")
278
279
280 def joke():
281     speak(pyjokes.get_joke())
282
283
284
285 if __name__ == "__main__":
286
287     while True:
288         query = TakeCommand().lower() # commands convert to lowercase then stored in variable query
289
290         if 'time' in query:
291             sys1 = SystemTasks()
292             sys1.time_()
293
294         elif 'date' in query:
295             sys1 = SystemTasks()
296             sys1.date_()
```

```
297
298     elif 'wishme' in query:
299         wishme()
300
301     elif 'wikipedia' in query:
302         print("what do you want to search on wikipedia")
303         speak("what do you want to search on wikipedia")
304
305     try:
306         query = TakeCommand().lower()
307         wiki = Info()
308         any1 = SearchingTasks(wiki)
309         any1.obj1.search(query)
310     except sr.UnknownValueError:
311         print("")
312     except sr.RequestError as e:
313         print("")
314
315     elif 'send email' in query:
316         try:
317             print("What should I say")
318             speak("What should I say")
319             content = TakeCommand()
320             # provide receiver email address
321             speak("Who is the receiver?")
322             receiver = input("Enter Receiver Email : ") #abc@gmail.com
323             to = receiver
324
325         if __name__ == "__main__":
326             while True:
```

```
323         to = reciever
324         sendEmail(to, content)
325         speak(content)
326         print("Email has been sent")
327         speak("Email has been sent")
328     except Exception as e:
329         print(e)
330         print("Unable to send email")
331         speak("Unable to send email")
332
333
334     elif 'whatsapp' in query:
335         print("here we go to whatsapp")
336         speak("here we go to whatsapp")
337
338     try:
339         print("Speak the name of receiver")
340         speak("Speak the name of receiver")
341         receiver = TakeCommand().lower()#input('Enter receiver')#TakeCommand().lower()#
342         print("Speak Message")
343         speak("Speak Message")
344         message = TakeCommand()
345         Whatsapp1 = Whatsapp()
346         Whatsapp1.send(receiver, message)
347     except Exception as e:
348         print(e)
349         print("Unable to send message")
350         speak("Unable to send message")
351
352 if __name__ == "__main__":
353     while True:
```

```
350     speak( "Would you like to send message ?" )  
  
351  
352     elif 'Google meet' in query or 'Google meeting' in query:  
353         print("Here we go to meet")  
354         speak("Here we go to meet")  
355         speak("Enter link of meet")  
356         LinkOfMeet = input("Enter Link")  
357         wb.open(LinkOfMeet)  
358         time.sleep(10)  
359         press('ctrl+d')  
360         press('ctrl+e')  
  
361  
362     elif 'meet' in query:  
363         print("Here we go to meet")  
364         speak("Here we go to meet")  
365         speak("Enter Link of meet")  
366         LinkOfMeet = input("Enter Link")  
367         wb.open(LinkOfMeet)  
  
368  
369     elif 'youtube' in query:  
370         print("Here we go to YouTube")  
371         speak("Here we go to YouTube")  
  
372  
373     try:  
374         print("What should i search on youtube")  
375         speak("What should i search on youtube")  
376         query = TakeCommand().lower()  
377         PlayOnYou = YouPlay()  
  
    if __name__ == "__main__":
        while True
```

```
377         rPlayOnYou = rootPlay()
378         any1 = InternetTasks(PlayOnYou)
379         any1.obj1.play(query)
380     except sr.UnknownValueError:
381         print("")
382     except sr.RequestError as e:
383         print("")
384
385     elif 'stackoverflow' in query or 'stack over flow' in query or 'stack overflow' in query or 'stackover flow' in query:
386         print("Here we go to StackOverflow")
387         speak("Here we go to StackOverflow")
388
389     try:
390         print("What should i search on StackOverflow")
391         speak("What should i search on StackOverflow")
392         query = TakeCommand().lower()
393         StackOver = StackOverflow()
394         StackOver.Search(query)
395     except Exception as e:
396         print(e)
397         print("Unable to search")
398         speak("Unable to search")
399
400     elif 'google' in query:
401         print("What should I search on google")
402         speak("What should I search on google")
403
404     try:
405         query = TakeCommand().lower()
406
407     if __name__ == "__main__":
408         while True:
```

```
404     query = takecommand().lower()
405     searchOnGoogle = Gsearch()
406     any1 = SearchingTasks(searchOnGoogle)
407     any1.obj1.search(query)
408 except sr.UnknownValueError:
409     print("")
410 except sr.RequestError as e:
411     print("")
412
413 elif 'joke' in query or 'jokes' in query:
414     joke()
415
416 elif 'go offline' in query or 'exit' in query or 'quit' in query or 'close' in query:
417     speak("Exiting the Application")
418     print("Exiting the Application")
419     quit()
420
421 elif 'cleaner' in query:
422     print("opening CCleaner")
423     speak("opening CCleaner")
424     Path = r'C:\Program Files\CCleaner\CCleaner.exe'
425     os.startfile(Path)
426
427 elif 'vs code' in query or 'vscode' in query or 'visual studio code' in query:
428     print("opening VS Code")
429     speak("opening VS Code")
430     Path = r'C:\Users\pc\AppData\Local\Programs\Microsoft VS Code\Code.exe'
431     os.startfile(Path)
432
433 if __name__ == "__main__":
434     while True:
```

```
431         os.startfile(Path)
432
433     elif 'pycharm' in query or 'intellij' in query or 'python IDE' in query or 'python' in query:
434         print("opening VS python IDE")
435         speak("opening VS python IDE")
436         Path = r'C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.4\bin\idea64.exe'
437         os.startfile(Path)
438
439     elif 'filmora' in query or 'wondershare Filmora' in query or 'video editing software' in query or 'video edit' in query:
440         print("opening Wondershare Filmora")
441         speak("opening Wondershare Filmora")
442         Path = r'C:\Program Files\Wondershare\Filmora\Filmora.exe'
443         os.startfile(Path)
444
445     elif 'zoom' in query:
446         print("opening zoom meeting")
447         speak("opening zoom meeting")
448         Path = r'C:\Users\pc\AppData\Roaming\Zoom\bin\Zoom.exe'
449         os.startfile(Path)
450
451     elif 'notepad++' in query or 'notepad plus plus' in query or 'notepadplusplus' in query:
452         print("opening Notepad ++")
453         speak("opening Notepad ++")
454         Path = r'C:\Program Files (x86)\Notepad++\notepad++.exe'
455         os.startfile(Path)
456
457     elif 'write a note' in query:
458         print("what should i write, Sir?")
459
460 if __name__ == "__main__":
461     while True:
```

```
456     print("What should I write, Sir?")  
457     speak("what should i write, Sir")  
458     notes = TakeCommand()  
459     file = open('notes.txt', 'w')  
460     print("Sir should i include Date and Time?")  
461     speak("Sir should i include Date and Time?")  
462     ans = TakeCommand()  
463  
464     if 'yes' in ans or 'sure' in ans:  
465         strTime = datetime.datetime.now().strftime("%H:%M:%S")  
466         file.write(strTime)  
467         file.write(':')  
468         file.write(notes)  
469         speak('Done Taking Notes, Sir')  
470     else:  
471         file.write(notes)  
472  
473     elif 'show notes' in query:  
474         print("showing notes")  
475         speak("Showing Notes")  
476         file = open('notes.txt', 'r')  
477         print(file.read())  
478         speak(file.read())  
479  
480     elif 'write in word' in query or 'write note in ms word' in query or 'write in ms word' in query:  
481         print("What is the name of file")  
482         speak("What is the name of file")  
483         fileName = TakeCommand()  
484  
485     if __name__ == "__main__":
if __name__ == "__main__":
    while True
```

```
485     fileName = TakeCommand()
486     print("what is the Heading?")
487     speak("what is the Heading?")
488     heading = TakeCommand()
489     #speak("What is the size of heading 0, 1, 2....")
490     sizeOfHeading = 1 #int(TakeCommand())
491     print("What should i write to notes")
492     speak("What should i write to notes")
493     notes = TakeCommand()
494     doc = docx.Document()
495     doc.add_heading(heading, sizeOfHeading)
496     doc.add_paragraph(notes)
497     doc.save(fileName + '.docx')

498

499     elif 'play music' in query:
500         songs_dir = 'E:\\Songs'
501         music = os.listdir(songs_dir)
502         print("what should i play")
503         speak("what should i play")
504         print("Select a number....") # number 1
505         speak("Select a number 1, 2, 3 .......")
506         ans = TakeCommand().lower()

507

508     while 'number' not in ans and ans != 'random' and ans != 'you choose' and ans != 'choose yourself':
509         speak("I could not understand you. Please try again")
510         ans = TakeCommand().lower()
511         if 'number' in ans: # "number 1" -> "1" -> integer
512             no = int(ans.replace('number', ''))

if __name__ == "__main__":
    while True
```

```
512     no = int(ans.replace(' number ', ''))  
513     elif 'random' or 'you choose' or 'choose yourself' in ans:  
514         no = random.randint(1, len(music))  
515         os.startfile(os.path.join(songs_dir, music[no]))  
516  
517     elif 'remember that' in query:  
518         print("What should I remember?")  
519         speak("What should I remember?")  
520         memory = TakeCommand()  
521         print("You asked to remember that " + memory)  
522         speak("You asked to remember that " + memory)  
523         remember = open('memory.txt', 'w')  
524         remember.write(memory)  
525         remember.close()  
526  
527     elif 'do you remember anything' in query:  
528         remember = open('memory.txt', 'r')  
529         print("You asked me to remember that" + remember.read())  
530         speak("You asked me to remember that" + remember.read())  
531  
532     elif 'where is' in query:  
533         query = query.replace("where is", "")  
534         location = query  
535         print("User asked to locate " + location)  
536         speak("User asked to locate " + location)  
537         wb.open_new_tab("https://www.google.com/maps/place/" + location)  
538  
539     elif 'calculate' in query: # login to this site please https://products.wolframalpha.com/api/  
if __name__ == "__main__" > while True
```

```
537     if "calculate" in query: # login to this site please https://products.wolframalpha.com/api/
538
539     client = wolframalpha.Client(wolframalpha_app_id)
540
541     indx = query.lower().split().index('calculate') # [calculate, sin, 45] "0"
542
543     query = query.split()[indx + 1:]
544
545     res = client.query(''.join(query))
546
547     answer = next(res.results)
548
549     #print(answer)
550
551     try:
552         print('The answer is : ' + str(answer['subpod']['img'][ '@title' ]))
553         speak('The Answer is : ' + str(answer['subpod']['img'][ '@title' ]))
554     except StopIteration:
555         print("No Results")
556
557
558     elif 'what is' in query or 'who is' in query:
559
560         # use the same API key that we generate earlier i.e wolframalpha
561         client = wolframalpha.Client(wolframalpha_app_id)
562
563         res = client.query(query)
564
565         answer = next(res.results)
566
567         #print(answer)
568
569         try:
570             print(str(answer['subpod']['img'][ '@title' ]))
571             speak(str(answer['subpod']['img'][ '@title' ]))
572         except StopIteration:
573             print("No Results")
574
575
576     if __name__ == "__main__":
577         while True:
```

```
567     elif 'stop listening' in query:
568         print('For How many seconds you want me to stop listening to your commands?')
569         speak('For How many seconds you want me to stop listening to your commands?')
570         ans = int(TakeCommand())
571         time.sleep(ans)
572         print(ans)
573
574     elif 'log out' in query:
575         sys1 = SystemTasks()
576         sys1.logout()
577
578     elif 'restart' in query:
579         sys1 = SystemTasks()
580         sys1.restart()
581
582     elif 'shutdown' in query:
583         sys1 = SystemTasks()
584         sys1.shutdown()
585
586     elif 'screenshot' in query:
587         sys1 = SystemTasks()
588         sys1.screenshot()
589
590     elif 'cpu' in query:
591         sys1.cpu()
592
593 if __name__ == "__main__":
594     while True:
```