

TOM - VOICE ASSISTANT

A Major Project Report Submitted
In partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

KIRTHI VAISHNAVI	18N31A05A8
K MOHSINA KAUSER	18N31A0589
KACHAM HARICHARAN	18N31A0593

Under the esteemed guidance of

BIKSHAPATHY PERUKA
Associate Professor



Department of Computer Science and Engineering

Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA & NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: www.mrcet.ac.in

2018-2022



Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA & NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: www.mrcet.ac.in

CERTIFICATE

This is to certify that this is the bonafide record of the project entitled “TOM – VOICE ASSISTANT”, submitted by KIRTHI VAISHNAVI (18N31A05A8), K MOHSINA KAUSER (18N31A0589) and KACHAM HARICHARAN (18N31A0593) of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering, Department of CSE during the year 2020-2021. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide

Bikshapathy peruka
Associate Professor

Head of the Department

Dr. T Venugopal
Professor

External Examiner

DECLARATION

We hereby declare that the project titled “TOM – VOICE ASSISTANT” submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original research carried-out in this thesis. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

KIRTHI VAISHNAVI (18N31A05A8)

K MOHSINA KAUSER (18N31A0589)

KACHAM HARICHARAN (18N31A0593)

ACKNOWLEDGEMENT

We feel ourselves honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous) and our principal Dr. S SRINIVASA RAO who gave us the opportunity to have experience in engineering and profound technical knowledge.

We express our heartiest thanks to our Head of the Department Dr. T Venugopal for encouraging us in every aspect of our project and helping us realize our full potential.

We would like to thank our internal guide and Project Coordinator Mr Bikshapathy Peruka for his regular guidance and constant encouragement. We are extremely grateful to him valuable suggestions and unflinching co-operation throughout project work.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project that gave us strength to do our project.

with regards and gratitude

KIRTHI VAISHNAVI (18N31A05A8)

K MOHSINA KAUSER (18N31A0589)

KACHAM HARICHARAN (18N31A0593)

ABSTRACT

1. Title of the Project: Voice Assistant

2. Objective / Vision: A digital assistant that uses voice recognition and voice synthesis to listen to specific voice commands and return relevant information or perform specific function as requested by the user.

3. Users of the System:

A) People from all over the world

4. Functional Requirements: Following are the minimum set of functions that my project has to perform.

- A) Converts audio to text.
- B) Searches in browser.
- C) Send email.
- D) Opens google, youTube, Wikipedia etc.
- E) Tell Jokes.
- F) Gives weather report.
- G) Gives current news.
- H) Search for a place and gives its location.
- I) Writes notes and reads it.
- J) Performs Simple Calculations.
- K) Empties recycle bin.
- L) Sleeps for sometime.
- M) Tells current time.

5. Non-Functional Requirements:

- 1. 24 X 7 availability
- 2. Better component design to get better performance at peak time
- 3. Flexible service-based architecture will be highly desirable for future extensions.

6. User Interface Priorities:

A) Use of Python for both front-end and back-end.

7. Other Important Issues:

A) Simplicity of interface.

8. Team Size: 3

9. Technologies to be used: Python and its packages

10. Tools to be Used:

- Jupyter Notebook/ Spyder/ Pycharm
- Windows will be the preferred OS.

11. Final Deliverable must include:

- A) Voice Assistant desktop application
- B) Application archive (.war/. rar) with source code
- C) Complete Source code

TABLE OF CONTENTS

S.NO	TITLE	PAGE NO
1	INTRODUCTION	11
	1.1 Project Overview	11
	1.2 Project description	11
	1.3 Scope of the Project	11
	1.4 Modules	12
2	SYSTEM ANALYSIS	13
	2.1 Hardware and Software requirements	13
	2.2 Software Requirement Specification	14
3	TECHNOLOGIES USED	16
	3.1 Python	16
	3.2 Python Packages	19
4	SYSTEM DESIGN & UML DIAGRAMS	42
	4.1 UML diagram	42
	4.2 Flowcharts	43
5	INPUT/OUTPUT DESIGN	47
	5.1 Input Design	47
	5.2 Output Design	47
6	IMPLEMENTATION	48
	6.1 code.py	48
7	TESTING	64
	7.1 Testing strategy	64
	7.2 Test cases	65

8	OUTPUT SCREENS	73
	8.1 Front End	73
9	CONCLUSION AND FUTURE SCOPE	74
	9.1 Conclusion	74
	9.2 Future Scope	74
10	BIBLIOGRAPHY	75
	10.1 References	75

LIST OF FIGURES

FIGURE.NO	NAME	PAGE NO
4.1	UML Diagram	42
4.2.1	code.py	43
4.2.1.1	click	43
4.2.1.1.1	speak	44
4.2.1.1.2	wishMe	44
4.2.1.1.3	username	45
4.2.1.1.4	takeCommand	45
4.2.1.1.5	sendEmail	46
4.2.1.2	shut	46
6.1(a)	import	48
6.1(b)	click	48
6.1(c)	wishMe	49
6.1(d)	username	50
6.1(e)	takeCommand	50
6.1(f)	sendEmail	51
6.1(g1)	__main__	52
6.1(g2)	__main__	53
6.1(g3)	__main__	54
6.1(g4)	__main__	55
6.1(g5)	__main__	56

6.1(g6)	__main__	57
6.1(g7)	__main__	58
6.1(h)	shut	62
6.1(i)	frontend	63
7.2.1	testcase(google)	66
7.2.2	testcase(YouTube)	67
7.2.3	testcase(search for a place)	69
7.2.4	testcase(Weather)	70
7.2.5	testcase(news)	72
8.1	frontend	73

1. INTRODUCTION

1.1 Project overview:

Voice assistant is a digital assistant that uses voice recognition, and voice synthesis to listen to specific voice commands and return relevant information or perform specific functions as requested by the user.

1.2 Project description:

As a personal assistant, TOM assists the end-user with day-to-day activities like general human conversation, Converts audio to text, Searches in browser, Send email, Opens google, YouTube, Wikipedia etc, tell Jokes, gives weather report, gives current news, Search for a place and gives its geographic location, Writes notes and reads it, performs simple calculations, empties recycle bin, sleeps for sometime, says current time. The user statements/ commands are analyzed with the help of Python and its packages to give an optimal solution.

Voice control is a major growing feature that change the way people can live. The voice assistant is commonly being used in smartphones and laptops. Python-based Voice assistant are the operating systems that can recognize human voice and respond via integrated voices. This voice assistant will gather the audio from the microphone and then convert that into text, later it is sent through GTTS (Google text to speech). GTTS engine will convert text into audio file in English language, then that audio is played using play sound package of python programming Language.

In this application we used python for front-end and back-end. We used the following packages in python: “**speechrecognition**”, “**pyttsx3**”, “**webbrowser**”, “**pyjokes**”. This application can also send e-mail through voice command given by users.

1.3 Scope of the project:

Voice assistants will continue to offer more individualized experiences as they get better at differentiating between voices. However, it's not just

developers that need to address the complexity of developing for voice as brands also need to understand the capabilities of each device and integration and if it makes sense for their specific brand.

Presently, TOM is being developed as a virtual assistant. Among the Various roles played by TOM are:

- A) Converts audio to text.
- B) Searches in browser.
- C) Send email.
- D) Opens google, youTube, Wikipedia etc.
- E) Tell Jokes.
- F) Gives weather report.
- G) Gives current news.
- H) Search for a place and gives its location.
- I) Writes notes and reads it.
- J) Performs Simple Calculations.
- K) Empties recycle bin.
- L) Sleeps for sometime.
- M) Tells current time.

1.4 Modules:

- tkinter
- subprocess
- pyttsx3
- speech_recognition
- datetime
- wikipedia
- webbrowser
- os
- winshell
- pyjokes
- smtplib
- ctypes
- time
- shutil
- twilio

2 SYSTEM ANALYSIS

In this chapter, we will discuss and analyze about the developing process of Voice assistant including software requirement specification (SRS) and comparison between existing and proposed system. The functional and non-functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out. Besides that, existing vs. proposed provides a view of how the proposed system will be more efficient than the existing one.

2.4 HARDWARE AND SOFTWARE REQUIREMENTS

2.4.1 Hardware Requirements:

Processor	:	Intel.
Ram	:	4 GB or above.
Memory	:	512GB

2.4.2 Software Requirements:

Technology/Language	:	Python 3.0 or later.
Operating System	:	Windows 7(32-bit) or above.
IDE	:	Visual studio/Pycharm/Spyder etc.
Browser	:	Any browser.

2.5 Software Requirement Specification:

2.2.1 SRS:

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

The SRS phase consists of two basic activities:

1) Problem/Requirement Analysis:

A digital assistant that uses voice recognition and voice synthesis to listen to specific voice commands and return relevant information or perform specific function as requested by the user.

2) Requirement Specification:

The following are the requirements of the project:

Processor	:	Intel.
Ram	:	4 GB or above.
Memory	:	512GB
Technology	:	Python 3.0 or later.
OS	:	Windows 7(32-bit) or above.
IDE	:	Any python IDEs
Browser	:	Any browser.

2.2.2 Existing System:

Some of the best existing voice assistants are:

1. Amazon Alexa - Best for Device Compatibility
2. Apple Siri - Best for Apple Users
3. Google Assistant - Best for Accurate Responses
4. Microsoft Cortana – Best for Windows10 Users
5. Samsung Bixby - Best for Samsung Users

2.2.2.1 Limitations:

1. Expensive
2. Existing voice assistants are limited to one specific OS
3. Some assistants do not have desktop support

2.2.3 Proposed System:

The goal of this project is to develop a digital assistant for desktop that uses voice recognition and voice synthesis to listen to specific voice commands and return relevant information or perform specific function as requested by the user.

2.2.3.1 Advantages:

1. Inexpensive
2. Can work on all operating systems (desktop)
3. Has desktop support

3 TECHNOLOGIES USED

3.1 Python:

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

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

History:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's *Benevolent Dictator For Life*, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the `2to3` utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

Advantages:

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Characteristics of Python:

Following are important characteristics of **Python Programming** –

- It supports functional and structured programming methods as well as OOP.

- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Applications of Python:

Python is one of the most widely used language over the web. Few of the applications are:

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

3.2 Python Packages:

3.2.1 tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example

```
#!/usr/bin/python

import Tkinter
top = Tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table –

Sr.No.	Operator & Description
1	<u>Button</u> The Button widget is used to display buttons in your application.
2	<u>Canvas</u> The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	<u>Checkbutton</u> The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	<u>Entry</u> The Entry widget is used to display a single-line text field for accepting values from a user.
5	<u>Frame</u> The Frame widget is used as a container widget to organize other widgets.
6	<u>Label</u> The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
7	<u>Listbox</u> The Listbox widget is used to provide a list of options to a user.
8	<u>Menubutton</u> The Menubutton widget is used to display menus in your application.
9	<u>Menu</u> The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	<u>Message</u> The Message widget is used to display multiline text fields for accepting values from a user.
11	<u>Radiobutton</u> The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
12	<u>Scale</u>

	The Scale widget is used to provide a slider widget.
13	<u>Scrollbar</u> The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
14	<u>Text</u> The Text widget is used to display text in multiple lines.
15	<u>Toplevel</u> The Toplevel widget is used to provide a separate window container.
16	<u>Spinbox</u> The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
17	<u>PanedWindow</u> A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	<u>LabelFrame</u> A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	<u>tkMessageBox</u> This module is used to display message boxes in your applications.

Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The *Pack()* Method – This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The *grid()* Method – This geometry manager organizes widgets in a table-like structure in the parent widget.

- The *place()* Method – This geometry manager organizes widgets by placing them in a specific position in the parent widget.

3.2.2 subprocess:

The subprocess module provides a consistent interface to creating and working with additional processes. It offers a higher-level interface than some of the other available modules, and is intended to replace functions such as `os.system()`, `os.spawn*()`, `os.popen*()`, `popen2.*()` and `commands.*()`. To make it easier to compare subprocess with those other modules, many of the examples here re-create the ones used for `os` and `popen`.

The subprocess module defines one class, `Popen` and a few wrapper functions that use that class. The constructor for `Popen` takes arguments to set up the new process so the parent can communicate with it via pipes. It provides all of the functionality of the other modules and functions it replaces, and more. The API is consistent for all uses, and many of the extra steps of overhead needed (such as closing extra file descriptors and ensuring the pipes are closed) are “built in” instead of being handled by the application code separately.

3.2.3 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. An application invokes the `pyttsx3.init()` factory function to get a reference to a `pyttsx3`. Engine instance. it is a very easy to use tool which converts the entered text into speech.

The `pyttsx3` module supports two voices first is female and the second is male which is provided by “sapi5” for windows.

It supports three TTS engines :

- *sapi5* – SAPI5 on Windows
- *nsss* – NSSpeechSynthesizer on Mac OS X
- *espeak* – eSpeak on every other platform

Installataion:

To install the pytttsx3 module, first of all, you have to open the terminal and write

pip install pytttsx3

3.2.4 speech_recognition:

Speech Recognition is an important feature in several applications used such as home automation, artificial intelligence, etc. This article aims to provide an introduction on how to make use of the SpeechRecognition library of Python. This is useful as it can be used on microcontrollers such as Raspberri Pis with the help of an external microphone.

The following must be installed:

Python Speech Recognition module:

```
sudo pip install SpeechRecognition
```

Speech Input Using a Microphone and Translation of Speech to Text

- **Configure Microphone (For external microphones):** It is advisable to specify the microphone during the program to avoid any glitches.
Type **lsusb** in the terminal. A list of connected devices will show up. The microphone name would look like this
USB Device 0x46d:0x825: Audio (hw:1, 0)
Make a note of this as it will be used in the program.
- **Set Chunk Size:** This basically involved specifying how many bytes of data we want to read at once. Typically, this value is specified in powers of 2 such as 1024 or 2048
- **Set Sampling Rate:** Sampling rate defines how often values are recorded for processing

- **Set Device ID to the selected microphone:** In this step, we specify the device ID of the microphone that we wish to use in order to avoid ambiguity in case there are multiple microphones. This also helps debug, in the sense that, while running the program, we will know whether the specified microphone is being recognized. During the program, we specify a parameter `device_id`. The program will say that `device_id` could not be found if the microphone is not recognized.
- **Allow Adjusting for Ambient Noise:** Since the surrounding noise varies, we must allow the program a second or too to adjust the energy threshold of recording so it is adjusted according to the external noise level.
- **speech to text translation:** This is done with the help of Google Speech Recognition. This requires an active internet connection to work. However, there are certain offline Recognition systems such as PocketSphinx, but have a very rigorous installation process that requires several dependencies. Google Speech Recognition is one of the easiest to use.

3.2.5 datetime:

Python, date and time are not a data type of its own, but a module named **datetime** can be imported to work with the date as well as time. **Datetime module** comes built into Python, so there is no need to install it externally.

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. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

The datetime classes are categorized into 6 main classes –

- **date** – An idealized naive date, assuming the current Gregorian calendar always was, and always will be, in effect. Its attributes are year, month and day.
- **time** – An idealized time, independent of any particular day, assuming that every day has exactly 24*60*60 seconds. Its attributes are hour, minute, second, microsecond, and tzinfo.

- `datetime` – Its a combination of date and time along with the attributes year, month, day, hour, minute, second, microsecond, and `tzinfo`.
- `timedelta` – A duration expressing the difference between two date, time, or `datetime` instances to microsecond resolution.
- `tzinfo` – It provides time zone information objects.
- `timezone` – A class that implements the `tzinfo` abstract base class as a fixed offset from the UTC (New in version 3.2).

Date class

When an object of this class is instantiated, it represents a date in the format YYYY-MM-DD. Constructor of this class needs three mandatory arguments year, month and date.

Constructor syntax:

```
class datetime.date(year, month, day)
```

The arguments must be in the following range –

- `MINYEAR <= year <= MAXYEAR`
- `1 <= month <= 12`
- `1 <= day <= number of days in the given month and year`

Current date

To return the current local date `today()` function of date class is used. `today()` function comes with several attributes (year, month and day). These can be printed individually.

Different functions available in date class are –

Function name	Description
<code>fromtimestamp(timestamp)</code>	Return the local date corresponding to the POSIX timestamp
<code>fromordinal(ordinal)</code>	Return the date corresponding to the proleptic Gregorian ordinal, where

Function name	Description
	January 1 of year 1 has ordinal 1.
<code>fromisoformat(date_string)</code>	Return a date corresponding to a <code>date_string</code> given in the format YYYY-MM-DD:
<code>fromisocalendar(year, week, day)</code>	Return a date corresponding to the ISO calendar date specified by year, week and day.

Time class

Time object represents local time, independent of any day.

Constructor Syntax:

```
class datetime.time(hour=0, minute=0, second=0, microsecond=0,
tzinfo=None, *, fold=0)
```

All the arguments are optional. `tzinfo` can be `None` otherwise all the attributes must be integer in the following range –

- $0 \leq \text{hour} < 24$
- $0 \leq \text{minute} < 60$
- $0 \leq \text{second} < 60$
- $0 \leq \text{microsecond} < 1000000$
- `fold` in `[0, 1]`

Datetime class

Information on both date and time is contained in this class. Like a date object, `datetime` assumes the current Gregorian calendar extended in both directions; like a time object, `datetime` assumes there are exactly 3600×24 seconds in every day.

Constructor Syntax:

```
class datetime.datetime(year, month, day, hour=0, minute=0,
second=0, microsecond=0, tzinfo=None, *, fold=0)
```

The year, month and day arguments are mandatory. tzinfo can be None, rest all the attributes must be an integer in the following range:

- MINYEAR <= year <= MAXYEAR
- 1 <= month <= 12
- 1 <= day <= number of days in the given month and year
- 0 <= hour < 24
- 0 <= minute < 60
- 0 <= second < 60
- 0 <= microsecond < 1000000
- fold in [0, 1]

Current date and time

You can print the current date and time using the now() function. now() function returns the current local date and time.

Other functions of datetime class are –

Function name	Description
utcnow()	Return the current UTC date and time, with tzinfo None.
fromtimestamp(timestamp, tz=None)	Return the local date and time corresponding to the POSIX timestamp.
utcfromtimestamp(timestamp)	Return the UTC datetime corresponding to the POSIX timestamp, with tzinfo None.
fromordinal(ordinal)	Return the datetime corresponding to the proleptic Gregorian ordinal, where January 1 of year 1 has ordinal 1.
combine(date, time, tzinfo=self.tzinfo)	Return a new datetime object whose date components are equal to the given date object's, and whose time components are equal to the given

Fucntion name	Description
	time object's.
fromisoformat(date_string)	Return a datetime corresponding to a date_string in one of the formats emitted by date.isoformat() and datetime.isoformat().
strptime(date_string, format)	Return a datetime corresponding to date_string, parsed according to format.

Timedelta class

Python timedelta() function is present under datetime library which is generally used for calculating differences in dates and also can be used for date manipulations in Python. It is one of the easiest ways to perform date manipulations.

Constructor syntax:

class datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)

Returns : Date

Tzinfo class

This is an abstract base class, meaning that this class should not be instantiated directly. An instance of (a concrete subclass of) tzinfo can be passed to the constructors for datetime and time objects. The latter objects view their attributes as being in local time, and the tzinfo object supports methods revealing offset of local time from UTC, the name of the time zone, and DST offset, all relative to a date or time object passed to them.

Timezone class

The timezone class is a subclass of tzinfo, each instance of which represents a timezone defined by a fixed offset from UTC.

Constructor syntax:

```
class datetime.timezone(offset, name=None)
```

The offset argument must be specified as a timedelta object representing the difference between the local time and UTC.

3.2.6 wikipedia:

Internet is the most significant source of information. All knowledge is just one click away from us if we have an internet connection. Therefore, it is necessary to know how we can gather correct information from the right source. When we retrieve the information from various sources, this term is called **Data Scraping**. We all have used Wikipedia. It is the land of the full of information.

Wikipedia is the largest platform on the internet, which contains tons of information. It is an open-source platform which manages by the community of volunteer editors using a wiki-based editing system. It is a multi-lingual encyclopedia.

Python provides the **Wikipedia module (or API)** to scrap the data from the Wikipedia pages. This module allows us to get and parse the information from Wikipedia. In simple words, we can say that it is worked as a little scrapper and can scrap only a limited amount of data. Before we start working with it, we need to install this module on our local machine.

Installation

This module wraps the official Wikipedia API. In the first step, we will install the Wikipedia module using the following pip command. Type the below command in the terminal-

```
$pip install wikipedia
```

The above command will install the module in the system. Now, we need to import it using the following command.

```
import wikipedia
```

3.2.7 webbrowser:

In Python, **webbrowser module** provides a high-level interface which allows displaying Web-based documents to users. The webbrowser module can be used to launch a browser in a platform-independent manner.

Syntax

- `webbrowser.open(url, new=0, autoraise=False)`
- `webbrowser.open_new(url)`
- `webbrowser.open_new_tab(url)`
- `webbrowser.get(usage=None)`
- `webbrowser.register(name, constructor, instance=None)`

Parameters

Parameter	Details
<code>webbrowser.open()</code>	
url	the URL to open in the web browser
new	0 opens the URL in the existing tab, 1 opens in a new window, 2 opens in new tab
autoraise	if set to True, the window will be moved on top of the other windows
<code>webbrowser.open_new()</code>	
url	the URL to open in the web browser
<code>webbrowser.open_new_tab()</code>	
url	the URL to open in the web browser
<code>webbrowser.get()</code>	
using	the browser to use
<code>webbrowser.register()</code>	
url	browser name
constructor	path to the executable browser (help)
instance	An instance of a web browser returned from the <code>webbrowser.get()</code> method

3.2.8 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. The `*os*` and `*os.path*` modules include many functions to interact with the file system.

Handling the Current Working Directory:

Consider **Current Working Directory(CWD)** as a folder, where the Python is operating. Whenever the files are called only by their name, Python assumes that it starts in the CWD which means that name-only reference will be successful only if the file is in the Python's CWD.

Getting the Current working directory:

To get the location of the current working directory `os.getcwd()` is used.

Changing the Current working directory:

To change the current working directory(CWD) `os.chdir()` method is used. This method changes the CWD to a specified path. It only takes a single argument as a new directory path.

Creating a Directory:

There are different methods available in the OS module for creating a director. These are –

- `os.mkdir()`
- `os.makedirs()`

Using `os.mkdir()`:

`os.mkdir()` method in Python is used to create a directory named path with the specified numeric mode. This method raise `FileExistsError` if the directory to be created already exists.

Using `os.makedirs()`:

`os.makedirs()` method in Python is used to create a directory recursively. That means while making leaf directory if any intermediate-level directory is missing, `os.makedirs()` method will create them all.

Listing out Files and Directories with Python:

os.listdir() method in Python is used to get the list of all files and directories in the specified directory. If we don't specify any directory, then list of files and directories in the current working directory will be returned.

Deleting Directory or Files using Python

OS module provides different methods for removing directories and files in Python. These are –

- Using `os.remove()`
- Using `os.rmdir()`

Using `os.remove()`

`os.remove()` method in Python is used to remove or delete a file path. This method can not remove or delete a directory. If the specified path is a directory then `OSError` will be raised by the method.

Commonly Used Functions:

1. `os.name`: This function gives the name of the operating system dependent module imported. The following names have currently been registered: 'posix', 'nt', 'os2', 'ce', 'java' and 'riscos'

2. `os.error`: All functions in this module raise `OSError` in the case of invalid or inaccessible file names and paths, or other arguments that have the correct type, but are not accepted by the operating system. `os.error` is an alias for built-in `OSError` exception.

3. `os.popen()`: This method opens a pipe to or from command. The return value can be read or written depending on whether the mode is 'r' or 'w'.

Syntax:

```
os.popen(command[, mode[, bufsize]])
```

Parameters `mode` & `bufsize` are not necessary parameters, if not provided, default 'r' is taken for `mode`.

- 4. `os.close()`:** Close file descriptor `fd`. A file opened using `open()`, can be closed by `close()` only. But file opened through `os.popen()`, can be closed with `close()` or `os.close()`. If we try closing a file opened with `open()`, using `os.close()`, Python would throw `TypeError`.
- 5. `os.rename()`:** A file `old.txt` can be renamed to `new.txt`, using the function `os.rename()`. The name of the file changes only if, the file exists and the user has sufficient privilege permission to change the file.
- 6. `os.remove()`:** Using the `Os` module we can remove a file in our system using the `remove()` method. To remove a file we need to pass the name of the file as a parameter.
- 7. `os.path.exists()`:** This method will check whether a file exists or not by passing the name of the file as a parameter. `OS` module has a sub-module named `PATH` by using which we can perform many more functions.
- 8. `os.path.getsize()`:** In this method, python will give us the size of the file in bytes. To use this method we need to pass the name of the file as a parameter.

3.2.9 winshell:

The `winshell` module is a light wrapper around the Windows shell functionality.

It includes convenience functions for accessing special folders, for using the shell's file copy, rename & delete functionality, and a certain amount of support for structured storage.

`winshell` is tested on all versions of Python from 2.4 to 3.2 (2.5+ for context management).

The `winshell` package allows you to find special folders on Windows, create shortcuts easily, work with metadata via "structured storage", use the Windows shell to accomplish file operations and work with the Windows Recycle Bin.

3.2.10 pyjokes:

Python supports creation of random jokes using one of its libraries. Let us explore it a little more, **Pyjokes** is a python library that is used to create one-line jokes for programmers. Informally, it can also be referred as a fun python library which is pretty simple to use. Let us see how you can actually use it to perform the required task,

Installation:

You can simply install it using pip with the following command:

```
pip install pyjokes
```

Usage:

Now to use it, we need to import the installed library in our python Script using the following command:

```
import pyjokes
```

it is necessary to get familiar with the two functions of Pyjokes library, namely **get_joke()** and **get_jokes()**.

Functions:

- **get_joke()**

Syntax:

```
get_joke(language,category)
```

As the name suggests, this function is used to actually return a single joke from a certain category and in a particular language, (Categories and languages will be introduced later in this article).

- **get_jokes()**

Syntax:

```
get_jokes(language,category)
```

It is similar to the `get_joke()` function, the only difference lies in the fact that instead of returning a single joke, it returns a list of random jokes from a certain category and in a particular language.

Parameters:

Language and category are the two parameters of `get_joke()` and `get_jokes()` functions.

Language specifies in which language you want the joke(s) to be displayed. By default, it is set to “en” that returns jokes in English. All other possible values for language parameter are described below:

Language	Values
-----------------	---------------

en	English
de	German
es	Spanish
it	Italian
gl	Galician
eu	Basque

Similarly, the category parameter specifies the category in which you want joke(s) to be displayed. By default, it is set to “neutral”. All other possible values for the category parameter are described below:

Category	Values
-----------------	---------------

neutral	Neutral geeky jokes
twister	Tongue-twister
all	All types of joke

3.2.11 smtplib:

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers.

Python provides **smtplib** module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

Here is a simple syntax to create one SMTP object, which can later be used to send an e-mail –

```
import smtplib

smtpObj = smtplib.SMTP( [host [, port [, local_hostname]]] )
```

Here is the detail of the parameters –

- **host** – This is the host running your SMTP server. You can specify IP address of the host or a domain name like `tutorialspoint.com`. This is optional argument.
- **port** – If you are providing *host* argument, then you need to specify a port, where SMTP server is listening. Usually this port would be 25.
- **local_hostname** – If your SMTP server is running on your local machine, then you can specify just *localhost* as of this option.

An SMTP object has an instance method called **sendmail**, which is typically used to do the work of mailing a message. It takes three parameters –

- The *sender* – A string with the address of the sender.
- The *receivers* – A list of strings, one for each recipient.
- The *message* – A message as a string formatted as specified in the various RFCs.

3.2.12 ctypes:

ctypes is a foreign function library for Python. It provides C compatible data types, and allows calling functions in DLLs or shared libraries. It can be used to wrap these libraries in pure Python. A foreign function library means that the Python code can call C functions using only Python, without requiring special or custom-made extensions.

This module is one of the most powerful libraries available to the Python developer. The **ctypes** library enables you to not only call functions in dynamically linked libraries (as described earlier), but can also be used for low-level memory manipulation. It is important that you understand the basics of how to use the **ctypes** library as it will be used for many examples and real-world cases throughout the book.

3.2.13 shutil:

Shutil module offers high-level operation on a file like a copy, create, and remote operation on the file. It comes under Python's standard utility modules. This module helps in automating the process of copying and removal of files and directories.

Copying Files to another directory:

shutil.copy() method in Python is used to copy the content of the source file to the destination file or directory. It also preserves the file's permission mode but other metadata of the file like the file's creation and modification times is not preserved. The source must represent a file but the destination can be a file or a directory. If the destination is a directory then the file will be copied into the destination using the base filename from the source. Also, the destination must be writable. If the destination is a file and already exists then it will be replaced with the source file otherwise a new file will be created.

Syntax: `shutil.copy(source, destination, *, follow_symlinks = True)`

Parameter:

- **source:** A string representing the path of the source file.
- **destination:** A string representing the path of the destination file or directory.
- **follow_symlinks (optional) :** The default value of this parameter is True. If it is False and source represents a symbolic link then destination will be created as a symbolic link.

Return Type: This method returns a string which represents the path of newly created file.

Copying the Metadata along with File:

shutil.copy2() method in Python is used to copy the content of the source file to the destination file or directory. This method is identical to **shutil.copy()** method but it also tries to preserve the file's metadata.

Syntax: `shutil.copy2(source, destination, *, follow_symlinks = True)`

Parameter:

- **source:** A string representing the path of the source file.
- **destination:** A string representing the path of the destination file or directory.
- **follow_symlinks (optional) :** The default value of this parameter is True. If it is False and source represents a symbolic link then it attempts to copy all metadata from the source symbolic link to the newly-created destination symbolic link. This functionality is platform dependent.

Return Type: This method returns a string which represents the path of newly created file.

Copying the content of one file to another:

shutil.copyfile() method in Python is used to copy the content of the source file to the destination file. The metadata of the file is not copied. Source and destination must represent a file and destination must be

writable. If the destination already exists then it will be replaced with the source file otherwise a new file will be created.

If source and destination represent the same file then SameFileError exception will be raised.

Syntax: `shutil.copyfile(source, destination, *, follow_symlinks = True)`

Parameter:

- **source:** A string representing the path of the source file.
- **destination:** A string representing the path of the destination file.
- **follow_symlinks (optional) :** The default value of this parameter is True. If False and source represents a symbolic link then a new symbolic link will be created instead of copying the file.

Return Type: This method returns a string which represents the path of newly created file.

- Python3

Replicating complete Directory:

shutil.copytree() method recursively copies an entire directory tree rooted at source (src) to the destination directory. The destination directory, named by (dst) must not already exist. It will be created during copying.

Syntax: `shutil.copytree(src, dst, symlinks = False, ignore = None, copy_function = copy2, ignore_dangling_symlinks = False)`

Parameters:

- **src:** A string representing the path of the source directory.
- **dest:** A string representing the path of the destination.
- **symlinks (optional) :** This parameter accepts True or False, depending on which the metadata of the original links or linked links will be copied to the new tree.
- **ignore (optional) :** If ignore is given, it must be a callable that will receive as its arguments the directory being visited by copytree(), and a list of its contents, as returned by os.listdir().
- **copy_function (optional):** The default value of this parameter is copy2. We can use other copy function like copy() for this

parameter. **ignore_dangling_symlinks (optional)** : This parameter value when set to True is used to put a silence on the exception raised if the file pointed by the symlink doesn't exist.

- **Return Value:** This method returns a string which represents the path of newly created directory.

Removing a Directory:

shutil.rmtree() is used to delete an entire directory tree, the path must point to a directory (but not a symbolic link to a directory).

Syntax: `shutil.rmtree(path, ignore_errors=False, onerror=None)`

Parameters:

- **path:** A path-like object representing a file path. A path-like object is either a string or bytes object representing a path.
- **ignore_errors:** If `ignore_errors` is true, errors resulting from failed removals will be ignored.
- **onerror:** If `ignore_errors` is false or omitted, such errors are handled by calling a handler specified by `onerror`.

Finding files:

shutil.which() method tells the path to an executable application that would be run if the given **cmd** was called. This method can be used to find a file on a computer which is present on the PATH.

Syntax: `shutil.which(cmd, mode = os.F_OK | os.X_OK, path = None)`

Parameters:

- **cmd:** A string representing the file.
- **mode:** This parameter specifies mode by which method should execute. **os.F_OK** tests existence of the path and **os.X_OK** Checks if path can be executed or we can say mode determines if the file exists and executable.
- **path:** This parameter specifies the path to be used, if no path is specified then the results of `os.environ()` are used
- **Return Value:** This method returns the path to an executable application

3.2.14 twilio:

Twilio is a developer platform for communications: the Twilio Customer Engagement Platform. Twilio's programmable application program interfaces (APIs) are a set of building blocks developers can use to build the exact customer experiences they want.

The Twilio Customer Engagement Platform can be used to build practically any digital experience, using capabilities like SMS, WhatsApp, Voice, Video, email, and even IoT, across the customer journey. Twilio powers communications for more than 190,000 businesses, and enables nearly 932 billion human interactions every year.

4. SYSTEM DESIGN & UML DIAGRAMS

4.1 UML Diagram

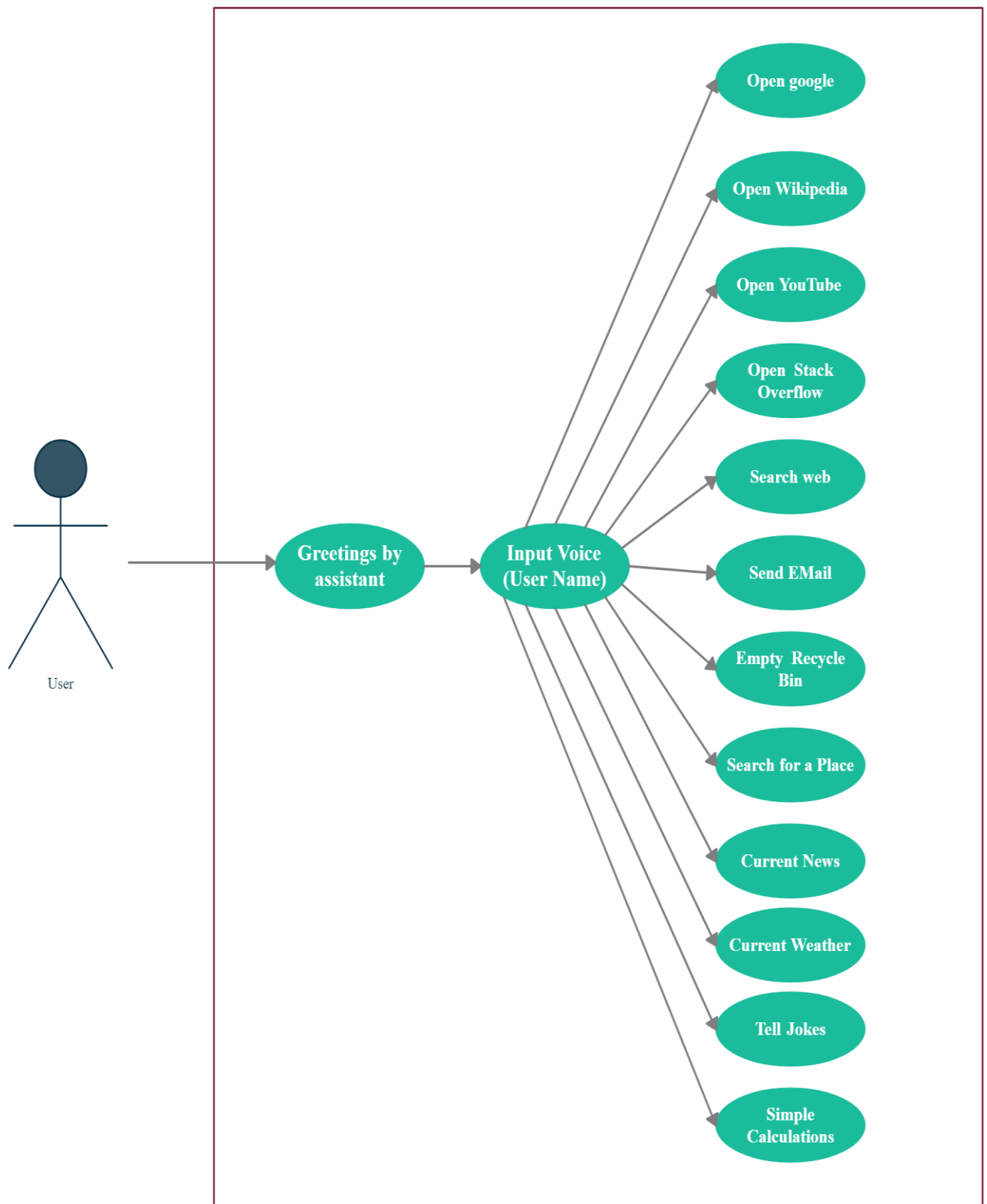


Fig 4.1:UML diagram

4.2 Flowcharts

4.2.1 code.py:

code.py(tkinter, subprocess, pyttsx3,speech_recognition,datetime,wikipedia,
webbrowser,os,winshell,pyjokes,smtplib,ctypes,time,shutil,twilio.rest)

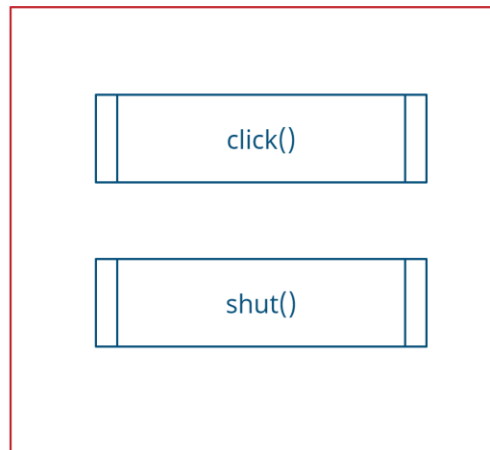


Fig 4.2.1:code.py

4.2.1.1 click():

click()

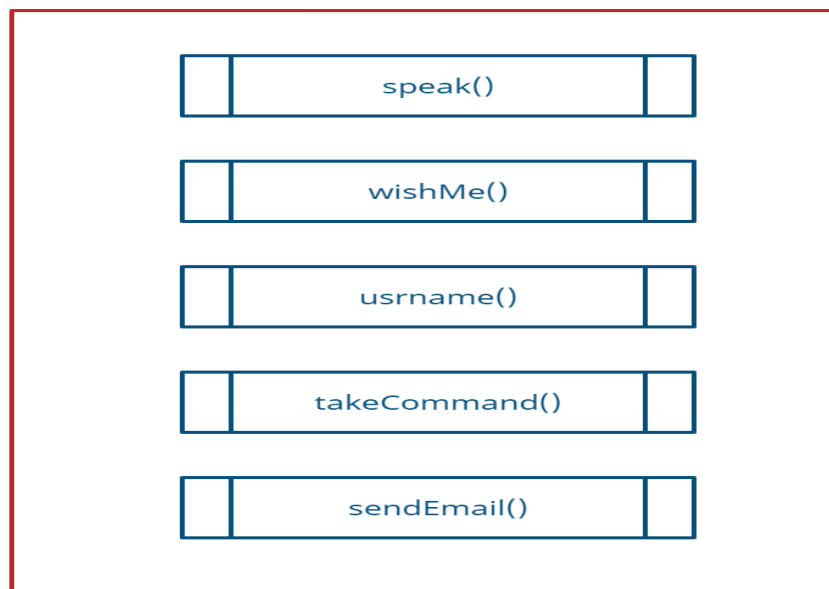


Fig 4.2.1.1:click

4.2.1.1.1 speak():

speak()

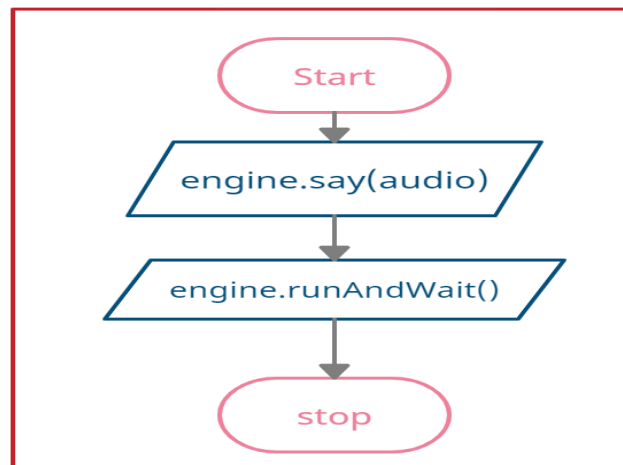


Fig 4.2.1.1.1:speak

4.2.1.1.2 wishMe():

wishMe()

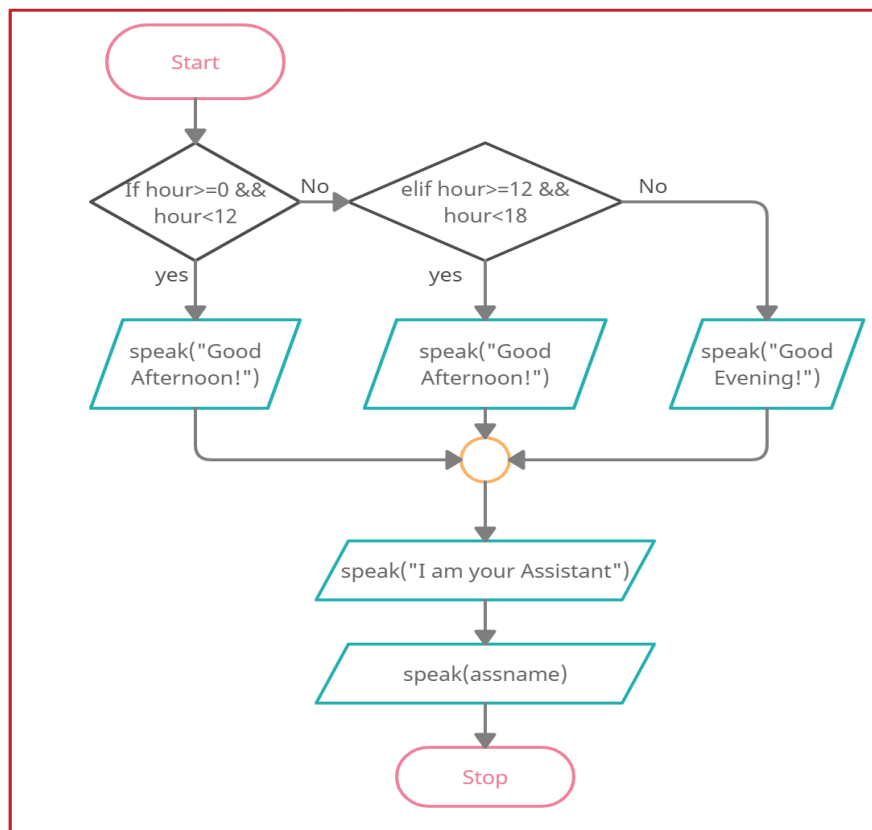


Fig 4.2.1.1.2:wishMe

4.2.1.1.3 username():

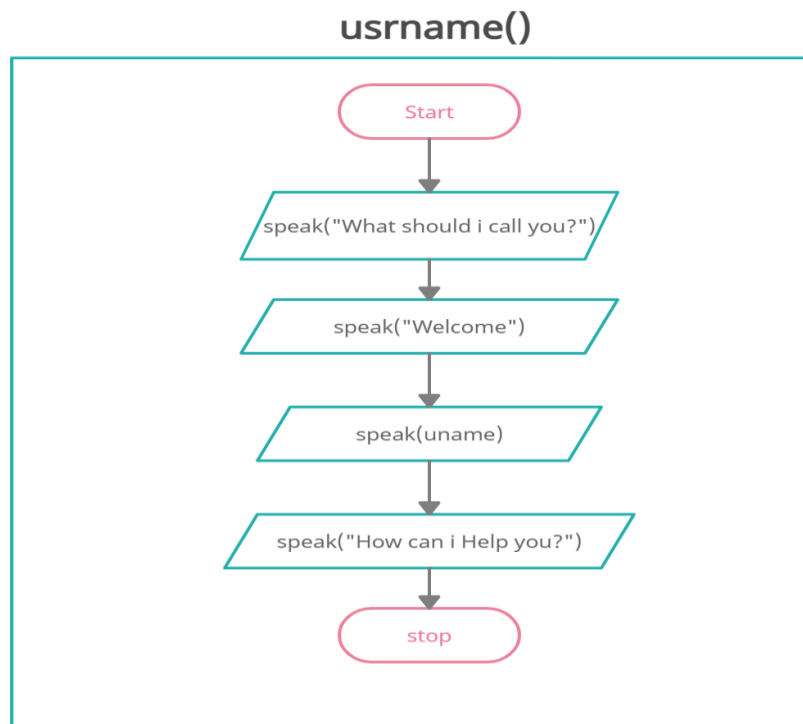


Fig 4.2.1.1.3:username

4.2.1.1.4 takeCommand():

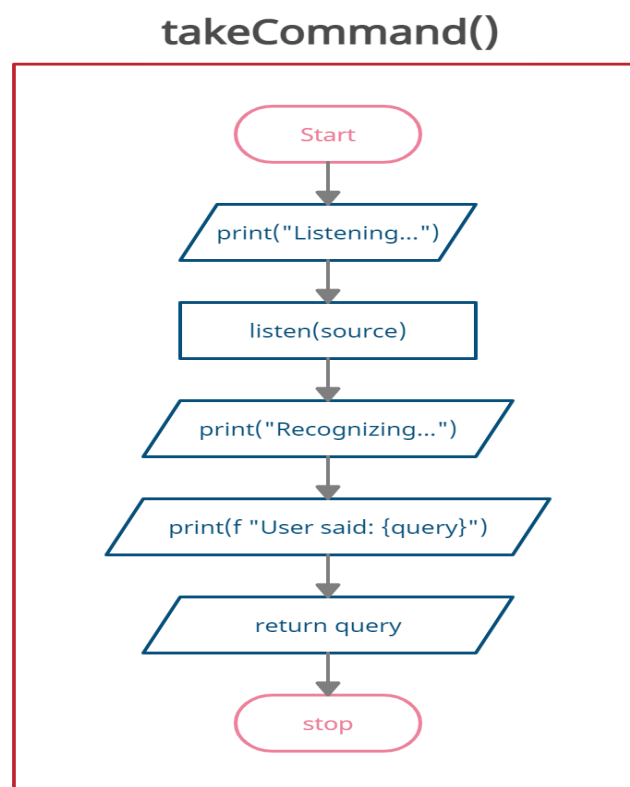


Fig 4.2.1.1.4:takeCommand

4.2.1.1.5 sendEmail():

sendEmail()

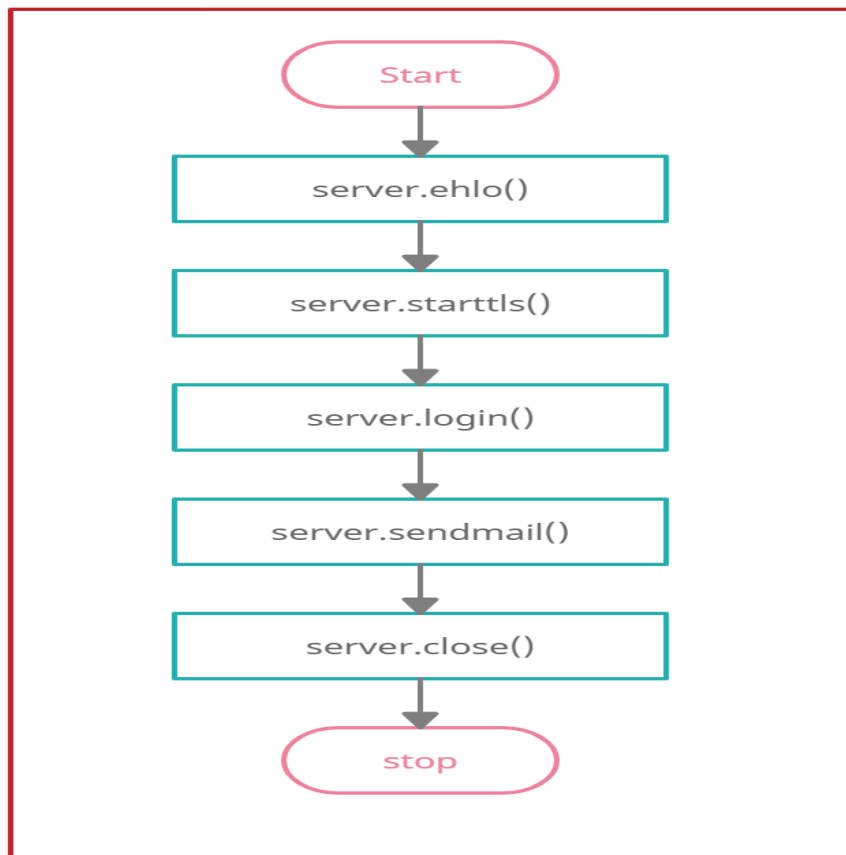


Fig 4.2.1.1.5:sendEmail

4.2.1.2 shut():

shut()

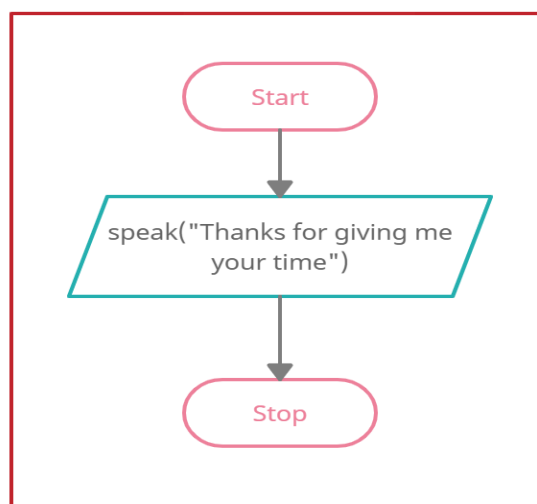


Fig 4.2.1.2:shut

5. INPUT/OUTPUT DESIGN

5.1 Input Design:

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations. Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors.

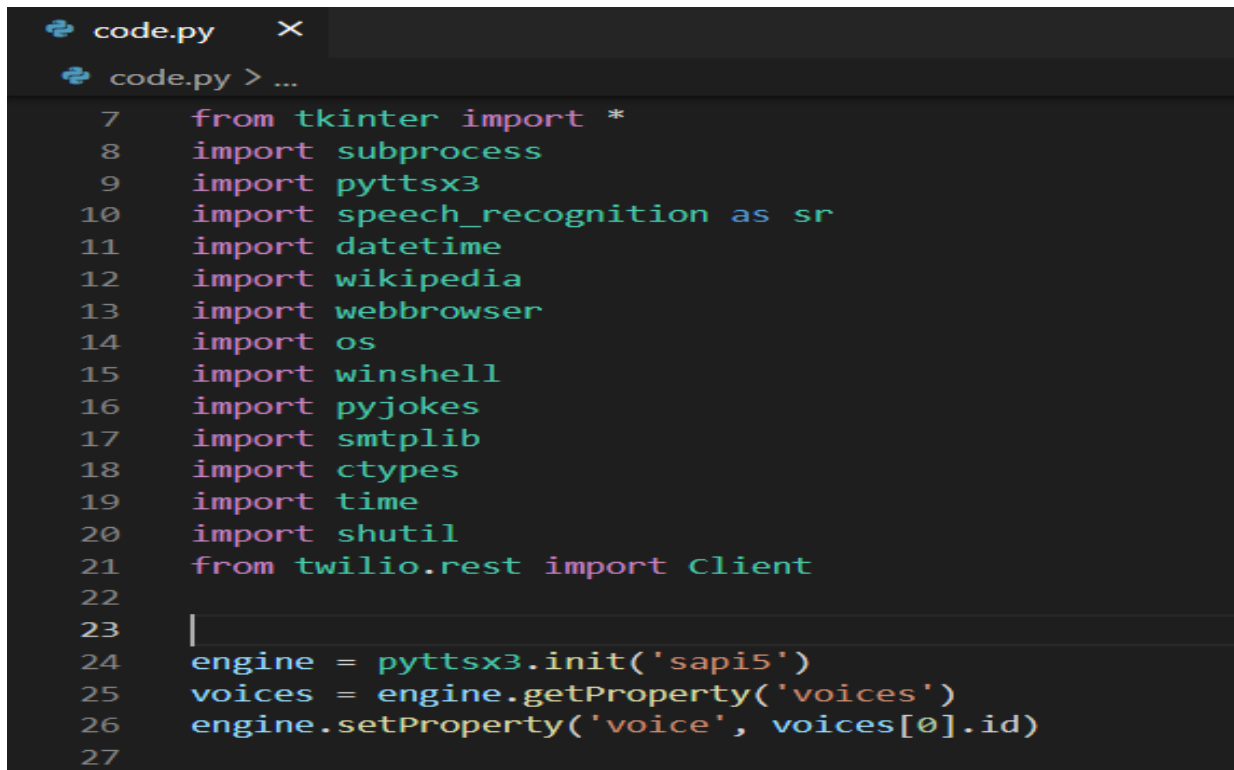
Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

5.2 Output Design:

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself.

6. IMPLEMENTATION

6.1 code.py



```
code.py  X
code.py > ...
7   from tkinter import *
8   import subprocess
9   import pyttsx3
10  import speech_recognition as sr
11  import datetime
12  import wikipedia
13  import webbrowser
14  import os
15  import winshell
16  import pyjokes
17  import smtplib
18  import ctypes
19  import time
20  import shutil
21  from twilio.rest import Client
22
23
24  engine = pyttsx3.init('sapi5')
25  voices = engine.getProperty('voices')
26  engine.setProperty('voice', voices[0].id)
27
```

Fig 6.1(a):import

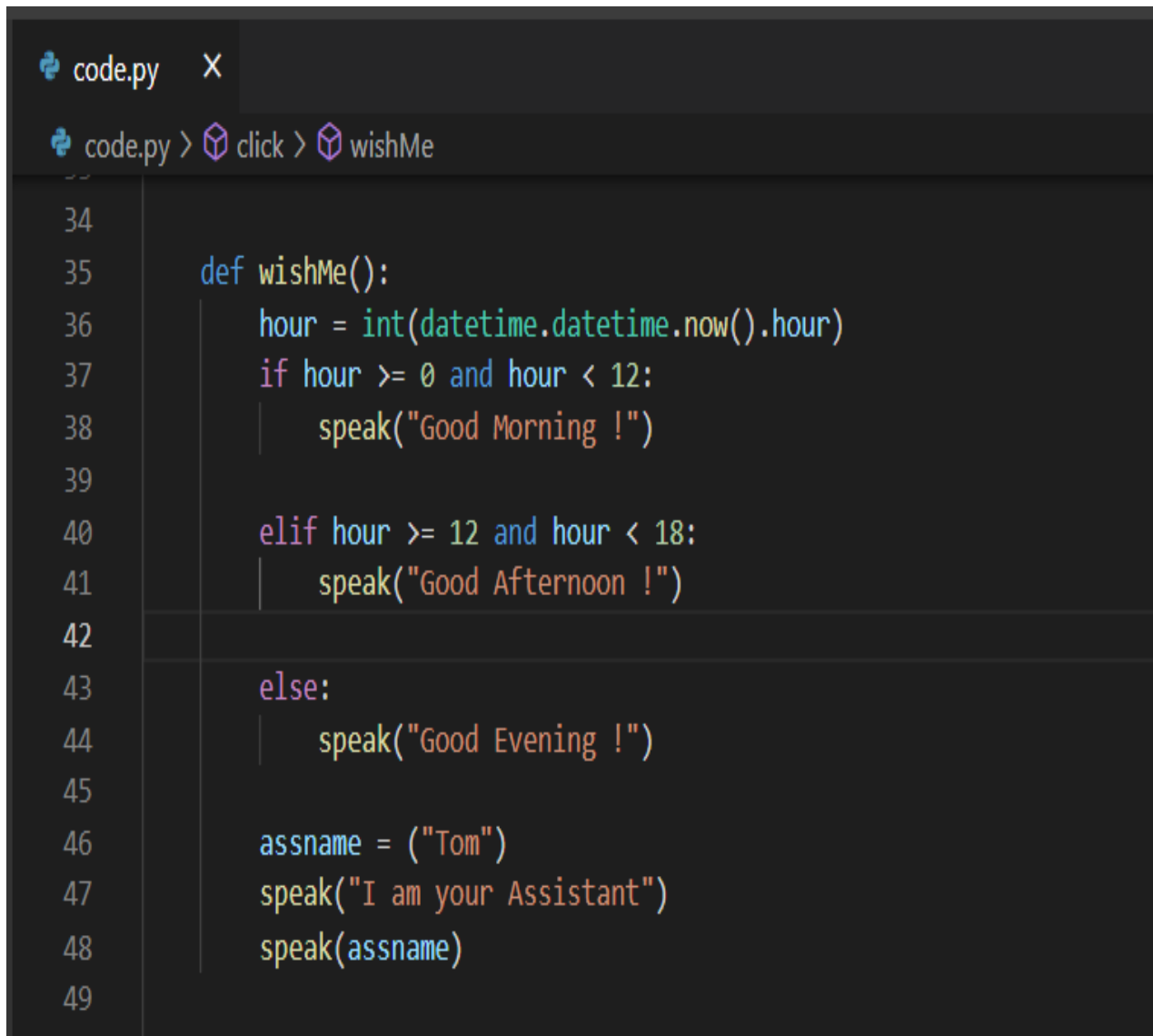
We have to import the above packages and set voice for the voice assistant. pyttsx3 module supports two voices one is female and other is male which is provided by sapi5 for windows. Here, we have selected male voice with id as 0.



```
code.py  X
code.py > click > wishMe
27
28  def click():
29      b1.config(text="Listening...")
30      def speak(audio):
31          engine.say(audio)
32          engine.runAndWait()
33
34
```

Fig 6.1(b):click

It prints listening on the screen and then it takes voice input from the user and performs the related tasks.



```
code.py X
code.py > click > wishMe
34
35     def wishMe():
36         hour = int(datetime.datetime.now().hour)
37         if hour >= 0 and hour < 12:
38             speak("Good Morning !")
39
40         elif hour >= 12 and hour < 18:
41             speak("Good Afternoon !")
42
43         else:
44             speak("Good Evening !")
45
46         assname = ("Tom")
47         speak("I am your Assistant")
48         speak(assname)
49
50
```

Fig 6.1(c):wishMe

First the voice assistant-TOM greets user based on the time. If the time is between 0 to 12 AM then it greets Good Morning!, if time is between 12PM and 18 PM then it greets Good Afternoon! Otherwise Good Evening!. After greeting the user then it introduces itself as "I am your Assistant TOM".

```
code.py X
code.py > click > wishMe
49
50
51     def username():
52         speak("What should i call you?")
53         uname = takeCommand()
54         speak("Welcome")
55         speak(uname)
56         columns = shutil.get_terminal_size().columns
57
58         print("#####".center(columns))
59         print("Welcome ", uname.center(columns))
60         print("#####".center(columns))
61
62         speak("How can i Help you?")
63
64
```

Fig 6.1(d):username

Now the assistant asks the user “What should I call you?” and then takes input from the user and speaks “Welcome” username. After welcoming the user it asks the user “How can I Help you?”.

```
code.py X
code.py > click > wishMe
64
65     def takeCommand():
66         r = sr.Recognizer()
67
68         with sr.Microphone() as source:
69
70             print("Listening...")
71             r.pause_threshold = 1
72             audio = r.listen(source)
73
74         try:
75             print("Recognizing...")
76             query = r.recognize_google(audio, language='en-in')
77             print(f"User said: {query}\n")
78
79         except Exception as e:
80             print(e)
81             print("Unable to Recognize your voice.")
82             return "None"
83
84         return query
85
```

Fig 6.1(e):takeCommand

Now, user can ask voice assistant to perform some tasks like opening any browser, searching for a place, Telling jokes etc. It prints “Listening” in the console, after it prints listening then only user can speak otherwise it prints “Unable to Recognize your voice”. If it recognizes the users voice it prints “User said: {query}” and returns the query and the query is the voice command given by the user.

A screenshot of a code editor window titled 'code.py'. The editor shows a Python function named 'sendEmail' starting at line 86. The function takes 'to' and 'content' as arguments. It creates an SMTP server object for 'smtp.gmail.com' on port 587, calls 'ehlo()', 'starttls()', and 'login()' with the email 'charanh117@gmail.com' and password '8328175557'. Then it calls 'sendmail()' with the same email, the 'to' parameter, and the 'content'. Finally, it calls 'close()'. The code is as follows:

```
86
87     def sendEmail(to, content):
88         server = smtplib.SMTP('smtp.gmail.com', 587)
89         server.ehlo()
90         server.starttls()
91
92         # Enable low security in gmail
93         server.login('charanh117@gmail.com', '8328175557')
94         server.sendmail('charanh117@gmail.com', to, content)
95         server.close()
96
97
```

Fig 6.1(f):sendEmail

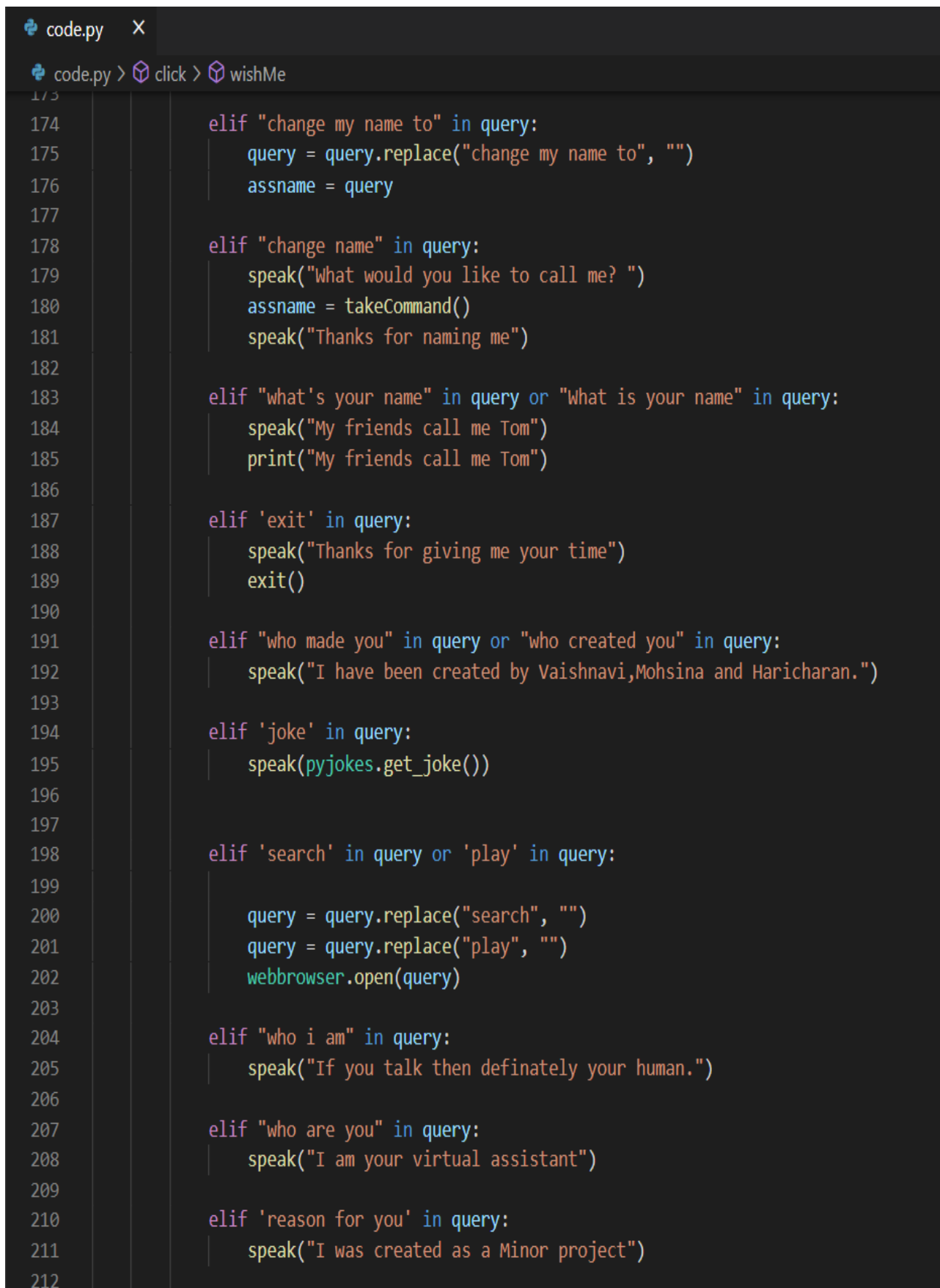
In this function, it connects to the server and logs into the users mail id and sends the mail when the user asks the TOM to send mail and then closes the server.

```
code.py X
code.py > click > wishMe
97
98     if __name__ == '__main__':
99         clear = lambda: os.system('cls')
100
101         clear()
102         wishMe()
103         usrname()
104
105     while True:
106
107         query = takeCommand().lower()
108
109         # All the commands said by user will be
110         # stored here in 'query' and will be
111         # converted to lower case for easily
112         # recognition of command
113         if 'from wikipedia' in query:
114             speak('Searching Wikipedia...')
115             query = query.replace("wikipedia", "")
116             results = wikipedia.summary(query, sentences = 3)
117             speak("According to Wikipedia")
118             print(results)
119             speak(results)
120         elif "open wikipedia" in query:
121             speak("Here you go to Wikipedia\n")
122             webbrowser.open("wikipedia.com")
123
124         elif 'open youtube' in query:
125             speak("Here you go to Youtube\n")
126             webbrowser.open("youtube.com")
127
128         elif 'open google' in query:
129             speak("Here you go to Google\n")
130             webbrowser.open("google.com")
131
132         elif 'open stackoverflow' in query:
133             speak("Here you go to Stack Over flow.Happy coding")
134             webbrowser.open("stackoverflow.com")
135
```

Fig 6.1(g1):__main__

```
code.py X
code.py > click > wishMe
135
136     elif 'open gmail' in query:
137         webbrowser.open_new_tab("gmail.com")
138         speak("Google Mail open now")
139
140     elif 'time' in query:
141         strTime = datetime.datetime.now().strftime('%H:%M %S')
142         speak(f"Current time is {strTime}")
143
144     elif 'email to charan' in query:
145         try:
146             speak("What should I say?")
147             content = takeCommand()
148             to = "keerthiyshu9@gmail.com"
149             sendEmail(to, content)
150             speak("Email has been sent !")
151         except Exception as e:
152             print(e)
153             speak("I am not able to send this email")
154
155     elif 'send a mail' in query:
156         try:
157             speak("What should I say?")
158             content = takeCommand()
159             speak("whom should i send")
160             to = input()
161             sendEmail(to, content)
162             speak("Email has been sent !")
163         except Exception as e:
164             print(e)
165             speak("I am not able to send this email")
166
167     elif 'how are you' in query:
168         speak("I am fine, Thank you")
169         speak("How are you")
170
171     elif 'fine' in query or "good" in query:
172         speak("It's good to know that your fine")
173
```

Fig 6.1 (g2):__main__



```
code.py X
code.py > click > wishMe
173
174     elif "change my name to" in query:
175         query = query.replace("change my name to", "")
176         assname = query
177
178     elif "change name" in query:
179         speak("What would you like to call me? ")
180         assname = takeCommand()
181         speak("Thanks for naming me")
182
183     elif "what's your name" in query or "What is your name" in query:
184         speak("My friends call me Tom")
185         print("My friends call me Tom")
186
187     elif 'exit' in query:
188         speak("Thanks for giving me your time")
189         exit()
190
191     elif "who made you" in query or "who created you" in query:
192         speak("I have been created by Vaishnavi,Mohsina and Haricharan.")
193
194     elif 'joke' in query:
195         speak(pyjokes.get_joke())
196
197
198     elif 'search' in query or 'play' in query:
199
200         query = query.replace("search", "")
201         query = query.replace("play", "")
202         webbrowser.open(query)
203
204     elif "who i am" in query:
205         speak("If you talk then definately your human.")
206
207     elif "who are you" in query:
208         speak("I am your virtual assistant")
209
210     elif 'reason for you' in query:
211         speak("I was created as a Minor project")
212
```

Fig 6.1 (g3):__main__

```
code.py X
code.py > click > wishMe
212
213 elif 'change background' in query:
214     ctypes.windll.user32.SystemParametersInfof(20,0,"location of wallpaper",0)
215     speak("Background changed succesfully")
216
217 elif 'news' in query:
218     news = webbrowser.open_new_tab("https://timesofindia.indiatimes.com/home/headlines")
219     speak('Here are some headlines from the Times of India,Happy reading')
220     time.sleep(6)
221
222
223 elif 'lock window' in query:
224     speak("locking the device")
225     ctypes.windll.user32.LockWorkStation()
226
227 elif 'shutdown system' in query:
228     speak("Hold On a Sec ! Your system is on its way to shut down")
229     subprocess.call('shutdown / p /f')
230
231 elif 'empty recycle bin' in query:
232     winshell.recycle_bin().empty(confirm=False, show_progress=False, sound=True)
233     speak("Recycle Bin Recycled")
234
235 elif "don't listen" in query or "stop listening" in query:
236     speak("for how much time you want to stop VaMoHa from listening commands")
237     a = int(takeCommand())
238     time.sleep(a)
239     print(a)
240
241 elif "where is" in query:
242     query = query.replace("where is", "")
243     location = query
244     speak("User asked to Locate")
245     speak(location)
246     webbrowser.open("https://www.google.nl/maps/place/" + location + "")
247
248 elif "restart" in query:
249     subprocess.call(["shutdown", "/r"])
```

Fig 6.1 (g4):__main__

```
code.py X
code.py > click > wishMe
250
251     elif "hibernate" in query or "sleep" in query:
252         speak("Hibernating")
253         subprocess.call("shutdown / h")
254
255     elif "log off" in query or "sign out" in query:
256         speak("Make sure all the application are closed before sign-out")
257         time.sleep(5)
258         subprocess.call(["shutdown", "/l"])
259
260     elif "write a note" in query:
261         speak("What should i write?")
262         note = takeCommand()
263         file = open(r'Tom.txt', 'w')
264         speak("Should i include date and time")
265         snfm = takeCommand()
266         if 'yes' in snfm or 'sure' in snfm:
267             strTime = datetime.datetime.now().strftime("%H: %M: %S")
268             file.write(strTime)
269             file.write(" :- ")
270             file.write(note)
271         else:
272             file.write(note)
273
274     elif "show note" in query or "open note" in query:
275         speak("Showing Notes")
276         file = open(r"Tom.txt", "r")
277         print(file.read())
278         speak(file.read(6))
279
280     # NPPR9-FWDCX-D2C8J-H872K-2YT43
281     elif "Tom" in query:
282
283         wishMe()
284         speak("Tom at your service")
285         speak(assname)
```

Fig 6.1(g5):__main__


```
code.py X
code.py > click > wishMe

286
287 elif 'current weather' in query:
288     search_term = query.split("for")[-1]
289     url = "https://www.google.com/search?sxsrf=ACYBGNSQmMLDBYBwdVFIUCbQgya-ET7AA%3A1578847393212&ei=0UwbxtbXDN-C4-EP-5u82AE&q=weather&oq=weather&gs_l=psy-ab.3..35i39i285i70i256j0i67l4j0i131i67j0i131j0i67l2j0.1630.4591..5475...1.2..2.322.1659.9j5j0j1.....0....1..gws-wiz.....10..0i71j35i39j35i362i39.5eSPD47bv8&ved=0ahUKEwIwPJvwmP7mAHvfwTgGHfsNDxsQ4dUDCAs&uact=5"
290     webbrowser.get().open(url)
291     speak("Here is what I found for on google")
292
293 elif "send message " in query:
294     # You need to create an account on Twilio to use this service
295     account_sid = 'Account Sid key'
296     auth_token = 'Auth token'
297     client = client(account_sid, auth_token)
298
299     message = client.messages \
300         .create(
301             body=takeCommand(),
302             from_='Sender No',
303             to='Receiver No'
304         )
305
306     print(message.sid)
307
308
309 elif "Good Morning" in query:
310     speak("A warm" + query)
311     speak("How are you")
312     speak(assname)
313
314 # most asked question from google Assistant
315 elif "will you be my gf" in query or "will you be my bf" in query:
316     speak("I'm not sure about, may be you should give me some time")
317
318 elif "how are you" in query:
319     speak("I'm fine, thanks for asking")
320
321 elif "i love you" in query:
322     speak("It's hard to understand")
```

Fig 6.1(g6):__main__

```
code.py X
code.py > click > wishMe

323
324     elif "add" in query:
325         sum=query.replace("add","").replace("plus","+")
326         ans=eval(sum)
327         speak(ans)
328
329     elif "subtract" in query:
330         sub=query.replace("subtract","").replace("minus","-")
331         ans=eval(sub)
332         speak(ans)
333
334     elif "multiply" in query:
335         star=query.replace("multiply","").replace("star","*")
336         ans=eval(star)
337         speak(ans)
338
339     elif "divide" in query:
340         By=query.replace("divide","").replace("by","/")
341         ans=eval(By)
342         speak(ans)
343
```

Fig 6.1(g7):__main__

From here,the main function starts, first the command given by the user is converted into lowercase and then assigned to query.

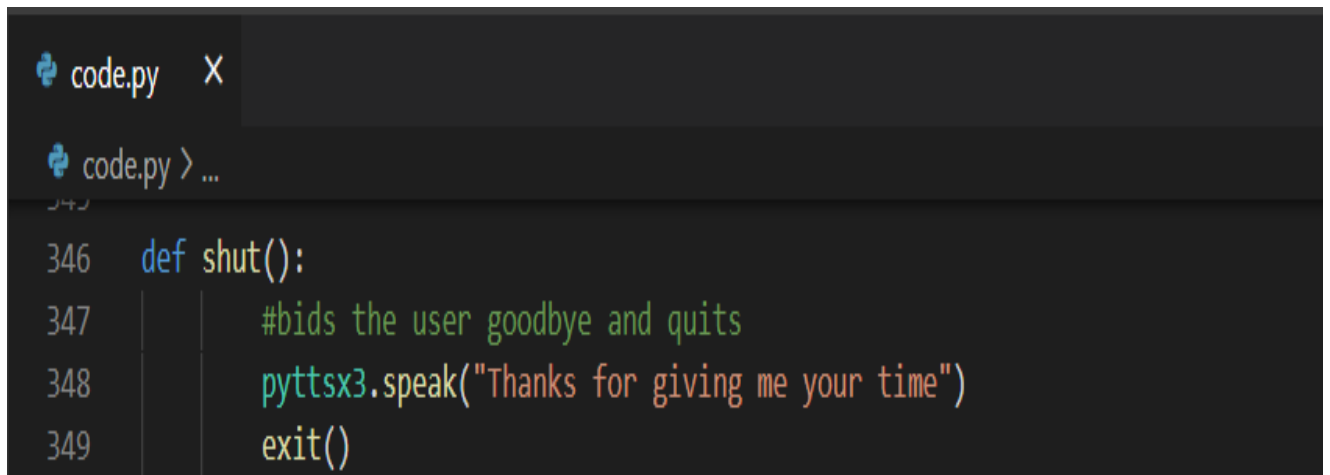
- If TOM finds “from wikipedia” in query then it speaks ‘searching wikipedia...’and it searches the query given by the user in wikipedia. After searching, it reads the summarized result by starting it with “According to wikipedia”.
- If TOM finds “open wikipedia” in query then it speaks ‘here you go to wikipedia’ and then opens wikipedia.

- If TOM finds “open youtube” in query then it speaks ‘here you go to youtube’ and then opens youtube.
- If TOM finds “open google” in query then it speaks ‘here you go to google’ and then opens google.
- If TOM finds “open stackoverflow” in query then it speaks ‘here you go to stack overflow. Happy coding’ and then opens stack overflow.
- If TOM finds “open gmail” in query then it speaks ‘here you google mail open now’ and then opens gmail.
- If TOM finds “time” in query then it speaks ‘current time is’ and tells the time.
- If TOM finds “email to charan” then it asks the user ‘what should i say?’ and takes content from the user, then sends the mail. After sending the mail it speaks “Email has been sent!”. If it cannot send the mail, it speaks “I am not able to send this Email”.
- If TOM finds “send a mail” then it asks the user ‘what should i say?’ and then it speaks “whom should I send?” and then takes content and recipient from the user, then sends the mail. After sending the mail it speaks “Email has been sent!”. If it cannot send the mail, it speaks “I am not able to send this Email”.
- If TOM finds “how are you” in query it replies “I am fine, Thank you” and then asks the user “How are you?”.
- If TOM finds “fine” or “good” in query then it replies “It’s good to know that you are fine”.
- If TOM finds “change my name to” in query then it replaces the name of the user with the current name given by user.
- If TOM finds “change name” in query then it asks “what would you like to call me?” and takes input from the user, then it replies “Thanks for naming me”.
- If TOM finds “what’s your name” or “what is your name”, it speaks “my friends call me TOM”.

- If TOM finds “exit” in query, it speaks “Thanks for giving me your time” and exits.
- If TOM finds “who made you” or “who created you” in the query, it speaks “I have been created by vaishnavi,mohsina and haricharan”.
- If TOM finds “joke” in query, it starts telling some random jokes.
- If TOM finds “search” or “play” in query then it searches the query in the web browser and opens the browser with the searched result.
- If TOM finds “who I am” in query then it tells “if you talk then definitely you are human”.
- If TOM finds “who are you” in query then it tells “I am your virtual assistant”.
- If TOM finds “reason for you” in query, it tells “I was created as a minor project”.
- If TOM finds “change background” in query, it searches for wallpapers in the system and then changes the background. After changing the background it speaks “background changed successfully”.
- If TOM finds “news” in query it opens Times of India in browser and gives the current headlines and then says “here are some headlines from Times of India, happy reading”.
- If TOM finds “lock window” in query then it speaks “locking the device” and then locks the device.
- If TOM finds “shut down system” in query then it speaks “hold on a second! Your system is on its way to shut down” and then shuts down the system.
- If TOM finds “empty recycle bin” in query, it empties the recycle bin and tells “recycle bin recycled”.

- If TOM finds “don’t listen” or “stop listening” in query then it asks “for how much time you want to stop TOM from listening commands” and then sleeps for a specific amount of time given by user.
- If TOM finds “where is” in query then it speaks “user asked to locate, location name “ and locates that place in the google maps.
- If TOM finds “restart” in query then it restarts the system.
- If TOM finds “hibernate” or “sleep” in query, it speaks “hibernating” and then sleeps.
- If TOM finds “logoff” or “signout” in query then it speaks “make sure all the applications are closed before signout” and then logs out.
- If TOM finds “write a note” in query, it asks user “what should I write?” and takes content from the user and writes it in a text document. Then it asks “should I include date and time?”, if user says “yes” it will include date and time otherwise it won’t include.
- If TOM finds “show note” or “open note” in query, it speaks “showing note” then opens the text document and reads it.
- If TOM finds “TOM” in query it wishes the user and speaks “TOM at your service”.
- If TOM finds “current weather” in query, it gives current weather based on user’s location and speaks “here is what I found on google”.
- If TOM finds “send message” in query then it sends the message to the person specified by the user.
- If TOM finds “good morning” in query then it says “a warm good morning” and asks “how are you, username”.
- If TOM finds “will you be my bf” in query then it speaks “I am not sure about that, may be you should give me some time”.

- If TOM finds “I love you” in query then it replies “its hard to understand”.
- If TOM finds “add” in query, it adds the numbers given by the user and then tells the result.
- If TOM finds “subtract” in query, it subtracts the numbers given by the user and then tells the result.
- If TOM finds “multiply” in query, it multiplies the numbers given by the user and then tells the result.
- If TOM finds “divide” in query, it divides the numbers given by the user and tells the result.



```
code.py X
code.py > ...
346 def shut():
347     #bids the user goodbye and quits
348     pyttsx3.speak("Thanks for giving me your time")
349     exit()
```

Fig 6.1 (h): shut

When this method is called, TOM speaks “Thanks for giving me your time.” And then exits.

```
code.py X
code.py > ...
350 root=Tk()
351 root.geometry("800x1000")
352 root.configure(bg="#EEE8AA")
353 root.title("VOICE ASSISTANT")
354
355 label1=Label(root,text="TOM",font=("Arial Bold",60),bg="#EEE8AA",fg="Black").pack(padx=10,pady=30)
356
357 canvas=Canvas(root,width=500,height=500,bg="#EEE8AA")
358 img=PhotoImage(file="va-icon.jpg")
359 canvas.create_image(250,250,anchor=CENTER,image=img)
360 canvas.pack(padx=10,pady=30)
361
362 global b1
363 b1=Button(root,text="Tap to Speak",bg="Black",fg="white",font='bold',width=30, command=click)
364 b1.pack(padx=10,pady=10)
365
366 b2=Button(root,text="Exit",bg="Black",fg="white",font='bold',width=30, command=shut)
367 b2.pack()
368
369
370 root.mainloop()
```

Fig 6.1 (i):front end

The above code is used to build the GUI for the voice assistant TOM. It contains a picture of voice assistant and two buttons where first button is “Tap to speak” and second is “Exit”. Here, we use canvas and photo image functions to create image in the GUI and we use button function to create button. We used label for the title on the GUI. When user clicks on “Tap to speak” button then click() function is called. When user clicks on “Exit” button then shut() function is called.

7. TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

7.1 TESTING STRATEGY

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-ordinate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding.

7.1.1 SYSTEM TESTING

Software once validated must be combined with other system elements (e.g. Hardware, people and database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

7.1.2 UNIT TESTING

In unit testing different modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to

uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module. In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this.

We have performed unit testing of our project which satisfy the requirements. We have tested each and every module which is satisfactorily as regards to the expected output from the module.

7.2 TEST CASES

7.2.1 Test Case 1

Test Case 1: open google

Test Objective: To open google search engine when user asks.

Test Description: google must be opened.

Requirements Verified: Yes.

Test Environment: The project can run in any python IDE's and must have proper internet connection.

Test Setup/Pre-Conditions:

Actions: User must ask tom to open google.

Expected Results: Successfully opens google.

Pass: Yes

Conditions pass: Yes

Fail: No

Problems / Issues: NIL

Notes: Successfully Executed

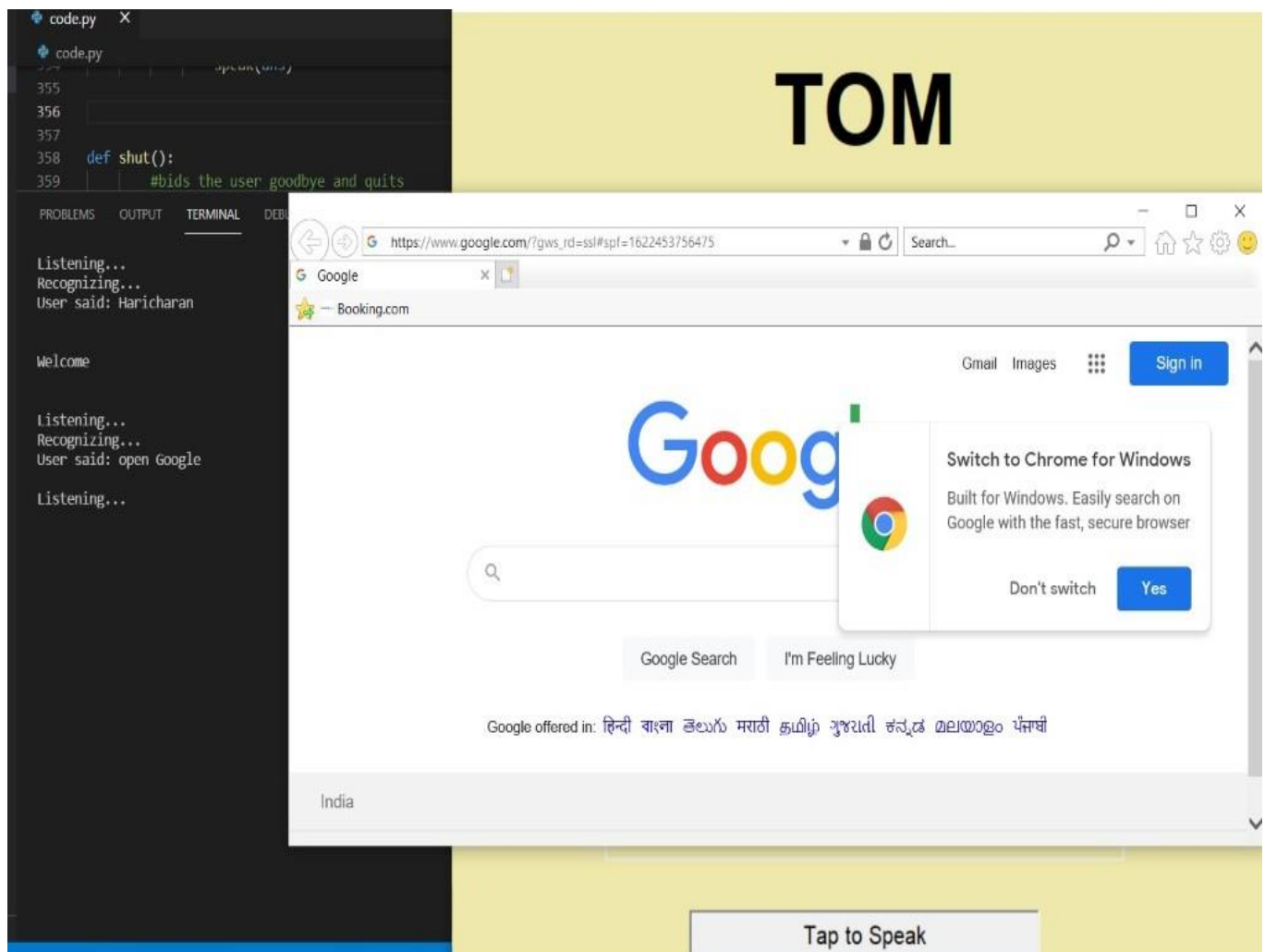


Fig 7.2.1: testcase(google)

7.2.2 Test Case 2

Test Case 2: open YouTube

Test Objective: To open YouTube when user asks.

Test Description: YouTube must be opened.

Requirements Verified: Yes.

Test Environment: The project can run in any python IDE's and must have proper internet connection.

Test Setup/Pre-Conditions:

Actions: User must ask tom to open YouTube.

Expected Results: Successfully opens YouTube.

Pass: Yes

Conditions pass: Yes

Fail: No

Problems / Issues: NIL

Notes: Successfully Executed

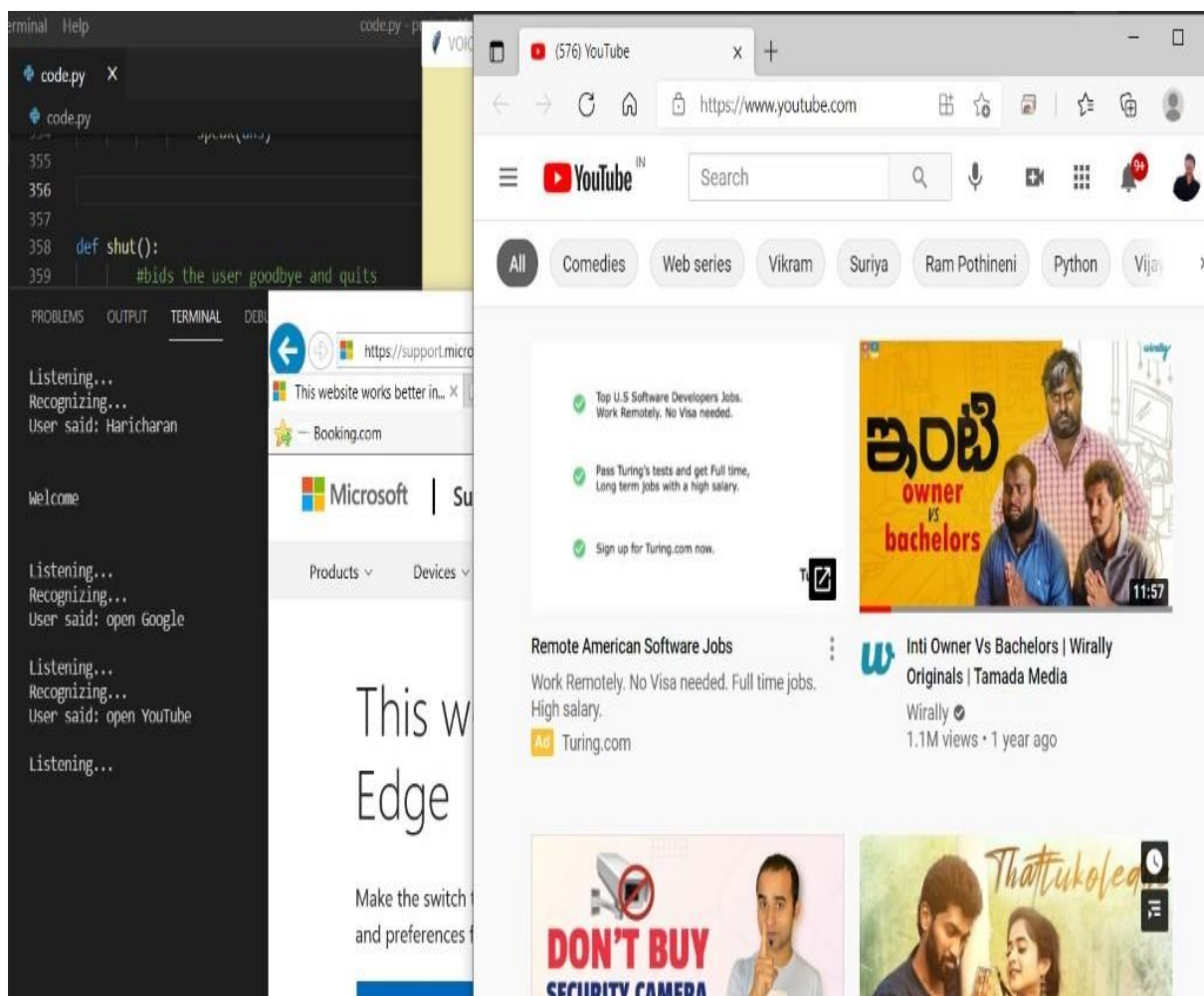


Fig 7.2.2: testcase(YouTube)

7.2.3 Test Case 3

Test Case 3: Search for a place

Test Objective: To search a place and show its location on google maps when user asks.

Test Description: Location of the searched place must be shown in google maps.

Requirements Verified: Yes.

Test Environment: The project can run in any python IDE's and must have proper internet connection.

Test Setup/Pre-Conditions:

Actions: User must ask tom to search for a place.

Expected Results: Successfully shows the location of a place asked by the user in google maps.

Pass: Yes

Conditions pass: Yes

Fail: No

Problems / Issues: NIL

Notes: Successfully Executed

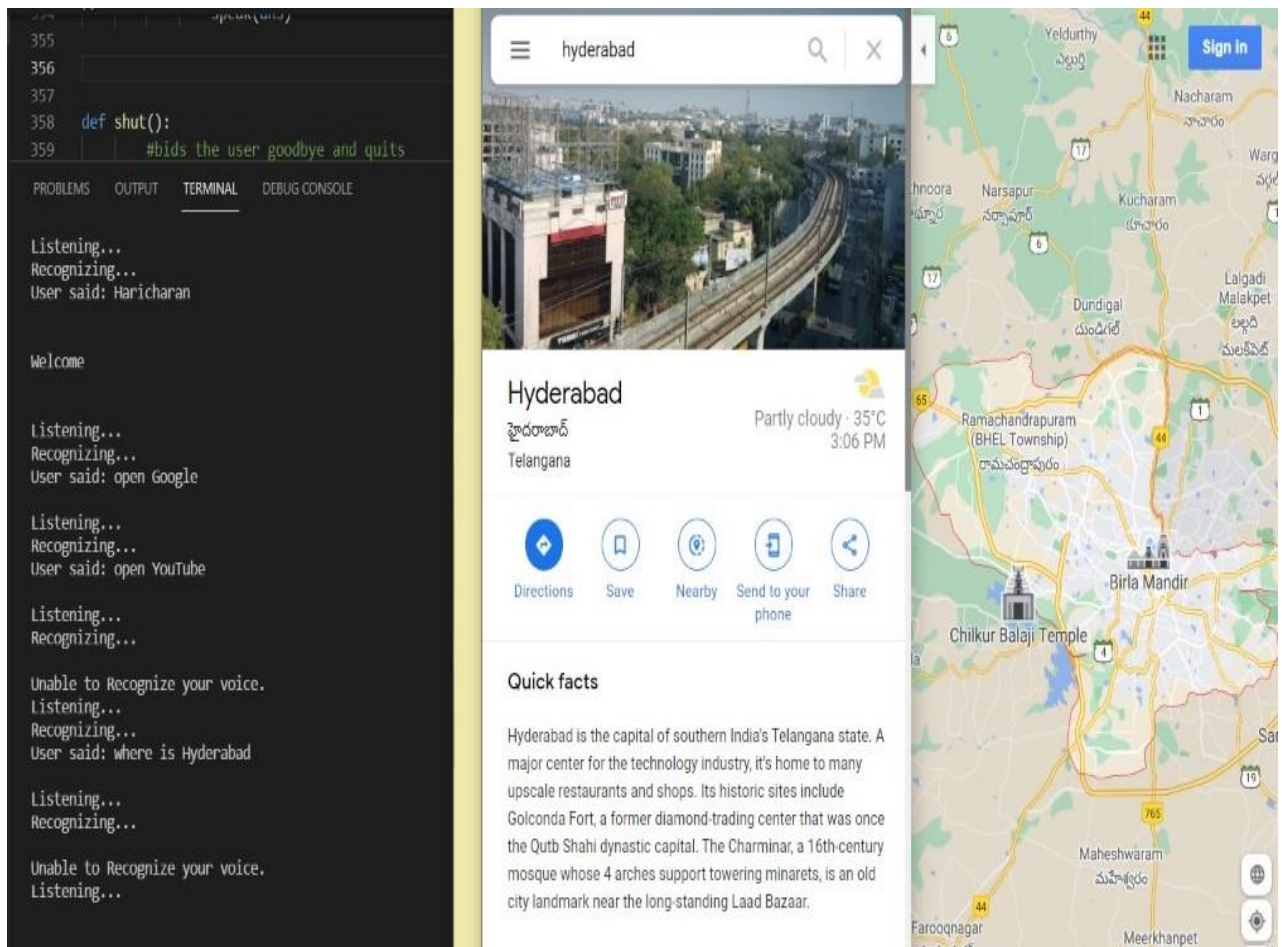


Fig 7.2.3: testcase(search for a place)

7.2.4 Test Case 4

Test Case 4: Gives weather in user's location

Test Objective: To display weather in user's location when user asks.

Test Description: Gives weather in user's current location.

Requirements Verified: Yes.

Test Environment: The project can run in any python IDE's and must have proper internet connection.

Test Setup/Pre-Conditions:

Actions: User must ask tom to give weather report.

Expected Results: Successfully shows the weather report.

Pass: Yes

Conditions pass: Yes

Fail: No

Problems / Issues: NIL

Notes: Successfully Executed

The image shows a terminal window on the left and a Google search result on the right. The terminal window displays a Python script with a function `def shut():` and a comment `#bids the user goodbye and quits`. The terminal output shows the program listening for voice input, recognizing it, and responding with "Welcome", "Listening...", "Recognizing...", and "User said: Haricharan". It then responds with "Welcome", "Listening...", "Recognizing...", and "User said: open Google". It then responds with "Listening...", "Recognizing...", and "User said: open YouTube". It then responds with "Unable to Recognize your voice.", "Listening...", "Recognizing...", and "User said: where is Hyderabad". It then responds with "Unable to Recognize your voice.", "Listening...", "Recognizing...", and "User said: current weather". It then responds with "Listening...".

The Google search result on the right shows the search term "weather" and the location "Vemulawada, Telangana". The current temperature is 38°C (100°F). The search result also shows the weather forecast for the next 7 days, with temperatures ranging from 34°C to 36°C. The search result also includes a link to the AccuWeather website and a section titled "People also ask" with the question "When there will be rain in Kolkata?".

Fig 7.2.4: testcase(Weather)

7.2.5 Test Case 5

Test Case 5: Gives headlines of current news.

Test Objective: To display headlines when user asks.

Test Description: Gives headlines from Times of India.

Requirements Verified: Yes.

Test Environment: The project can run in any python IDE's and must have proper internet connection.

Test Setup/Pre-Conditions:

Actions: User must ask tom to give current news.

Expected Results: Successfully shows the headlines.

Pass: Yes

Conditions pass: Yes

Fail: No

Problems / Issues: NIL

Notes: Successfully Executed

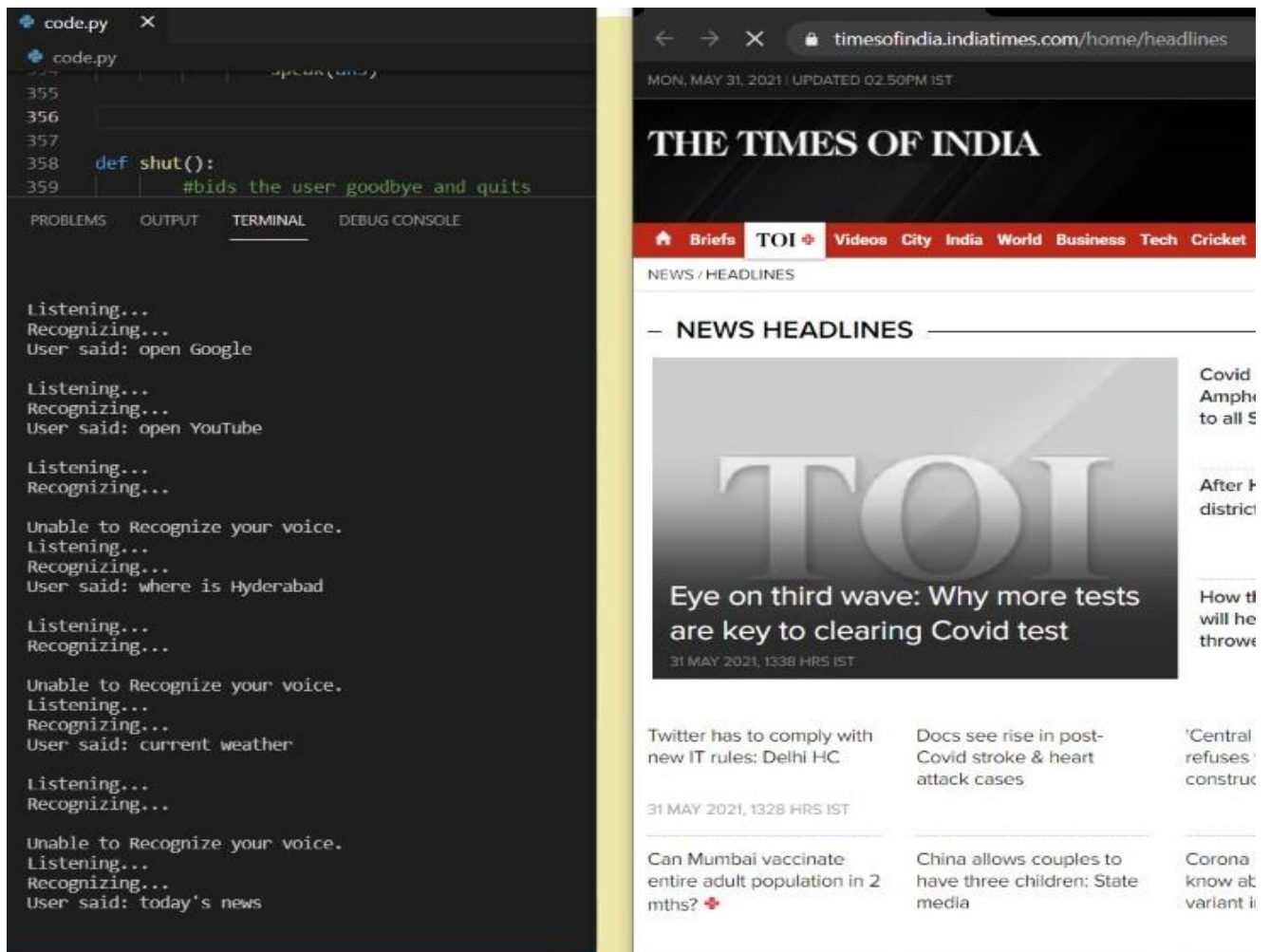


Fig 7.2.5: testcase(news)

8. OUTPUT SCREENS

8.1 Front End:

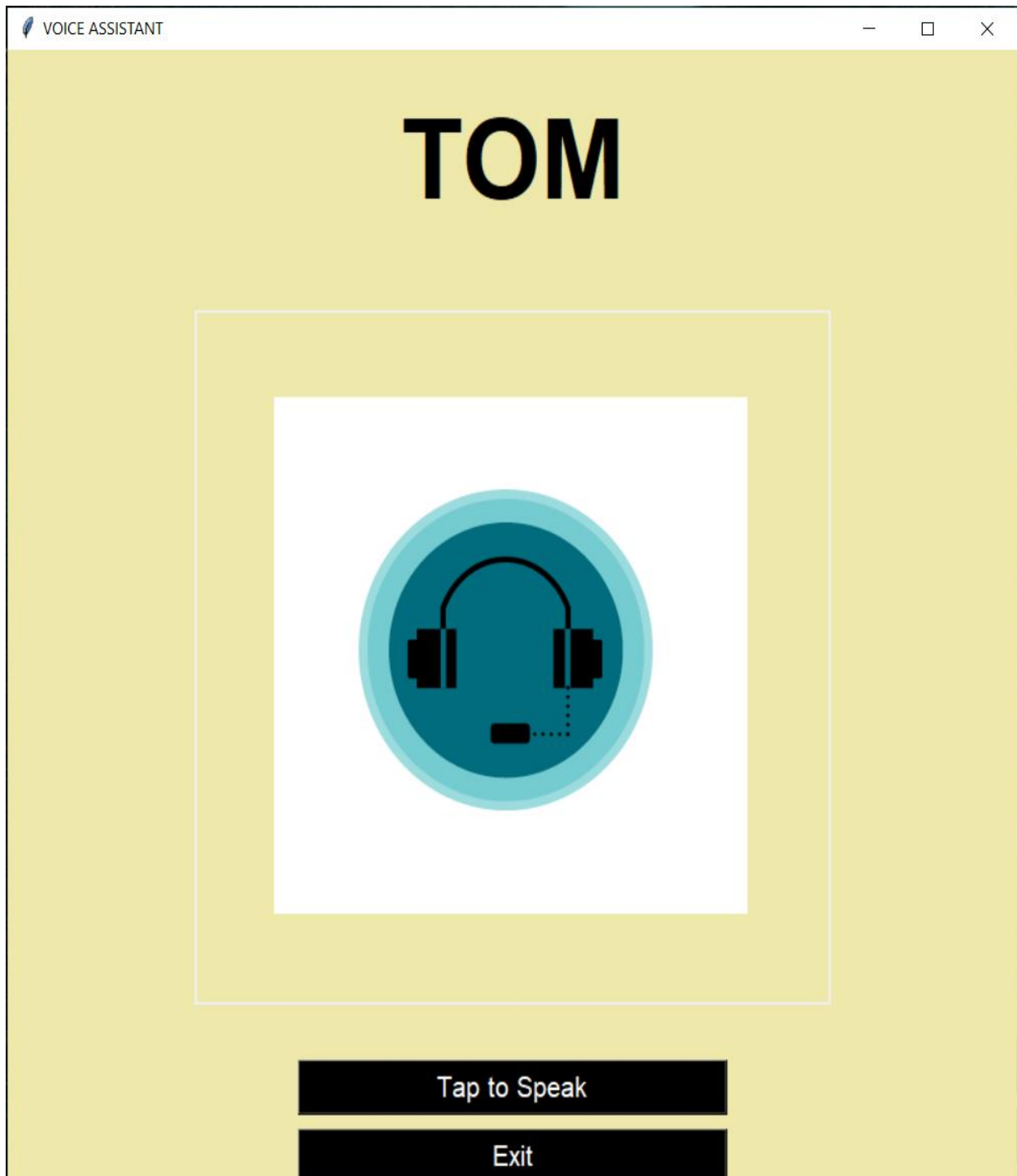


Fig 8.1: frontend

9. CONCLUSION AND FUTURE SCOPE

9.1 Conclusion:

We created a desktop voice assistant that helps the user in his day to day tasks. It performs various tasks such as opening google, searching for places, putting the desktop to sleep etc.

The intended audience of this assistant are students, working professionals and all the other people who work frequently on a desktop.

Benefits:

- Inexpensive
- Can work on all operating systems (desktop)
- Has desktop support

9.2 Future Scope:

- Design Improvements
- Humanized Voice Recognition
- Additional Functions
- Improved Interface
- Integration with ML models to enhance the user experience

10. BIBLIOGRAPHY

10.1 References:

Books:

- Fluent Python: Clear, Concise, and Effective Programming by Luciano Ramalho
- Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming by Eric Matthes

Websites:

- <https://www.geeksforgeeks.org/>
- <https://en.wikipedia.org/wiki/>
- <https://stackoverflow.com/>

YouTube:

- freeCodeCamp.org
- Edureka!
- Heroprogrammer.org