Real-time Research Project(CS456PC)

on

Personal Assistant - chatbot

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

Bachelor of Technology
in
Computer Science and Engineering
by

M.Pavan Kumar - 22261A6740
U.Veman Kumar - 22261A6758
M.Sanjay - 23265A6703
P.Tharun - 23265A6705

Under the guidance of

Mr. M Srikanth Sagar

Ms . B Varalakshmi

Dr . D Koteshwar rao

Ms. D. Deepika

(Assistant Professors)

Department of Computer Science and Engineering
Mahatma Gandhi Institute of Technology
(Affiliated to Jawaharlal Nehru Technological University Hyderabad) Kokapet(V), Gandipet(M), Hyderabad.
Telangana - 500 075.

2023-24

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad) GAN-

DIPET, HYDERABAD – 500075, Telangana

CERTIFICATE

This is to certify that the project entitled **— "PERSONAL ASSISTANT – chatbot"** is being submitted by **M.Pavan(22261A6740), U.Veman(22261A6758), M.Sanjay(23265A6703) and P.Tharun(23264A6705)** in partial fulfillment of the requirements for the Real-time Research Project (CS456PC) in **COMPUTER SCIENCE AND ENGINEERING** is a record of bonafide work carried out by us.

The results of the investigations enclosed in this report have been verified and found satisfactory.

PROJECT GUIDE

Mr. M Srikanth Saga
(Assistant Professor)

Ms . B Varalakshmi
(Assistant Professor)

Dr . D Koteshwar rao
(Assistant Professor)

Ms. D. Deepika
(Assistant Professor)

HEAD OF THE DEPARTMENT

Dr.M.Rama Bai
(Professor & HOD of ET)

External Examiner

DECLARATION

This is to certify that the work reported in this project titled – **"PERSONAL ASSISTANT – chatbot"** is a record of work done by us in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by me and not copied from any other source.

M.Pavan kumar - 22261A6740

U.Veman kumar - 22261A6758

M.Sanjay - 23265A6703

P.Tharun - 23265A6705

ACKNOWLEDGEMENTS

We would like to express my sincere thanks to **Dr.G.Chandra Mohan Reddy, Principal MGIT**, for providing the working facilities in college.

We wish to express my sincere thanks and gratitude to **Dr.M.Rama Bai, Professor and HOD**, Department of CSE, MGIT, for all the timely support and valuable suggestions during the period of project.

We are extremely thankful to **Ms. D. Deepika, Assistant Professor** and and **Mr. Mallela Srikanth, Assistant Professor, Mr. K. Vikas, Assistant professor,** Department of ET, MGIT, real time project coordinators for their encouragement and support throughout the project.

We are extremely thankful and indebted to my internal guide **Mr. M Srikanth Saga , Ms . B Varalakshmi , Dr . D Koteshwar rao , Ms. D. Deepika** Department of ET, for their constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank all the faculty and staff of CSE Department who helped me directly or indirectly, for completing this project.

M.Pavan kumar - 22261A6740

U.Veman kumar - 22261A6758

M.Sanjay - 23265A6703

P.Tharun - 23265A6705

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In today's fast-paced world, where time is a precious commodity, the need for efficient and intelligent personal assistants has become increasingly prevalent. This project aims to develop a comprehensive personal assistance chatbot leveraging the power of Python programming language and various cutting-edge libraries. The chatbot is designed to streamline daily tasks, enhance productivity, and provide a seamless user experience by offering a wide range of functionalities.

This project aims to develop a comprehensive personal assistance chatbot capable of performing a wide range of tasks and functionalities using Python programming language and various libraries. The chatbot incorporates features such as playing games (e.g., Rock Paper Scissors with a graphical user interface), searching Wikipedia and Google Maps, playing videos from YouTube, sending emails and WhatsApp messages, providing weather information, telling jokes, and delivering news updates. Additionally, the chatbot offers enhanced security through face unlock authentication, photo capture capabilities, and mathematical calculations. It can perform timer operations, display in-built search images, provide smart dictionary searches, and retrieve operating system and battery information. Furthermore, it includes providing directions on maps and taking screenshots. The project's goal is to create an intelligent and versatile personal assistant that can streamline daily tasks, enhance productivity, and provide a seamless user experience.

1. INTRODUCTION

In the modern fast-paced world, where time is a precious commodity, the need for efficient and intelligent virtual assistants has become increasingly prevalent. Personal digital assistants can greatly enhance productivity by streamlining daily tasks, providing quick access to information, and automating repetitive activities. This project aims to develop a comprehensive personal assistance chatbot leveraging the power of Python programming language and various state-of-the-art libraries.

The chatbot is designed to be a versatile virtual companion capable of performing a wide range of functionalities, from entertainment and gaming to information retrieval, communication, and utility tools. At its core, the personal assistant chatbot is powered by Python, a highly versatile and widely-adopted programming language known for its simplicity, readability, and extensive library ecosystem. The project harnesses the capabilities of various Python libraries and frameworks to implement the desired features and functionalities, ensuring a seamless and efficient integration of different components.

The chatbot's capabilities include playing games with a graphical user interface, searching Wikipedia and Google Maps, playing videos from YouTube, sending emails and WhatsApp messages, providing weather information, telling jokes, and delivering news updates. Additionally, it offers enhanced security through face unlock authentication, photo capture capabilities, and mathematical calculations. The chatbot can perform timer operations, display in-built search images, provide smart dictionary searches, and retrieve operating system and battery information. Furthermore, it includes providing directions on maps and taking screenshots.

The project's ultimate goal is to create an intelligent and versatile personal assistant that can enhance user productivity, streamline daily routines, and provide a seamless and engaging user experience. By leveraging the power of Python and its vast ecosystem of libraries, the personal assistant chatbot aims to push the boundaries of what virtual assistants can achieve, paving the way for more advanced and capable digital companions in the future.

1.1. Motivation

In our busy daily lives, we often find ourselves juggling multiple tasks and responsibilities. From checking the weather forecast to scheduling appointments, looking up information online, or even just trying to stay entertained - there are so many little things that take up our time and attention. Wouldn't it be great to have a personal assistant that could handle all these minor tasks for us? That's the motivation behind this project - to create a smart, capable virtual assistant that can make our lives easier and more convenient.

1.2 Problem Definition

The main problem we are trying to solve is how to develop a single, unified digital assistant that can understand and respond to our needs across various domains. We want an assistant that can play games with us for fun, look up information from sources like Wikipedia or maps, send messages on our behalf, give us weather updates, and even perform utility tasks like calculations or setting timers.

Additionally, we want this assistant to have some advanced capabilities like face recognition for secure access, the ability to capture photos or screenshots, and integrate with our computer's operating system to retrieve useful information.

In essence, we are aiming to create an intelligent chatbot that can comprehend natural language, understand the context of our requests, and then utilize different technologies and data sources to provide relevant responses or take appropriate actions. The goal is to have a multipurpose virtual companion that can streamline our daily routines, enhance productivity, and provide a seamless, engaging user experience. By combining the power of the Python programming language,computer vision, web automation, and more, we hope to develop a personal assistant chatbot that can tackle this multifaceted problem and become an indispensable part of our daily lives.

1.3 Existing System

In the current scenario, personal assistants are primarily software-based applications that rely on text-based input or predefined voice commands. These assistants have limited functionality and are often restricted to specific tasks or applications. Some key limitations of existing systems include:

- Lack of multimodal interaction: Most existing systems rely solely on voice input, limiting their ability to process visual information or combine multiple input modalities for more intuitive interactions.
- Restricted functionality: Existing assistants are often designed for specific tasks or domains, such as web searches, setting reminders, or controlling smart home devices, but lack the flexibility to handle a wide range of user requests.

DISADVANTAGES :

- Limited functionality and customization options.
- Difficulties in understanding complex or context dependent voice commands.
- Inability to process multi-modal inputs (example:visual data).
- Lack of personalization and adaptation to individual user needs.

1.4 Proposed System

The proposed personal assistance chatbot system aims to address the limitations of existing systems by incorporating advanced technologies such as natural language processing, computer vision, and machine learning. The key features of the proposed system include:

- Expanded functionality: The system will be designed to handle a wide range of tasks and user requests, from playing games and navigating maps to opening folders and performing system operations, providing a versatile and flexible personal assistant.
- Personalization and adaptation: The system will learn and adapt to individual user preferences, habits, and contexts over time, providing a personalized experience tailored to each user's needs.

ADVANTAGES :

- Improved understanding and response to complex voice commands.
- Ability to process multi-modal inputs.
- Expanded functionality and versatility for various tasks.
- Personalized experience through adaptation to individual user preferences and contexts.

1.5 Requirements Specification

1.5.1 Software Requirements

- Language : Python 3.6
- Operating system : Windows / Linux
- IDE : Pycharm

1.5.2 Hardware Requirements

- Processor : 13/Intel Processor
- RAM : 4GB (min)
- Hard Disk : 128GB

1.5.1.1 Python

Python is a high-level, interpreted, interactive and object-oriented scripting language created by Guido Rossum in 1989. Python is designed to be highly readable. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large- scale projects. It is ideally designed for rapid prototyping of complex applications.

It has interfaces to many OS system calls and libraries and is extensible to C or C++. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science.

History of Python

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language, capable of exception handling and interfacing with the Amoeba operating system. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but

now shares his leadership as a member of a five-person steering council.

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 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Features of Python :

Python's features include:

- **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.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- 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.

SOFTWARE INSTALLATION FOR MACHINE LEARNING PROJECTS:

Installing Python:

1. To download and install Python visit the official website of Python https://www.python.org/downloads/ and choose your version.

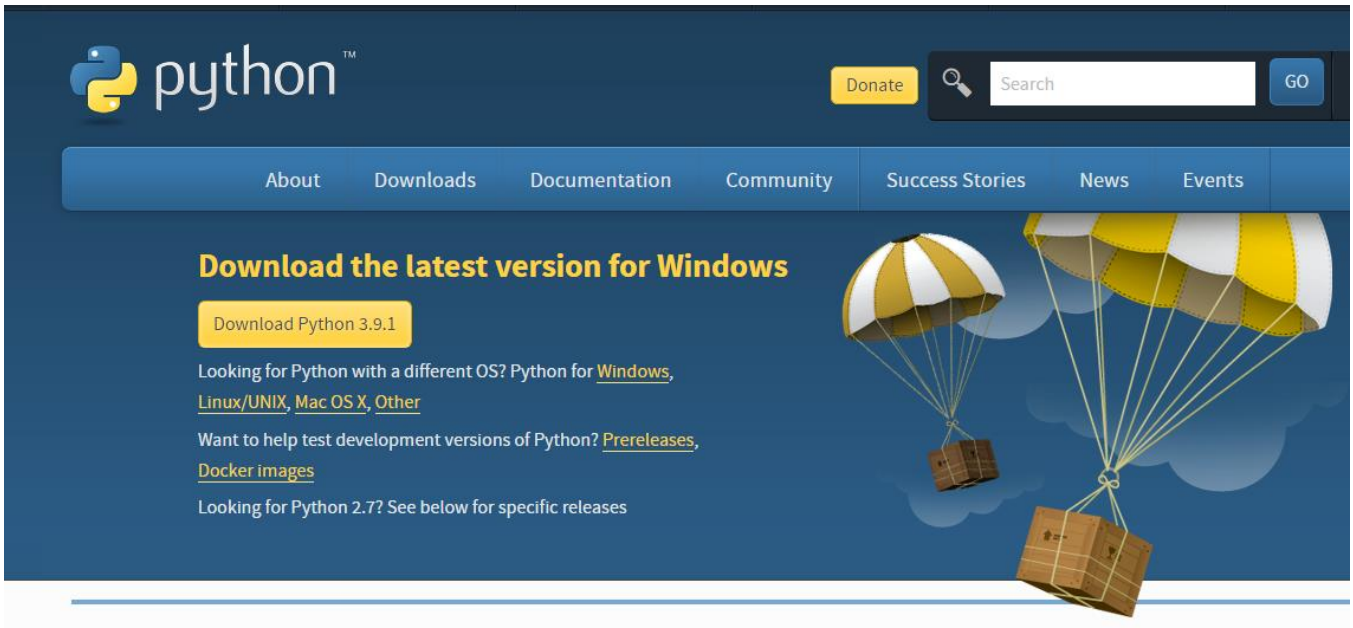

Figure 1.1 Home page of python

2. Once the download is complete, run the exe for install Python. Now click on Install Now.

3. You can see Python installing at this point.

4. When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".

Installing PyCharm:

1. To download PyCharm visit website https://www.jetbrains.com/pycharm/download/ and Click the "DOWNLOAD" link under the Community Section.

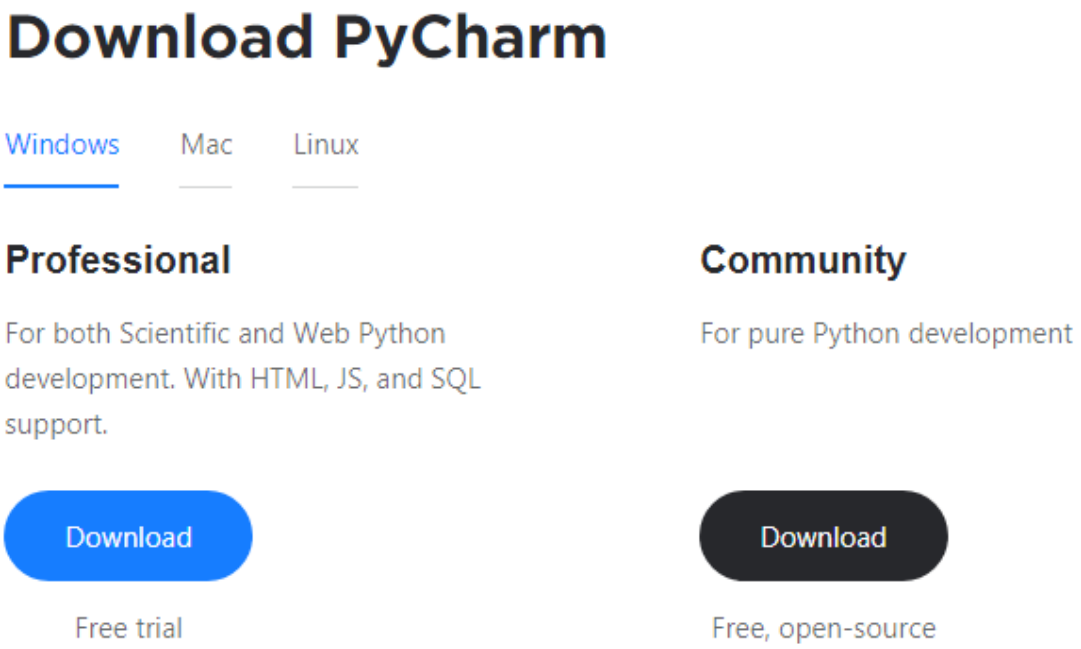

Figure 1.2 Pycharm downloads page

2. Once the download is complete, run the exe for install PyCharm. The setup wizard should have started. Click “Next”.
3. On the next screen, Change the installation path if required. Click “Next”.
4. On the next screen, you can create a desktop shortcut if you want and click on “Next”.
5. Choose the start menu folder. Keep selected JetBrains and click on “Install”.
6. Wait for the installation to finish.
7. Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.

8. After you click on "Finish," the Following screen will appear.

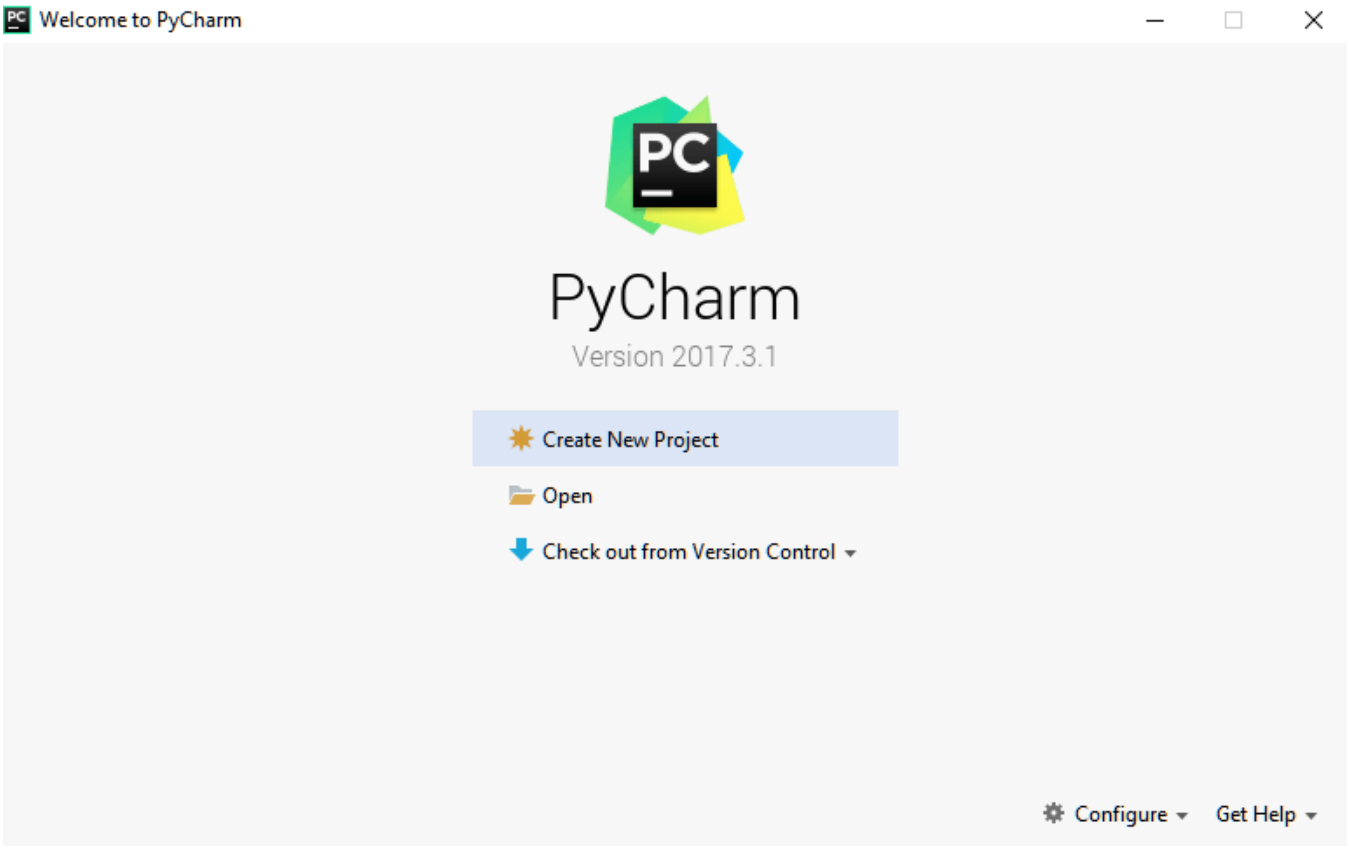

Figure 1.3 Pycharm setup window

9. You need to install some packages to execute your project in a proper way.

10. Open the command prompt/ anaconda prompt or terminal as administrator.

11. The prompt will get open, with specified path, type “pip install package name” which you want to install (like numpy, pandas, seaborn, scikit-learn, matplotlib.pyplot)

Ex: pip install numpy

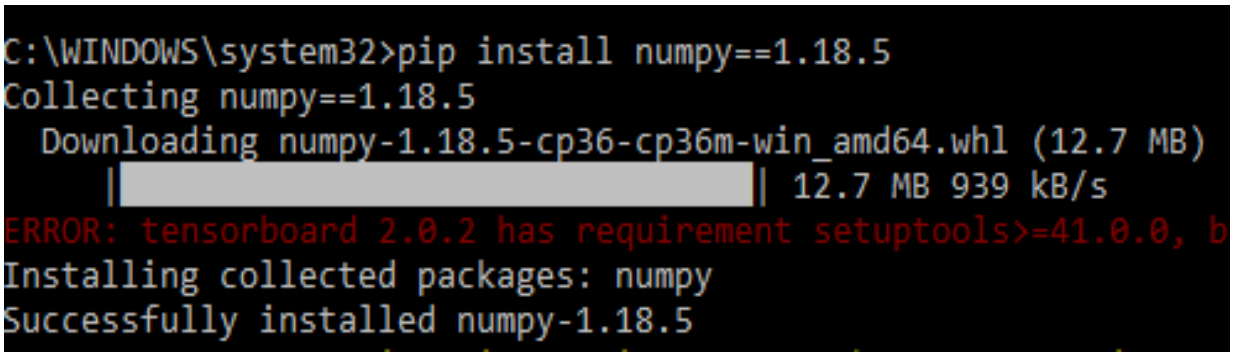

Figure 1..4 module numpy installation

2. Literature Survey

[1] Harhs Pariyani, Anushika Sinha,Preeti Bhat and Prof. N.A. Mulla published a paper title-AI-based system for conducting e-interviews and evaluating candidates. The system uses natural language processing, speech recognition, and facial emotion detection to automate parts of the recruitment process. It allows candidates to take one-way video interviews at their convenience, which are then analyzed by the AI to assess their responses and reactions. The system aims to make recruiting more efficient, fair, and data-driven.

[2] Takaaki Kobayashi,Yuka Nishina,Hana Tomoi published a paper title- "Corowa-kun: Impact of a COVID-19 vaccine information chatbot on vaccine hesitancy, Japan 2021," examines the effect of a COVID-19 vaccine information chatbot on vaccine hesitancy in Japan. The authors created a chatbot, Corowa-kun, in the popular messaging app LINE, which provided instant, automated answers to frequently asked COVID-19 vaccine questions. A cross-sectional survey assessed COVID-19 vaccine hesitancy among Corowa-kun users.

[3] Mini-Chang Sung The research paper "A study on the wh-questions used by an English education chatbot: Based on the language forms in the 2022 revised national curriculum of English" explores the effectiveness of a chatbot in teaching English to students, focusing on the use of wh-questions (what, where, when, why, how) in the context of the 2022 revised national curriculum of English.

[4] Nina van Zanten The article presents a study on a Voting Assistant Chatbot designed to increase voter turnout at local elections. The chatbot is a cost-effective and reliable electronic voting system (EVS) that addresses the limitations of manual elections. It features a secure One-Time Password (OTP) authentication mechanism and a user-friendly interface for both election directors and voters.

[5] Matthew Chase The authors, Rodriguez and Mune, wrote a case study titled "Uncoding library chatbots: Deploying a new virtual reference tool at the San Jose State University Library". They examined the use of an AI-powered chatbot for reference services at a university library.

S.NO:	YEAR	AUTHOR	TITLE	TECHNIQUES	AD-VANTAGES	DISAD-VANTAGES
1	2020	Harhs Pariyani, Anushika Sinha,Preeti Bhat and Prof. N.A.	AI-based system for conducting e-interviews and evaluating candidates	OpenCV for image processing, Sentiment analysis of text responses, Speech recognition to convert audio to text	1. Convenience for candidates (can interview from home, anytime) 2. Automated screening of large numbers of candidates	1. Requires good internet connection, camera, and microphone 2. Speech recognition may be inaccurate in noisy environments 3. May not capture nuances of human interaction
2	2021	Takaaki Kobayashi, Yuka Nishina., Hana Tomoi	Corowa-kun: Impact of a COVID-19 vaccine information chatbot on vaccine hesitancy	Cross Sectional survey, Logistic regression, Chatbot devlopment	1. This study is one of the first to assess the impact of a COVID-19 vaccine information chatbot on vaccine hesitancy 2. The study included a large sample size (n=10,192), which increases the generalizability of the findings.	1. The study was limited to users of the LINE app, which may not be representative of the general population. 2. The study used a cross-sectional design, which does not allow for causal inference.

3	2023	Min-Chang Sung	A study on the wh-questions used by an English education chatbot	Natural Language Processing, Machine Learning, Language Modeling	1. The chatbot provides personalized learning experiences for students, catering to their individual needs and learning styles. 2. The chatbot can reach a wider audience, including students with disabilities or those in remote locations.	1. The chatbot may struggle to understand the context of a question or conversation, leading to inaccurate or irrelevant responses. 2. The chatbot's responses may lack the emotional intelligence and empathy of a human teacher, potentially affecting student motivation and engagement.
4	2024	Nina van Zanten	Voting Assistant Chatbot for increasing Voter Turnout at Local Elections	One-Time Password, Electronic Voting System, Chatbot technology	1. The chatbot aims to encourage more people to participate in local elections. 2. The electronic voting system is more cost-efficient than traditional manual elections.	1. The chatbot may not be accessible to voters without internet access or those who are not tech-savvy. 2. The chatbot may not be suitable for areas with limited technological infrastructure.
5	2024	Matthew Chase	Uncoding library chatbots: Deploying a new virtual reference tool at the San Jose State University Library	Chatbot Development Software, Natural Language Processing,User analytics	1. The chatbot extended reference services beyond live hours, making them available to users at any time. 2. The chatbot handled common questions.	1. Only 10% of users engaged the chatbot beyond the initial response. This highlights the need to understand user perceptions and expectations of chatbot services.

Figure 2.1 Comparison of Literature survey.

3 . Chatbot Methodology

3.1 Flowchart

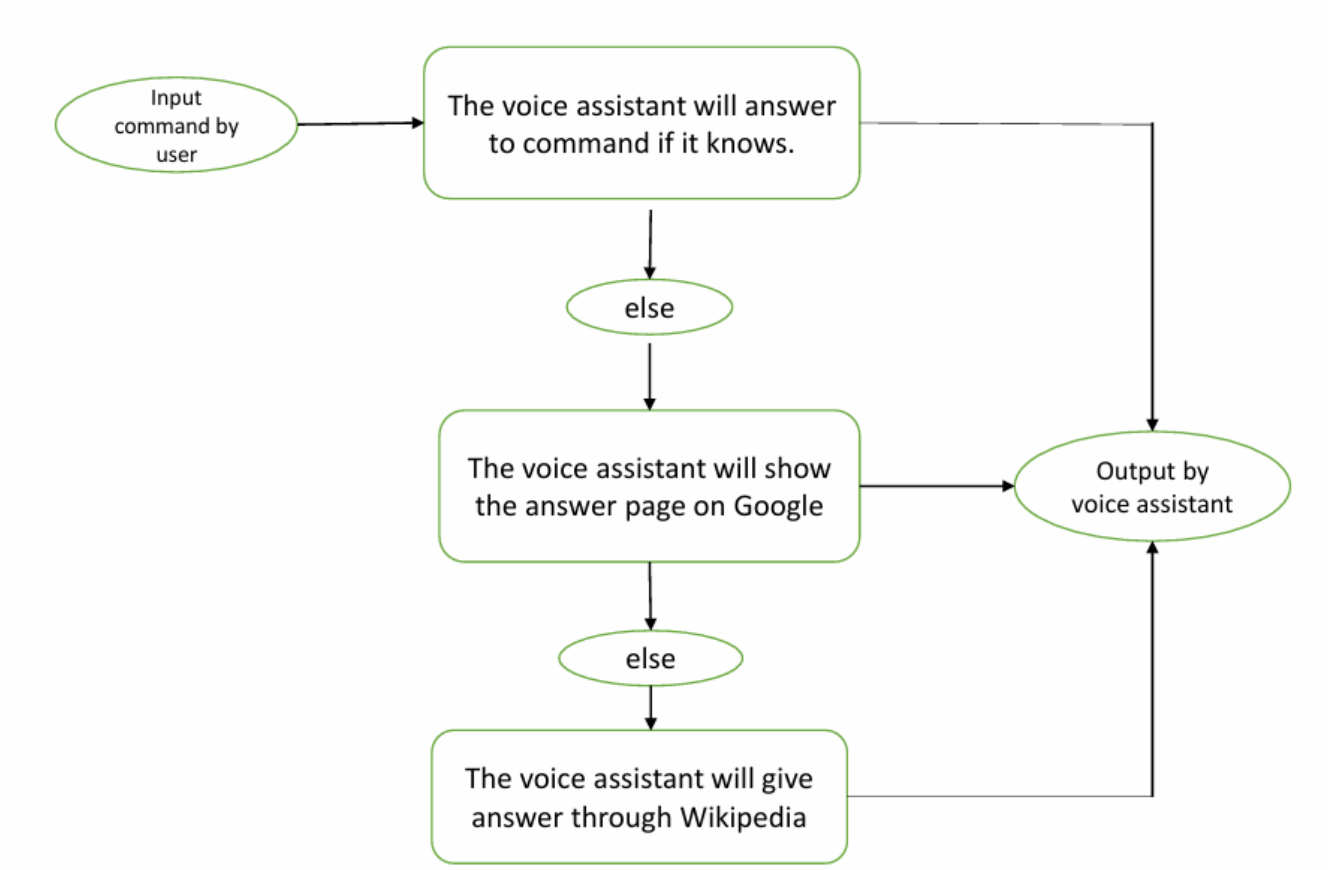

Figure 3.1 Flowchart Diagram

3.2 Working Principle

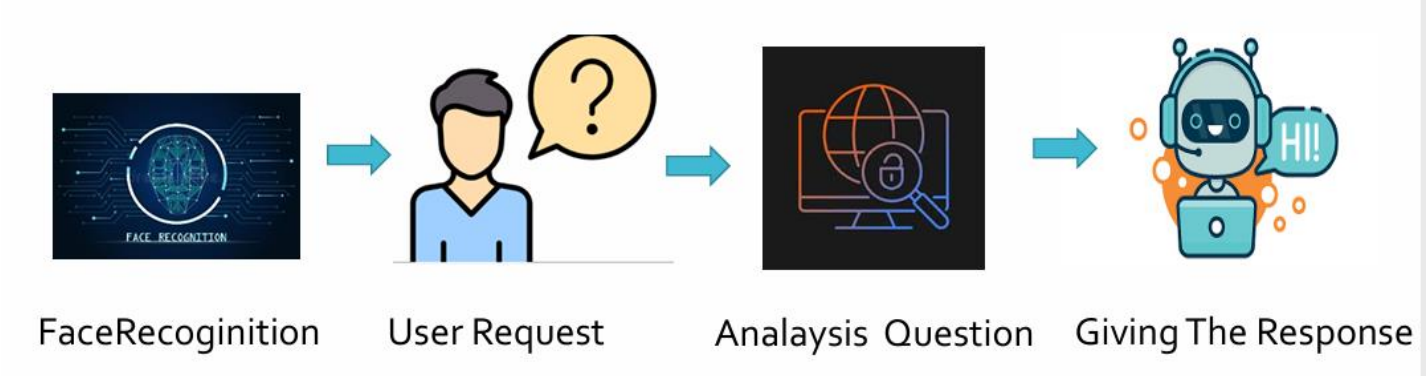

Figure 3.2 Working Principle Diagram

3.3 Use Case Diagram

The Design of Chatbot is represented here:

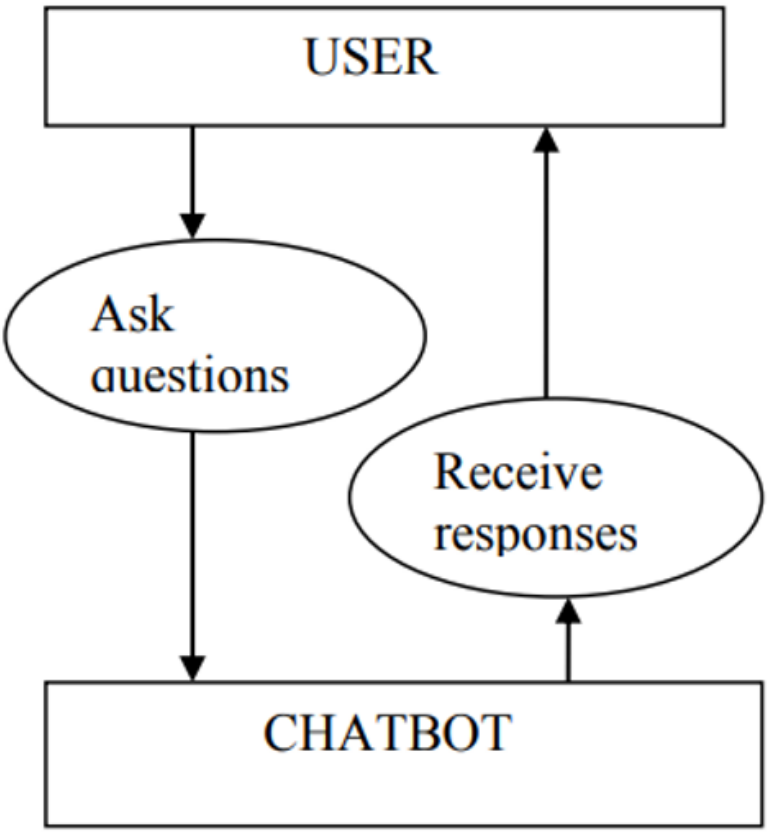

Figure 3.3 Use Case Diagram

3.5 Class Diagram

Class Diagram is a static diagram.It represents the static view of an application. Class Diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

The class user has two attributes command that it sends in audio and the response it receives which is also audio. It performs function to listen the user command. Interpret it and then reply or sends back response accordingly. Question class has the command in string form as it is interpreted by interpret class. It sends it to general or about or search function based on its identification.The task class also has interpreted command in string format.

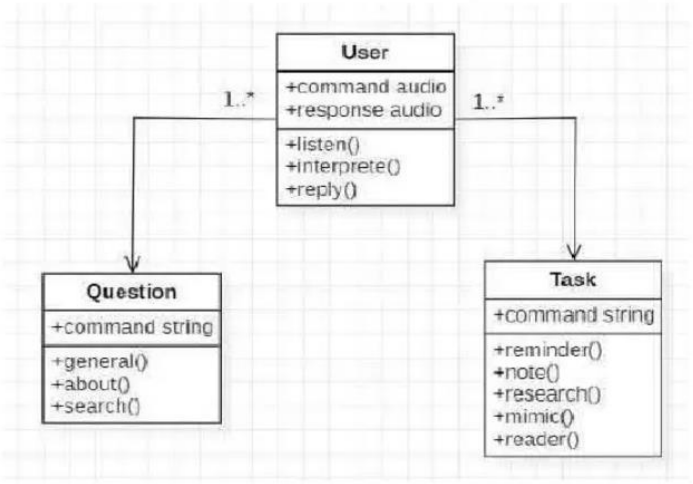

Figure 3.4 Class Diagram

3.6 Sequence Diagram

A sequence diagram is a Unified Model Language(UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.

The below sequence diagram shows how an answer ask by the user is being fetched from internet. The audio query is interpreted and send to Web scraper.The web scraper searches and finds the answer. It is then sent back to speaker, where it speaks the answer to user.

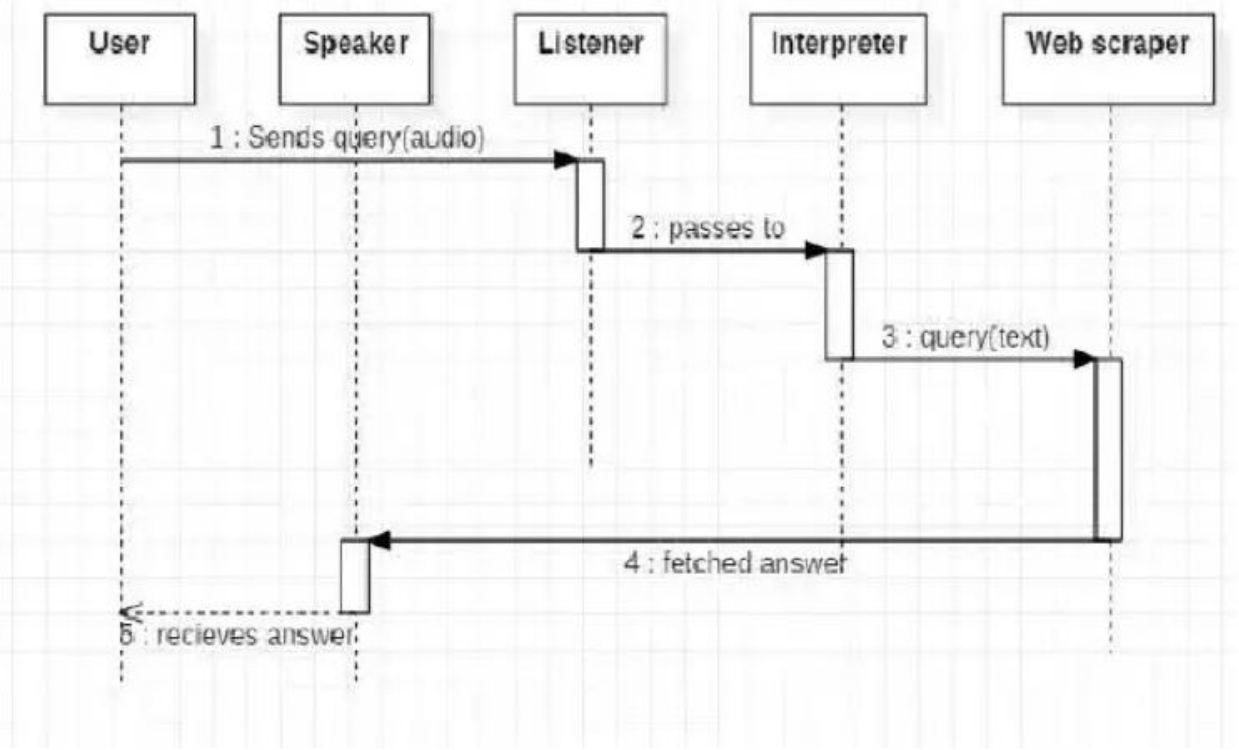

Figure 3.5 Sequence Diagram

4. RESULTS

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

In conclusion, the development of a comprehensive personal assistance chatbot using Python programming language and various cutting-edge libraries has been successfully achieved. The chatbot offers a wide range of functionalities, including playing games, searching Wikipedia and Google Maps, playing videos from YouTube, sending emails and WhatsApp messages, providing weather information, telling jokes, and delivering news updates. Additionally, the chatbot incorporates enhanced security features, such as face unlock authentication, photo capture capabilities, and mathematical calculations. The chatbot's ability to perform timer operations, display in-built search images, provide smart dictionary searches, and retrieve operating system and battery information makes it a versatile and intelligent personal assistant.

The project's goal of creating a seamless user experience and enhancing productivity has been achieved through the chatbot's user-friendly interface and efficient task management capabilities. The chatbot's ability to streamline daily tasks and provide a personalized experience makes it an ideal tool for individuals seeking to optimize their daily routines.

5.2 Future work

While the chatbot has achieved its primary objectives, there are several areas that can be explored for future development and improvement:

- **Integration with IoT devices : i**ntegrating the chatbot with IoT devices, such as smart home devices, can enable users to control their devices remotely and receive real-time updates.
- **Enhanced Natural Language Processing :** Improving the chatbot's natural language processing capabilities can enable it to better understand user requests and provide more accurate responses.
- **Cloud-based Deployment :** Deploying the chatbot on cloud-based platforms can enable scalability and accessibility from anywhere, at any time.
- **Continuous learning and improvement :** Implementing machine learning algorithms can enable the chatbot to learn from user interactions and improve its performance over time.

By exploring these areas, the chatbot can continue to evolve and improve, providing an even more and personalized experience for user.

APPENDIX

app_control.py

```
import pyscreenshot as ImageGrab
import time
import subprocess
from pynput.keyboard import Key, Controller
import psutil

class SystemTasks:
def __init__(self):
self.keyboard = Controller()

def openApp(self, appName):
appName = appName.replace('paint', 'mspaint')
appName = appName.replace('wordpad', 'write')
appName = appName.replace('word', 'write')
appName = appName.replace('calculator', 'calc')
try: subprocess.Popen('C:\\Windows\\System32\\'+appName[5:]+'.exe')
except: pass

def write(self, text):
text = text[5:]
for char in text:
self.keyboard.type(char)
time.sleep(0.02)

def select(self):
self.keyboard.press(Key.ctrl)
self.keyboard.press('a')
self.keyboard.release('a')
```

```
self.keyboard.release(Key.ctrl)

def hitEnter(self):
self.keyboard.press(Key.enter)
self.keyboard.release(Key.enter)

def delete(self):
self.keyboard.press(Key.backspace)
self.keyboard.release(Key.enter)

def save(self, text):
if "don't" in text:
self.keyboard.press(Key.right)
else:
self.keyboard.press(Key.ctrl)
self.keyboard.press('s')
self.keyboard.release('s')
self.keyboard.release(Key.ctrl)
self.hitEnter()

class TabOpt:
def __init__(self):
self.keyboard = Controller()

def switchTab(self):
self.keyboard.press(Key.ctrl)
self.keyboard.press(Key.tab)
self.keyboard.release(Key.tab)
self.keyboard.release(Key.ctrl)

def closeTab(self):
```

```
self.keyboard.press(Key.ctrl)
self.keyboard.press('w')
self.keyboard.release('w')
self.keyboard.release(Key.ctrl)

def newTab(self):
self.keyboard.press(Key.ctrl)
self.keyboard.press('n')
self.keyboard.release('n')
self.keyboard.release(Key.ctrl)


class WindowOpt:
def __init__(self):
self.keyboard = Controller()

def openWindow(self):
self.maximizeWindow()

def closeWindow(self):
self.keyboard.press(Key.alt_l)
self.keyboard.press(Key.f4)
self.keyboard.release(Key.f4)
self.keyboard.release(Key.alt_l)

def minimizeWindow(self):
for i in range(2):
self.keyboard.press(Key.cmd)
self.keyboard.press(Key.down)
self.keyboard.release(Key.down)
self.keyboard.release(Key.cmd)
```

```
time.sleep(0.05)

def maximizeWindow(self):
self.keyboard.press(Key.cmd)
self.keyboard.press(Key.up)
self.keyboard.release(Key.up)
self.keyboard.release(Key.cmd)

def moveWindow(self, operation):
self.keyboard.press(Key.cmd)

if "left" in operation:
self.keyboard.press(Key.left)
self.keyboard.release(Key.left)
elif "right" in operation:
self.keyboard.press(Key.right)
self.keyboard.release(Key.right)
elif "down" in operation:
self.keyboard.press(Key.down)
self.keyboard.release(Key.down)
elif "up" in operation:
self.keyboard.press(Key.up)
self.keyboard.release(Key.up)
self.keyboard.release(Key.cmd)

def switchWindow(self):
self.keyboard.press(Key.alt_l)
self.keyboard.press(Key.tab)
self.keyboard.release(Key.tab)
self.keyboard.release(Key.alt_l)
def takeScreenShot(self):
```

```
from random import randint
im = ImageGrab.grab()
im.save(f'Files and Document/ss_{randint(1, 100)}.jpg')

def isContain(text, lst):
for word in lst:
if word in text:
return True
return False

def Win_Opt(operation):
w = WindowOpt()
if isContain(operation, ['open']):
w.openWindow()
elif isContain(operation, ['close']):
w.closeWindow()
elif isContain(operation, ['mini']):
w.minimizeWindow()
elif isContain(operation, ['maxi']):
w.maximizeWindow()
elif isContain(operation, ['move', 'slide']):
w.moveWindow(operation)
elif isContain(operation, ['switch','which']):
w.switchWindow()
elif isContain(operation, ['screenshot','capture','snapshot']):
w.takeScreenShot()
return

def Tab_Opt(operation):
t = TabOpt()
if isContain(operation, ['new','open','another','create']):
```

```
t.newTab()
elif isContain(operation, ['switch','move','another','next','previous','which']):
t.switchTab()
elif isContain(operation, ['close','delete']):
t.closeTab()
else:
return

def System_Opt(operation):
s = SystemTasks()
if 'delete' in operation:
s.delete()
elif 'save' in operation:
s.save(operation)
elif 'type' in operation:
s.write(operation)
elif 'select' in operation:
s.select()
elif 'enter' in operation:
s.hitEnter()
elif isContain(operation, ['notepad','paint','calc','word']):
s.openApp(operation)
elif isContain(operation, ['music','video']):
s.playMusic(operation)
else:
open_website(operation)
return

keyboard = Controller()
def mute():
for i in range(50):
```

```
keyboard.press(Key.media_volume_down)
keyboard.release(Key.media_volume_down)

def full():
for i in range(50):
keyboard.press(Key.media_volume_up)
keyboard.release(Key.media_volume_up)

def volumeControl(text):
if 'full' in text or 'max' in text: full()
elif 'mute' in text or 'min' in text: mute()
elif 'incre' in text:
for i in range(5):
keyboard.press(Key.media_volume_up)
keyboard.release(Key.media_volume_up)
elif 'decre' in text:
for i in range(5):
keyboard.press(Key.media_volume_down)
keyboard.release(Key.media_volume_down)

def systemInfo():
import wmi
c = wmi.WMI()
my_system_1 = c.Win32_LogicalDisk()[0]
my_system_2 = c.Win32_ComputerSystem()[0]
info = ["Total Disk Space: " + str(round(int(my_system_1.Size)/(1024**3),2)) + " GB",
"Free Disk Space: " + str(round(int(my_system_1.Freespace)/(1024**3),2)) + " GB",
"Manufacturer: " + my_system_2.Manufacturer,
"Model: " + my_system_2. Model,
"Owner: " + my_system_2.PrimaryOwnerName,
"Number of Processors: " + str(my_system_2.NumberOfProcessors),
```

```
"System Type: " + my_system_2.SystemType]
return info

def batteryInfo():
# usage = str(psutil.cpu_percent(interval=0.1))
battery = psutil.sensors_battery()
pr = str(battery.percent)
if battery.power_plugged:
return "Your System is currently on Charging Mode and it's " + pr + "% done."
return "Your System is currently on " + pr + "% battery life."

def OSHandler(query):
if isContain(query, ['system', 'info']):
return ['Here is your System Information...', '\n'.join(systemInfo())]
elif isContain(query, ['cpu', 'battery']):
return batteryInfo()

from difflib import get_close_matches
import json
from random import choice
import webbrowser

data = json.load(open('assets/websites.json', encoding='utf-8'))

def open_website(query):
query = query.replace('open','')
if query in data:
response = data[query]
else:
query = get_close_matches(query, data.keys(), n=2, cutoff=0.5)
if len(query)==0: return "None"
```

```
response = choice(data[query[0]])
webbrowser.open(response)
```

app_timer.py :

```
from time import sleep
import re
import playsound
from tkinter import *
from threading import Thread

def startTimer(query):
nums = re.findall(r'[0-9]+', query)
time = 0
if "minute" in query and "second" in query:
time = int(nums[0])*60 + int(nums[1])
elif "minute" in query:
time = int(nums[0])*60
elif "second" in query:
time = int(nums[0])
else: return

print("Timer Started")
sleep(time)
Thread(target=timer).start()
playsound.playsound("assets/audios/Timer.mp3")

def timer():
root = Tk()
root.title("Timer")
root.iconbitmap("assets/images/timer.ico")
w_width, w_height = 300, 150
```

```
s_width, s_height = root.winfo_screenwidth(), root.winfo_screenheight()
x, y = (s_width/2)-(w_width/2), (s_height/2)-(w_height/2)
root.geometry('%dx%d+%d+%d' % (w_width,w_height,x,y-30))
root['bg'] = 'white'

Label(root, text="Time's Up", font=("Arial Bold", 20), bg='white').pack(pady=20)
Button(root, text="  OK  ", font=("Arial", 15), relief=FLAT, bg='#14A769', fg='white', com-
mand=lambda:quit()).pack()

root.mainloop()
```

avatar_selection.py :

```
from tkinter import *
from PIL import Image, ImageTk
from tkinter import ttk
from pynput.keyboard import Key, Controller
import user_handler
from user_handler import UserData



u = UserData()
u.extractData()
avatarChoosen = u.getUserPhoto()

def closeWindow():
keyboard = Controller()
keyboard.press(Key.alt_l)
keyboard.press(Key.f4)
keyboard.release(Key.f4)
keyboard.release(Key.alt_l)
def SavePhoto():
```

```
user_handler.UpdateUserPhoto(avatarChoosen)
closeWindow()

def selectAVATAR(avt=0):
global avatarChoosen
avatarChoosen = avt

i=1
for avtr in
(avtb1,avtb2,avtb3,avtb4,avtb5,avtb6,avtb7,avtb8,avtb9,avtb10,avtb11,avtb12,avtb13,avtb14,avtb15):
if i==avt:
avtr['state'] = 'disabled'
else:
avtr['state'] = 'normal'
i+=1

if __name__ == "__main__":

background = '#F6FAFB'
avtrRoot = Tk()
avtrRoot.title("Choose Avatar")
avtrRoot.configure(bg=background)
w_width, w_height = 500, 450
s_width, s_height = avtrRoot.winfo_screenwidth(), avtrRoot.winfo_screenheight()
x, y = (s_width/2)-(w_width/2), (s_height/2)-(w_height/2)
avtrRoot.geometry('%dx%d+%d+%d' % (w_width,w_height,x,y-30))

Label(avtrRoot, text="Choose Your Avatar", font=('arial bold', 15), bg=background,
fg='#303E54').pack(pady=10)

avatarContainer = Frame(avtrRoot, bg=background)
```

```
avatarContainer.pack(pady=10, ipadx=50, ipady=20)
size = 100

#create a main frame
main_frame = Frame(avatarContainer)
main_frame.pack(fill=BOTH, expand=1)

#create a canvas
my_canvas = Canvas(main_frame, bg=background)
my_canvas.pack(side=LEFT, expand=1, fill=BOTH)

#add a scrollbar to the canvas
my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL, command=my_canvas.yview)
my_scrollbar.pack(side=RIGHT, fill=Y)

#configure the canvas
my_canvas.configure(yscrollcommand=my_scrollbar.set)
my_canvas.bind('<Configure>', lambda e: my_canvas.configure(scrollregion = my_canvas.bbox('all')))

#create another frame inside the canvas
second_frame = Frame(my_canvas)

#add that new frame to a window in the canvas
my_canvas.create_window((0,0), window=second_frame, anchor='nw')
avtr1 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a1.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr2 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a2.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr3 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a3.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr4 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a4.png').resize((size, size)), Im-
```

```
age.ANTIALIAS)
avtr5 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a5.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr6 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a6.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr7 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a7.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr8 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a8.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr9 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a9.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr10 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a10.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr11 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a11.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr12 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a12.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr13 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a13.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr14 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a14.png').resize((size, size)), Im-
age.ANTIALIAS)
avtr15 = ImageTk.PhotoImage(Image.open('assets/images/avatars/a15.png').resize((size, size)), Im-
age.ANTIALIAS)
avtb1 = Button(second_frame, image=avtr1, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(1))
avtb2 = Button(second_frame, image=avtr2, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(2))
avtb3 = Button(second_frame, image=avtr3, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(3))
avtb4 = Button(second_frame, image=avtr4, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(4))
```

```
avtb5 = Button(second_frame, image=avtr5, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(5))
avtb6 = Button(second_frame, image=avtr6, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(6))
avtb7 = Button(second_frame, image=avtr7, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(7))
avtb8 = Button(second_frame, image=avtr8, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(8))
avtb9 = Button(second_frame, image=avtr9, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(9))
avtb10 = Button(second_frame, image=avtr10, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(10))
avtb11 = Button(second_frame, image=avtr11, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(11))
avtb12 = Button(second_frame, image=avtr12, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(12))
avtb13 = Button(second_frame, image=avtr13, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(13))
avtb14 = Button(second_frame, image=avtr14, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(14))
avtb15 = Button(second_frame, image=avtr15, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(15)
avtb1.grid( row=0, column=0, ipadx=25, ipady=10)
avtb2.grid( row=0, column=1, ipadx=25, ipady=10)
avtb3.grid( row=0, column=2, ipadx=25, ipady=10)
avtb4.grid( row=1, column=0, ipadx=25, ipady=10)
avtb5.grid( row=1, column=1, ipadx=25, ipady=10)
avtb6.grid( row=1, column=2, ipadx=25, ipady=10)
avtb7.grid( row=2, column=0, ipadx=25, ipady=10)
avtb8.grid( row=2, column=1, ipadx=25, ipady=10)
avtb9.grid( row=2, column=2, ipadx=25, ipady=10)
```

```
avtb10.grid(row=3, column=0, ipadx=25, ipady=10)
avtb11.grid(row=3, column=1, ipadx=25, ipady=10)
avtb12.grid(row=3, column=2, ipadx=25, ipady=10)
avtb13.grid(row=4, column=0, ipadx=25, ipady=10)
avtb14.grid(row=4, column=1, ipadx=25, ipady=10)
avtb15.grid(row=4, column=2, ipadx=25, ipady=10)

BottomFrame = Frame(avtrRoot, bg=background)
BottomFrame.pack(pady=10)
Button(BottomFrame, text='        Update        ', font=('Montserrat Bold', 15), bg='#01933B', fg='white',
bd=0, relief=FLAT, command=SavePhoto).grid(row=0, column=0, padx=10)
Button(BottomFrame, text='        Cancel        ', font=('Montserrat Bold', 15), bg='#EDEDED',
fg='#3A3834', bd=0, relief=FLAT, command=closeWindow).grid(row=0, column=1, padx=10)

avtrRoot.iconbitmap("assets/images/changeProfile.ico")
avtrRoot.mainloop()
```

dictionary.py :

```
from difflib import get_close_matches
import json
from random import choice

data = json.load(open('assets/dict_data.json', encoding='utf-8'))

def getMeaning(word):
if word in data:
return word, data[word], 1
elif len(get_close_matches(word, data.keys())) > 0:
word = get_close_matches(word, data.keys())[0]
return word, data[word], 0
else:
```

```
return word, ["This word doesn't exists in the dictionary."],

def translate(query):
query = query.replace('dictionary', '')
if 'meaning' in query:
ind = query.index('meaning of')
word = query[ind+10:].strip().lower()
elif 'definition' in query:
try:
ind = query.index('definition of')
word = query[ind+13:].strip().lower()
except:
ind = query.index('definition')
word = query[ind+10:].strip().lower()
else: word = query

word, result, check = getMeaning(word)
result = choice(result)

if check==1:
return ["Here's the definition of \"" +word.capitalize()+ "", result]
elif check==0:
return ["I think you're looking for \"" +word.capitalize()+ "", "It's definition is,\n" + result]
else:
return [result, '']
```

face_unlocker.py :

```
import cv2
import os
from os.path import isfile, join
```

```
face_classifier = cv2.CascadeClassifier('Cascade/haarcascade_frontalface_default.xml')

def face_detector(img, size=0.5):
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(gray, 1.3, 5)

if faces is ():
return img,[]

for (x,y,w,h) in faces:
cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,255), 2)
#region of interest
roi = img[y:y+h, x:x+w]
roi = cv2.resize(roi, (200,200))

return img,roi

def startDetecting():
try:
model = cv2.face.LBPHFaceRecognizer_create()
model.read('userData/trainer.yml')
except:
print('Please Add your face')
return None

flag = False
cap = cv2.VideoCapture(0)

while True:
ret, frame = cap.read()
image, face = face_detector(frame)
```

```
try:
face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
result = model.predict(face)

if result[1] < 500:
confindence = int((1-(result[1])/300) * 100) #percentange
display_string = str(confindence) + '%'
cv2.putText(image, display_string, (100, 120), cv2.FONT_HERSHEY_COMPLEX, 1, (250, 120, 255),
2)

if confindence > 80:
cv2.putText(image, "Unlocked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
cv2.imshow('Face Cropper', image)
flag = True
break
else:
cv2.putText(image, "Locked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 255), 2)
cv2.imshow('Face Cropper', image)

except Exception as e:
cv2.putText(image, "Face Not Found", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0),
2)
cv2.imshow('Face Cropper', image)
pass

if cv2.waitKey(1) == ord('q'):
break

cap.release()
cv2.destroyAllWindows()
return flag
```

```
imageName = ''
def clickPhoto():
global imageName
if os.path.exists('Camera')==False:
os.mkdir('Camera')

from time import sleep
import playsound
from datetime import datetime

cam = cv2.VideoCapture(0)
_, frame = cam.read()
playsound.playsound('assets/audios/photoclick.mp3')
imageName = 'Camera/Camera_'+str(datetime.now())[:19].replace(':', '_')+'.png'
cv2.imwrite(imageName, frame)
cam.release()
cv2.destroyAllWindows()

def viewPhoto():
from PIL import Image
img = Image.open(imageName)
img.show()
```

File_handler.py :

```
import subprocess
import wmi
import os
import sys
import webbrowser

if os.path.exists('Files and Document') == False:
```

```
os.mkdir('Files and Document')
path = 'Files and Document/'

def isContain(text, list):
for word in list:
if word in text:
return True
return False

def createFile(text):
appLocation = "C:\\Users\\Roshan\\AppData\\Local\\Programs\\Microsoft VS
Code\\Code.exe"#"C:\\Program Files\\Sublime Text 3\\sublime_text.exe"

if isContain(text, ["ppt","power point","powerpoint"]):
file_name = "sample_file.ppt"
appLocation = "C:\\Program Files (x86)\\Microsoft Office\\Office15\\POWERPNT.exe"

elif isContain(text, ['excel','spreadsheet']):
file_name = "sample_file.xsl"
appLocation = "C:\\Program Files (x86)\\Microsoft Office\\Office15\\EXCEL.EXE"

elif isContain(text, ['word','document']):
file_name = "sample_file.docx"
appLocation = "C:\\Program Files (x86)\\Microsoft Office\\Office15\\WINWORD.EXE"

elif isContain(text, ["text","simple","normal"]): file_name = "sample_file.txt"
elif "python" in text: file_name = "sample_file.py"
elif "css" in text:                    file_name = "sample_file.css"
elif "javascript" in text: file_name = "sample_file.js"
elif "html" in text: file_name = "sample_file.html"
elif "c plus plus" in text or "c + +" in text: file_name = "sample_file.cpp"
```

```
elif "java" in text: file_name = "sample_file.java"
elif "json" in text: file_name = "sample_file.json"
else: return "Unable to create this type of file"

file = open(path + file_name, 'w')
file.close()
subprocess.Popen([appLocation, path + file_name])
return "File is created.\nNow you can edit this file"

def CreateHTMLProject(project_name='Sample'):

if os.path.isdir(path + project_name):
webbrowser.open(os.getcwd() + '/' + path + project_name + "\\index.html")
return 'There is a same project which is already created, look at this...'
else:
os.mkdir(path + project_name)

os.mkdir(path+project_name+ '/images')
os.mkdir(path+project_name+ '/videos')

htmlContent = '<html>\n\t<head>\n\t\t<title> ' + project_name + ' </title>\n\t\t<link rel="stylesheet"
type="text/css" href="style.css">\n\t</head>\n<body>\n\t<p id="label"></p>\n\t<button id="btn" on-
click="showText()"> Click Me </button>\n\t<script src="script.js"></script>\n</body>\n</html>'

htmlFile = open(path+project_name+ '/index.html', 'w')
htmlFile.write(htmlContent)
htmlFile.close()

cssContent = '* {\n\tmargin:0;\n\tpadding:0;\n}\nbody {\n\theight:100vh;\n\tdisplay:flex;\n\tjustify-
content:center;\n\talign-items:center;\n}\n#btn {\n\twidth:200px;\n\tpadding: 20px 10px;\n\tborder-
radius:5px;\n\tbackground-color:red;\n\tcolor:#fff;\n\toutline:none;border:none;\n}\np {\n\tfont-
```

```
size:30px;\n}'

cssFile = open(path+project_name+ '/style.css', 'w')
cssFile.write(cssContent)
cssFile.close

jsContent = 'function showText() {\n\tdocument.getElementById("label").innerHTML="Successfully
Created '+ project_name +' Project";\n\tdocument.getElementById("btn").style="background-
color:green;"\n}'

jsFile = open(path+project_name+ '/script.js', 'w')
jsFile.write(jsContent)
jsFile.close()

appLocation = "C:\\Program Files\\Sublime Text 3\\sublime_text.exe"
# subprocess.Popen([appLocation, path + project_name])
subprocess.Popen([appLocation, path + project_name + "/index.html"])
subprocess.Popen([appLocation, path + project_name + "/style.css"])
subprocess.Popen([appLocation, path + project_name + "/script.js"])

webbrowser.open(os.getcwd() + '/' + path + project_name + "\\index.html")


return f'Successfully Created {project_name} Project'
```

gui_assistant.py :

```
#########################
# GLOBAL VARIABLES USED #
#########################
ai_name = 'F.R.I.D.Y.'.lower()
EXIT_COMMANDS = ['bye','exit','quit','shut down', 'shutdown']

rec_email, rec_phoneno = "", ""
WAEMEntry = None

avatarChoosen = 0
choosedAvtrImage = None

botChatTextBg = "#007cc7"
botChatText = "white"
userChatTextBg = "#4da8da"

chatBgColor = '#12232e'
background = '#203647'
textColor = 'white'
AITaskStatusLblBG = '#203647'
KCS_IMG = 1 #0 for light, 1 for dark
voice_id = 0 #0 for female, 1 for male
ass_volume = 1 #max volume
ass_voiceRate = 200 #normal voice rate

######################################## IMPORTING MODULES
##########################################
""" User Created Modules """
try:
```

```
import normal_chat
import math_function
import app_control
import web_scrapping
import game
import app_timer
import dictionary
import todo_handler
import file_handler
from user_handler import UserData
from face_unlocker import clickPhoto, viewPhoto
from dotenv import load_dotenv

load_dotenv()
except Exception as e:
raise e

""" System Modules """
try:
import os
import speech_recognition as sr
import pyttsx3
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from tkinter import colorchooser
from PIL import Image, ImageTk
from time import sleep
from threading import Thread
except Exception as e:
print(e)
```

```
############################################ LOGIN CHECK
##############################################
try:
user = UserData()
user.extractData()
ownerName = user.getName().split()[0]
ownerDesignation = "Sir"
if user.getGender()=="Female": ownerDesignation = "Ma'am"
ownerPhoto = user.getUserPhoto()
except Exception as e:
print("You're not Registered Yet !\nRun SECURITY.py file to register your face.")
raise SystemExit
########################################### BOOT UP WINDOW
###########################################
def ChangeSettings(write=False):
import pickle
global background, textColor, chatBgColor, voice_id, ass_volume, ass_voiceRate, AITaskStatusLblBG,
KCS_IMG, botChatTextBg, botChatText, userChatTextBg
setting = {'background': background,
'textColor': textColor,
'chatBgColor': chatBgColor,
'AITaskStatusLblBG': AITaskStatusLblBG,
'KCS_IMG': KCS_IMG,
'botChatText': botChatText,
'botChatTextBg': botChatTextBg,
'userChatTextBg': userChatTextBg,
'voice_id': voice_id,
'ass_volume': ass_volume,
'ass_voiceRate': ass_voiceRate
}
if write:
```

```
with open('userData/settings.pck', 'wb') as file:
pickle.dump(setting, file)
return
try:
with open('userData/settings.pck', 'rb') as file:
loadSettings = pickle.load(file)
background = loadSettings['background']
textColor = loadSettings['textColor']
chatBgColor = loadSettings['chatBgColor']
AITaskStatusLblBG = loadSettings['AITaskStatusLblBG']
KCS_IMG = loadSettings['KCS_IMG']
botChatText = loadSettings['botChatText']
botChatTextBg = loadSettings['botChatTextBg']
userChatTextBg = loadSettings['userChatTextBg']
voice_id = loadSettings['voice_id']
ass_volume = loadSettings['ass_volume']
ass_voiceRate = loadSettings['ass_voiceRate']
except Exception as e:
pass

if os.path.exists('userData/settings.pck')==False:
ChangeSettings(True)

def changeTheme():
global background, textColor, AITaskStatusLblBG, KCS_IMG, botChatText, botChatTextBg, us-
erChatTextBg, chatBgColor
if themeValue.get()==1:
background, textColor, AITaskStatusLblBG, KCS_IMG = "#203647", "white", "#203647",1
cbl['image'] = cblDarkImg
kbBtn['image'] = kbphDark
settingBtn['image'] = sphDark
```

```
AITaskStatusLbl['bg'] = AITaskStatusLblBG
botChatText, botChatTextBg, userChatTextBg = "white", "#007cc7", "#4da8da"
chatBgColor = "#12232e"
colorbar['bg'] = chatBgColor
else:
background, textColor, AITaskStatusLblBG, KCS_IMG = "#F6FAFB", "#303E54", "#14A769", 0
cbl['image'] = cblLightImg
kbBtn['image'] = kbphLight
settingBtn['image'] = sphLight
AITaskStatusLbl['bg'] = AITaskStatusLblBG
botChatText, botChatTextBg, userChatTextBg = "#494949", "#EAEAEA", "#23AE79"
chatBgColor = "#F6FAFB"
colorbar['bg'] = '#E8EBEF'

root['bg'], root2['bg'] = background, background
settingsFrame['bg'] = background
settingsLbl['fg'], userPhoto['fg'], userName['fg'], assLbl['fg'], voiceRateLbl['fg'], volumeLbl['fg'],
themeLbl['fg'], chooseChatLbl['fg'] = textColor, textColor, textColor, textColor, textColor, textColor,
textColor, textColor
settingsLbl['bg'], userPhoto['bg'], userName['bg'], assLbl['bg'], voiceRateLbl['bg'], volumeLbl['bg'],
themeLbl['bg'], chooseChatLbl['bg'] = background, background, background, background, background,
background, background, background
s.configure('Wild.TRadiobutton', background=background, foreground=textColor)
volumeBar['bg'], volumeBar['fg'], volumeBar['highlightbackground'] = background, textColor, back-
ground
chat_frame['bg'], root1['bg'] = chatBgColor, chatBgColor
userPhoto['activebackground'] = background
ChangeSettings(True)

def changeVoice(e):
global voice_id
```

```
voice_id=0
if assVoiceOption.get()=='Male': voice_id=1
engine.setProperty('voice', voices[voice_id].id)
ChangeSettings(True)

def changeVolume(e):
global ass_volume
ass_volume = volumeBar.get() / 100
engine.setProperty('volume', ass_volume)
ChangeSettings(True)

def changeVoiceRate(e):
global ass_voiceRate
temp = voiceOption.get()
if temp=='Very Low': ass_voiceRate = 100
elif temp=='Low': ass_voiceRate = 150
elif temp=='Fast': ass_voiceRate = 250
elif temp=='Very Fast': ass_voiceRate = 300
else: ass_voiceRate = 200
print(ass_voiceRate)
engine.setProperty('rate', ass_voiceRate)
ChangeSettings(True)

ChangeSettings()

############################################## SET UP VOICE
##############################################
try:
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[voice_id].id) #male
```

```
engine.setProperty('volume', ass_volume)
except Exception as e:
print(e)
######################################## SET UP TEXT TO SPEECH
########################################
def speak(text, display=False, icon=False):
AITaskStatusLbl['text'] = 'Speaking...'
if icon: Label(chat_frame, image=botIcon, bg=chatBgColor).pack(anchor='w',pady=0)
if display: attachTOframe(text, True)
print('\n'+ai_name.upper()+': '+text)
try:
engine.say(text)
engine.runAndWait()
except:
print("Try not to type more...")

######################################## SET UP SPEECH TO TEXT
########################################
def record(clearChat=True, iconDisplay=True):
print('\nListening...')
AITaskStatusLbl['text'] = 'Listening...'
r = sr.Recognizer()
r.dynamic_energy_threshold = False
r.energy_threshold = 4000
with sr.Microphone() as source:
r.adjust_for_ambient_noise(source)
audio = r.listen(source)
said = ""
try:
AITaskStatusLbl['text'] = 'Processing...'
said = r.recognize_google(audio)
```

```
print(f"\nUser said: {said}")
if clearChat:
clearChatScreen()
if iconDisplay: Label(chat_frame, image=userIcon, bg=chatBgColor).pack(anchor='e',pady=0)
attachTOframe(said)
except Exception as e:
print(e)
# speak("I didn't get it, Say that again please...")
if "connection failed" in str(e):
speak("Your System is Offline...", True, True)
return 'None'
return said.lower()

def voiceMedium():
while True:
query = record()
if query == 'None': continue
if isContain(query, EXIT_COMMANDS):
speak("Shutting down the System. Good Bye "+ownerDesignation+"!", True, True)
break
else: main(query.lower())
app_control.Win_Opt('close')

def keyboardInput(e):
user_input = UserField.get().lower()
if user_input!="":
clearChatScreen()
if isContain(user_input, EXIT_COMMANDS):
speak("Shutting down the System. Good Bye "+ownerDesignation+"!", True, True)
else:
Label(chat_frame, image=userIcon, bg=chatBgColor).pack(anchor='e',pady=0)
```

```
attachTOframe(user_input.capitalize())
Thread(target=main, args=(user_input,)).start()
UserField.delete(0, END)

######################################## TASK/COMMAND HANDLER
########################################
def isContain(txt, lst):
for word in lst:
if word in txt:
return True
return False

def main(text):

if "project" in text:
if isContain(text, ['make', 'create']):
speak("What do you want to give the project name ?", True, True)
projectName = record(False, False)
speak(file_handler.CreateHTMLProject(projectName.capitalize()), True)
return

if "create" in text and "file" in text:
speak(file_handler.createFile(text), True, True)
return

if "translate" in text:
speak("What do you want to translate?", True, True)
sentence = record(False, False)
speak("Which langauage to translate ?", True)
langauage = record(False, False)
result = normal_chat.lang_translate(sentence, langauage)
```

```
if result=="None": speak("This langauage doesn't exists")
else:
speak(f"In {langauage.capitalize()} you would say:", True)
if langauage=="hindi":
attachTOframe(result.text, True)
speak(result.pronunciation)
else: speak(result.text, True)
return

if 'list' in text:
if isContain(text, ['add', 'create', 'make']):
speak("What do you want to add?", True, True)
item = record(False, False)
todo_handler.toDoList(item)
speak("Alright, I added to your list", True)
return
if isContain(text, ['show', 'my list']):
items = todo_handler.showtoDoList()
if len(items)==1:
speak(items[0], True, True)
return
attachTOframe('\n'.join(items), True)
speak(items[0])
return

if isContain(text, ['battery', 'system info']):
result = app_control.OSHandler(text)
if len(result)==2:
speak(result[0], True, True)
attachTOframe(result[1], True)
else:
```

```
speak(result, True, True)
return

if isContain(text, ['meaning', 'dictionary', 'definition', 'define']):
result = dictionary.translate(text)
speak(result[0], True, True)
if result[1]==": return
speak(result[1], True)
return

if 'selfie' in text or ('click' in text and 'photo' in text):
speak("Sure "+ownerDesignation+"...", True, True)
clickPhoto()
speak('Do you want to view your clicked photo?', True)
query = record(False)
if isContain(query, ['yes', 'sure', 'yeah', 'show me']):
Thread(target=viewPhoto).start()
speak("Ok, here you go...", True, True)
else:
speak("No Problem "+ownerDesignation, True, True)
return

if 'volume' in text:
app_control.volumeControl(text)
Label(chat_frame, image=botIcon, bg=chatBgColor).pack(anchor='w',pady=0)
attachTOframe('Volume Settings Changed', True)
return

if isContain(text, ['timer', 'countdown']):
Thread(target=app_timer.startTimer, args=(text,)).start()
speak('Ok, Timer Started!', True, True)
```

```
return

if 'whatsapp' in text:
speak("Sure "+ownerDesignation+"...", True, True)
speak('Whom do you want to send the message?', True)
WAEMPOPUP("WhatsApp", "Phone Number")
attachTOframe(rec_phoneno)
speak('What is the message?', True)
message = record(False, False)
Thread(target=web_scrapping.sendWhatsapp, args=(rec_phoneno, message,)).start()
speak("Message is on the way. Do not move away from the screen.")
attachTOframe("Message Sent", True)
return

if 'email' in text:
speak('Whom do you want to send the email?', True, True)
WAEMPOPUP("Email", "E-mail Address")
attachTOframe(rec_email)
speak('What is the Subject?', True)
subject = record(False, False)
speak('What message you want to send ?', True)
message = record(False, False)
Thread(target=web_scrapping.email, args=(rec_email,message,subject,) ).start()
speak('Email has been Sent', True)
return

if isContain(text, ['covid','virus']):
result = web_scrapping.covid(text)
if 'str' in str(type(result)):
speak(result, True, True)
return
```

```
speak(result[0], True, True)
result = '\n'.join(result[1])
attachTOframe(result, True)
return

if isContain(text, ['youtube','video']):
speak("Ok "+ownerDesignation+", here a video for you...", True, True)
try:
speak(web_scrapping.youtube(text), True)
except Exception as e:
print(e)
speak("Desired Result Not Found", True)
return

if isContain(text, ['search', 'image']):
if 'image' in text and 'show' in text:
Thread(target=showImages, args=(text,)).start()
speak('Here are the images...', True, True)
return
speak(web_scrapping.googleSearch(text), True, True)
return

if isContain(text, ['map', 'direction']):
if "direction" in text:
speak('What is your starting location?', True, True)
startingPoint = record(False, False)
speak("Ok "+ownerDesignation+", Where you want to go?", True)
destinationPoint = record(False, False)
speak("Ok "+ownerDesignation+", Getting Directions...", True)
try:
distance = web_scrapping.giveDirections(startingPoint, destinationPoint)
```

```
speak('You have to cover a distance of '+ distance, True)
except:
speak("I think location is not proper, Try Again!")
else:
web_scrapping.maps(text)
speak('Here you go...', True, True)
return

if isContain(text, ['factorial','log','value of','math',' + ',' - ',' x ','multiply','divided
by','binary','hexadecimal','octal','shift','sin ','cos ','tan ']):
try:
speak(('Result is: ' + math_function.perform(text)), True, True)
except Exception as e:
return
return

if "joke" in text:
speak('Here is a joke...', True, True)
speak(web_scrapping.jokes(), True)
return

if isContain(text, ['news']):
speak('Getting the latest news...', True, True)
headlines,headlineLinks = web_scrapping.latestNews(2)
for head in headlines: speak(head, True)
speak('Do you want to read the full news?', True)
text = record(False, False)
if isContain(text, ["no","don't"]):
speak("No Problem "+ownerDesignation, True)
else:
speak("Ok "+ownerDesignation+", Opening browser...", True)
```

```
web_scrapping.openWebsite('https://indianexpress.com/latest-news/')
speak("You can now read the full news from this website.")
return

if isContain(text, ['weather']):
data = web_scrapping.weather()
speak(", False, True)
showSingleImage("weather", data[:-1])
speak(data[-1])
return

if isContain(text, ['screenshot']):
Thread(target=app_control.Win_Opt, args=('screenshot',)).start()
speak("Screen Shot Taken", True, True)
return

if isContain(text, ['window','close that']):
app_control.Win_Opt(text)
return

if isContain(text, ['tab']):
app_control.Tab_Opt(text)
return

if isContain(text, ['setting']):
raise_frame(root2)
clearChatScreen()
return

if isContain(text, ['open','type','save','delete','select','press enter']):
app_control.System_Opt(text)
```

```
return

if isContain(text, ['wiki', 'who is']):
Thread(target=web_scrapping.downloadImage, args=(text, 1,)).start()
speak('Searching...', True, True)
result = web_scrapping.wikiResult(text)
showSingleImage('wiki')
speak(result, True)
return

if isContain(text, ['game']):
speak("Which game do you want to play?", True, True)
attachTOframe(game.showGames(), True)
text = record(False)
if text=="None":
speak("Didn't understand what you say?", True, True)
return
if 'online' in text:
speak("Ok "+ownerDesignation+", Let's play some online games", True, True)
web_scrapping.openWebsite('https://www.agame.com/games/mini-games/')
return
if isContain(text, ["don't", "no", "cancel", "back", "never"]):
speak("No Problem "+ownerDesignation+", We'll play next time.", True, True)
else:
speak("Ok "+ownerDesignation+", Let's Play " + text, True, True)
os.system(f"python -c \"from modules import game; game.play('{text}')\"")
return

if isContain(text, ['coin','dice','die']):
if "toss" in text or "roll" in text or "flip" in text:
speak("Ok "+ownerDesignation, True, True)
```

```
result = game.play(text)
if "Head" in result: showSingleImage('head')
elif "Tail" in result: showSingleImage('tail')
else: showSingleImage(result[-1])
speak(result)
return

if isContain(text, ['time','date']):
speak(normal_chat.chat(text), True, True)
return

if 'my name' in text:
speak('Your name is, ' + ownerName, True, True)
return

if isContain(text, ['voice']):
global voice_id
try:
if 'female' in text: voice_id = 0
elif 'male' in text: voice_id = 1
else:
if voice_id==0: voice_id=1
else: voice_id=0
engine.setProperty('voice', voices[voice_id].id)
ChangeSettings(True)
speak("Hello "+ownerDesignation+", I have changed my voice. How may I help you?", True, True)
assVoiceOption.current(voice_id)
except Exception as e:
print(e)
return
```

```
if isContain(text, ['morning','evening','noon']) and 'good' in text:
speak(normal_chat.chat("good"), True, True)
return

result = normal_chat.reply(text)
if result != "None": speak(result, True, True)
else:
speak("I don't know anything about this. Do you want to search it on web?", True, True)
response = record(False, True)
if isContain(response, ["no","don't"]):
speak("Ok "+ownerDesignation, True)
else:
speak("Here's what I found on the web... ", True, True)
web_scrapping.googleSearch(text)


##################################### DELETE USER ACCOUNT
########################################
def deleteUserData():
result = messagebox.askquestion('Alert', 'Are you sure you want to delete your Face Data ?')
if result=='no': return
messagebox.showinfo('Clear Face Data', 'Your face has been cleared\nRegister your face again to use.')
import shutil
shutil.rmtree('userData')
root.destroy()

####################
####### GUI #########
####################

############ ATTACHING BOT/USER CHAT ON CHAT SCREEN ###########
```

```
def attachTOframe(text,bot=False):
if bot:
botchat = Label(chat_frame,text=text, bg=botChatTextBg, fg=botChatText, justify=LEFT,
wraplength=250, font=('Montserrat',12, 'bold'))
botchat.pack(anchor='w',ipadx=5,ipady=5,pady=5)
else:
userchat = Label(chat_frame, text=text, bg=userChatTextBg, fg='white', justify=RIGHT,
wraplength=250, font=('Montserrat',12, 'bold'))
userchat.pack(anchor='e',ipadx=2,ipady=2,pady=5)

def clearChatScreen():
for wid in chat_frame.winfo_children():
wid.destroy()

### SWITCHING BETWEEN FRAMES ###
def raise_frame(frame):
frame.tkraise()
clearChatScreen()

################## SHOWING DOWNLOADED IMAGES ###############
img0, img1, img2, img3, img4 = None, None, None, None, None
def showSingleImage(type, data=None):
global img0, img1, img2, img3, img4
try:
img0 = ImageTk.PhotoImage(Image.open('Downloads/0.jpg').resize((90,110), Image.ANTIALIAS))
except:
pass
img1 = ImageTk.PhotoImage(Image.open('assets/images/heads.jpg').resize((220,200), Im-
age.ANTIALIAS))
img2 = ImageTk.PhotoImage(Image.open('assets/images/tails.jpg').resize((220,200), Im-
age.ANTIALIAS))
```

```
img4 = ImageTk.PhotoImage(Image.open('assets/images/WeatherImage.png'))

if type=="weather":
weather = Frame(chat_frame)
weather.pack(anchor='w')
Label(weather, image=img4, bg=chatBgColor).pack()
Label(weather, text=data[0], font=('Arial Bold', 45), fg='white', bg='#3F48CC').place(x=65,y=45)
Label(weather, text=data[1], font=('Montserrat', 15), fg='white', bg='#3F48CC').place(x=78,y=110)
Label(weather, text=data[2], font=('Montserrat', 10), fg='white', bg='#3F48CC').place(x=78,y=140)
Label(weather, text=data[3], font=('Arial Bold', 12), fg='white', bg='#3F48CC').place(x=60,y=160)

elif type=="wiki":
Label(chat_frame, image=img0, bg='#EAEAEA').pack(anchor='w')
elif type=="head":
Label(chat_frame, image=img1, bg='#EAEAEA').pack(anchor='w')
elif type=="tail":
Label(chat_frame, image=img2, bg='#EAEAEA').pack(anchor='w')
else:
img3 = ImageTk.PhotoImage(Image.open('assets/images/dice/'+type+'.jpg').resize((200,200), Im-
age.ANTIALIAS))
Label(chat_frame, image=img3, bg='#EAEAEA').pack(anchor='w')

def showImages(query):
global img0, img1, img2, img3
web_scrapping.downloadImage(query)
w, h = 150, 110
#Showing Images
imageContainer = Frame(chat_frame, bg='#EAEAEA')
imageContainer.pack(anchor='w')
#loading images
img0 = ImageTk.PhotoImage(Image.open('Downloads/0.jpg').resize((w,h), Image.ANTIALIAS))
```

```
img1 = ImageTk.PhotoImage(Image.open('Downloads/1.jpg').resize((w,h), Image.ANTIALIAS))
img2 = ImageTk.PhotoImage(Image.open('Downloads/2.jpg').resize((w,h), Image.ANTIALIAS))
img3 = ImageTk.PhotoImage(Image.open('Downloads/3.jpg').resize((w,h), Image.ANTIALIAS))
#Displaying
Label(imageContainer, image=img0, bg='#EAEAEA').grid(row=0, column=0)
Label(imageContainer, image=img1, bg='#EAEAEA').grid(row=0, column=1)
Label(imageContainer, image=img2, bg='#EAEAEA').grid(row=1, column=0)
Label(imageContainer, image=img3, bg='#EAEAEA').grid(row=1, column=1)



############################## WAEM - WhatsApp Email
##################################
def sendWAEM():
global rec_phoneno, rec_email
data = WAEMEntry.get()
rec_email, rec_phoneno = data, data
WAEMEntry.delete(0, END)
app_control.Win_Opt('close')
def send(e):
sendWAEM()

def WAEMPOPUP(Service='None', rec='Reciever'):
global WAEMEntry
PopUProot = Tk()
PopUProot.title(f'{Service} Service')
PopUProot.configure(bg='white')

if Service=="WhatsApp": PopUProot.iconbitmap("assets/images/whatsapp.ico")
else: PopUProot.iconbitmap("assets/images/email.ico")
w_width, w_height = 410, 200
s_width, s_height = PopUProot.winfo_screenwidth(), PopUProot.winfo_screenheight()
```

```
x, y = (s_width/2)-(w_width/2), (s_height/2)-(w_height/2)
PopUProot.geometry('%dx%d+%d+%d' % (w_width,w_height,x,y-30)) #center location of the screen
Label(PopUProot, text=f'Reciever {rec}', font=('Arial', 16), bg='white').pack(pady=(20, 10))
WAEMEntry = Entry(PopUProot, bd=10, relief=FLAT, font=('Arial', 12), justify='center',
bg='#DCDCDC', width=30)
WAEMEntry.pack()
WAEMEntry.focus()

SendBtn = Button(PopUProot, text='Send', font=('Arial', 12), relief=FLAT, bg='#14A769', fg='white',
command=sendWAEM)
SendBtn.pack(pady=20, ipadx=10)
PopUProot.bind('<Return>', send)
PopUProot.mainloop()

######################### CHANGING CHAT BACKGROUND COLOR
#########################
def getChatColor():
global chatBgColor
myColor = colorchooser.askcolor()
if myColor[1] is None: return
chatBgColor = myColor[1]
colorbar['bg'] = chatBgColor
chat_frame['bg'] = chatBgColor
root1['bg'] = chatBgColor
ChangeSettings(True)

chatMode = 1
def changeChatMode():
global chatMode
if chatMode==1:
# appControl.volumeControl('mute')
```

```
VoiceModeFrame.pack_forget()
TextModeFrame.pack(fill=BOTH)
UserField.focus()
chatMode=0
else:
# appControl.volumeControl('full')
TextModeFrame.pack_forget()
VoiceModeFrame.pack(fill=BOTH)
root.focus()
chatMode=1

################################################ GUI
################################################

def onhover(e):
userPhoto['image'] = chngPh
def onleave(e):
userPhoto['image'] = userProfileImg

def UpdateIMAGE():
global ownerPhoto, userProfileImg, userIcon

os.system('python modules/avatar_selection.py')
u = UserData()
u.extractData()
ownerPhoto = u.getUserPhoto()
userProfileImg = Im-
ageTk.PhotoImage(Image.open("assets/images/avatars/a"+str(ownerPhoto)+".png").resize((120, 120)))

userPhoto['image'] = userProfileImg
userIcon = PhotoImage(file="assets/images/avatars/ChatIcons/a"+str(ownerPhoto)+".png")
```

```
def SelectAvatar():
Thread(target=UpdateIMAGE).start()


#####################################  MAIN GUI
#########################################################

#### SPLASH/LOADING SCREEN ####
def progressbar():
s = ttk.Style()
s.theme_use('clam')
s.configure("white.Horizontal.TProgressbar", foreground='white', background='white')
progress_bar = ttk.Progressbar(splash_root,style="white.Horizontal.TProgressbar", ori-
ent="horizontal",mode="determinate", length=303)
progress_bar.pack()
splash_root.update()
progress_bar['value'] = 0
splash_root.update()

while progress_bar['value'] < 100:
progress_bar['value'] += 5
# splash_percentage_label['text'] = str(progress_bar['value']) + ' %'
splash_root.update()
sleep(0.1)

def destroySplash():
splash_root.destroy()

if __name__ == '__main__':
splash_root = Tk()
```

```
splash_root.configure(bg='#3895d3')
splash_root.overrideredirect(True)
splash_label = Label(splash_root, text="Processing...", font=('montserrat',15),bg='#3895d3',fg='white')
splash_label.pack(pady=40)
# splash_percentage_label = Label(splash_root, text="0 %",
font=('montserrat',15),bg='#3895d3',fg='white')
# splash_percentage_label.pack(pady=(0,10))

w_width, w_height = 400, 200
s_width, s_height = splash_root.winfo_screenwidth(), splash_root.winfo_screenheight()
x, y = (s_width/2)-(w_width/2), (s_height/2)-(w_height/2)
splash_root.geometry('%dx%d+%d+%d' % (w_width,w_height,x,y-30))

progressbar()
splash_root.after(10, destroySplash)
splash_root.mainloop()

root = Tk()
root.title('F.R.I.D.A.Y')
w_width, w_height = 400, 650
s_width, s_height = root.winfo_screenwidth(), root.winfo_screenheight()
x, y = (s_width/2)-(w_width/2), (s_height/2)-(w_height/2)
root.geometry('%dx%d+%d+%d' % (w_width,w_height,x,y-30)) #center location of the screen
root.configure(bg=background)
# root.resizable(width=False, height=False)
root.pack_propagate(0)

root1 = Frame(root, bg=chatBgColor)
root2 = Frame(root, bg=background)
root3 = Frame(root, bg=background)
```

```
for f in (root1, root2, root3):
f.grid(row=0, column=0, sticky='news')

################################
########  CHAT SCREEN  #########
################################

#Chat Frame
chat_frame = Frame(root1, width=380,height=551,bg=chatBgColor)
chat_frame.pack(padx=10)
chat_frame.pack_propagate(0)

bottomFrame1 = Frame(root1, bg='#dfdfdf', height=100)
bottomFrame1.pack(fill=X, side=BOTTOM)
VoiceModeFrame = Frame(bottomFrame1, bg='#dfdfdf')
VoiceModeFrame.pack(fill=BOTH)
TextModeFrame = Frame(bottomFrame1, bg='#dfdfdf')
TextModeFrame.pack(fill=BOTH)

# VoiceModeFrame.pack_forget()
TextModeFrame.pack_forget()

cblLightImg = PhotoImage(file='assets/images/centralButton.png')
cblDarkImg = PhotoImage(file='assets/images/centralButton1.png')
if KCS_IMG==1: cblimage=cblDarkImg
else: cblimage=cblLightImg
cbl = Label(VoiceModeFrame, fg='white', image=cblimage, bg='#dfdfdf')
cbl.pack(pady=17)
AITaskStatusLbl = Label(VoiceModeFrame, text='    Offline', fg='white', bg=AITaskStatusLblBG,
font=('montserrat', 16))
AITaskStatusLbl.place(x=140,y=32)
```

```
#Settings Button
sphLight = PhotoImage(file = "assets/images/setting.png")
sphLight = sphLight.subsample(2,2)
sphDark = PhotoImage(file = "assets/images/setting1.png")
sphDark = sphDark.subsample(2,2)
if KCS_IMG==1: sphimage=sphDark
else: sphimage=sphLight
settingBtn = Button(VoiceModeFrame,image=sphimage,height=30,width=30,
bg='#dfdfdf',borderwidth=0,activebackground="#dfdfdf",command=lambda: raise_frame(root2))
settingBtn.place(relx=1.0, y=30,x=-20, anchor="ne")

#Keyboard Button
kbphLight = PhotoImage(file = "assets/images/keyboard.png")
kbphLight = kbphLight.subsample(2,2)
kbphDark = PhotoImage(file = "assets/images/keyboard1.png")
kbphDark = kbphDark.subsample(2,2)
if KCS_IMG==1: kbphimage=kbphDark
else: kbphimage=kbphLight
kbBtn = Button(VoiceModeFrame,image=kbphimage,height=30,width=30,
bg='#dfdfdf',borderwidth=0,activebackground="#dfdfdf", command=changeChatMode)
kbBtn.place(x=25, y=30)

#Mic
micImg = PhotoImage(file = "assets/images/mic.png")
micImg = micImg.subsample(2,2)
micBtn = Button(TextModeFrame,image=micImg,height=30,width=30,
bg='#dfdfdf',borderwidth=0,activebackground="#dfdfdf", command=changeChatMode)
micBtn.place(relx=1.0, y=30,x=-20, anchor="ne")

#Text Field
```

```
TextFieldImg = PhotoImage(file='assets/images/textField.png')
UserFieldLBL = Label(TextModeFrame, fg='white', image=TextFieldImg, bg='#dfdfdf')
UserFieldLBL.pack(pady=17, side=LEFT, padx=10)
UserField = Entry(TextModeFrame, fg='white', bg='#203647', font=('Montserrat', 16), bd=6, width=22,
relief=FLAT)
UserField.place(x=20, y=30)
UserField.insert(0, "Ask me anything...")
UserField.bind('<Return>', keyboardInput)

#User and Bot Icon
userIcon = PhotoImage(file="assets/images/avatars/ChatIcons/a"+str(ownerPhoto)+".png")
botIcon = PhotoImage(file="assets/images/assistant2.png")
botIcon = botIcon.subsample(2,2)



############################
########  SETTINGS  #######
############################

settingsLbl = Label(root2, text='Settings', font=('Arial Bold', 15), bg=background, fg=textColor)
settingsLbl.pack(pady=10)
separator = ttk.Separator(root2, orient='horizontal')
separator.pack(fill=X)
#User Photo
userProfileImg = Image.open("assets/images/avatars/a"+str(ownerPhoto)+".png")
userProfileImg = ImageTk.PhotoImage(userProfileImg.resize((120, 120)))
userPhoto = Button(root2, image=userProfileImg, bg=background, bd=0, relief=FLAT, activeback-
ground=background, command=SelectAvatar)
userPhoto.pack(pady=(20, 5))

#Change Photo
```

```
chngPh = ImageTk.PhotoImage(Image.open("assets/images/avatars/changephoto2.png").resize((120, 120)))

userPhoto.bind('<Enter>', onhover)
userPhoto.bind('<Leave>', onleave)

#Username
userName = Label(root2, text=ownerName, font=('Arial Bold', 15), fg=textColor, bg=background)
userName.pack()

#Settings Frame
settingsFrame = Frame(root2, width=300, height=300, bg=background)
settingsFrame.pack(pady=20)

assLbl = Label(settingsFrame, text='Assistant Voice', font=('Arial', 13), fg=textColor, bg=background)
assLbl.place(x=0, y=20)
n = StringVar()
assVoiceOption = ttk.Combobox(settingsFrame, values=('Female', 'Male'), font=('Arial', 13), width=13, textvariable=n)
assVoiceOption.current(voice_id)
assVoiceOption.place(x=150, y=20)
assVoiceOption.bind('<<ComboboxSelected>>', changeVoice)

voiceRateLbl = Label(settingsFrame, text='Voice Rate', font=('Arial', 13), fg=textColor, bg=background)
voiceRateLbl.place(x=0, y=60)
n2 = StringVar()
voiceOption = ttk.Combobox(settingsFrame, font=('Arial', 13), width=13, textvariable=n2)
voiceOption['values'] = ('Very Low', 'Low', 'Normal', 'Fast', 'Very Fast')
voiceOption.current(ass_voiceRate//50-2) #100 150 200 250 300
voiceOption.place(x=150, y=60)
```

```
voiceOption.bind('<<ComboboxSelected>>', changeVoiceRate)

volumeLbl = Label(settingsFrame, text='Volume', font=('Arial', 13), fg=textColor, bg=background)
volumeLbl.place(x=0, y=105)
volumeBar = Scale(settingsFrame, bg=background, fg=textColor, sliderlength=30, length=135,
width=16, highlightbackground=background, orient='horizontal', from_=0, to=100, com-
mand=changeVolume)
volumeBar.set(int(ass_volume*100))
volumeBar.place(x=150, y=85)



themeLbl = Label(settingsFrame, text='Theme', font=('Arial', 13), fg=textColor, bg=background)
themeLbl.place(x=0,y=143)
themeValue = IntVar()
s = ttk.Style()
s.configure('Wild.TRadiobutton', font=('Arial Bold', 10), background=background, fore-
ground=textColor, focuscolor=s.configure(".")["background"])
darkBtn = ttk.Radiobutton(settingsFrame, text='Dark', value=1, variable=themeValue,
style='Wild.TRadiobutton', command=changeTheme, takefocus=False)
darkBtn.place(x=150,y=145)
lightBtn = ttk.Radiobutton(settingsFrame, text='Light', value=2, variable=themeValue,
style='Wild.TRadiobutton', command=changeTheme, takefocus=False)
lightBtn.place(x=230,y=145)
themeValue.set(1)
if KCS_IMG==0: themeValue.set(2)

chooseChatLbl = Label(settingsFrame, text='Chat Background', font=('Arial', 13), fg=textColor,
bg=background)
chooseChatLbl.place(x=0,y=180)
```

```
cimg = PhotoImage(file = "assets/images/colorchooser.png")
cimg = cimg.subsample(3,3)
colorbar = Label(settingsFrame, bd=3, width=18, height=1, bg=chatBgColor)
colorbar.place(x=150, y=180)
if KCS_IMG==0: colorbar['bg'] = '#E8EBEF'
Button(settingsFrame, image=cimg, relief=FLAT, command=getChatColor).place(x=261, y=180)

backBtn = Button(settingsFrame, text='   Back   ', bd=0, font=('Arial 12'), fg='white', bg='#14A769', re-
lief=FLAT, command=lambda:raise_frame(root1))
clearFaceBtn = Button(settingsFrame, text='   Clear Facial Data   ', bd=0, font=('Arial 12'), fg='white',
bg='#14A769', relief=FLAT, command=deleteUserData)
backBtn.place(x=5, y=250)
clearFaceBtn.place(x=120, y=250)

try:
# pass
Thread(target=voiceMedium).start()
except:
pass
try:
# pass
Thread(target=web_scrapping.dataUpdate).start()
except Exception as e:
print('System is Offline...')

root.iconbitmap('assets/images/assistant2.ico')
raise_frame(root1)
root.mainloop()
```

math_function.py :

```
import math
def basicOperations(text):
if 'root' in text:
temp = text.rfind(' ')
num = int(text[temp+1:])
return round(math.sqrt(num),2)

text = text.replace('plus', '+')
text = text.replace('minus', '-')
text = text.replace('x', '*')
text = text.replace('multiplied by', '*')
text = text.replace('multiply', '*')
text = text.replace('divided by', '/')
text = text.replace('to the power', '**')
text = text.replace('power', '**')
result = eval(text)
return round(result,2)

def bitwiseOperations(text):
if 'right shift' in text:
temp = text.rfind(' ')
num = int(text[temp+1:])
return num>>1
elif 'left shift' in text:
temp = text.rfind(' ')
num = int(text[temp+1:])
return num<<1
text = text.replace('and', '&')
text = text.replace('or', '|')
text = text.replace('not of', '~')
```

```
text = text.replace('not', '~')
text = text.replace('xor', '^')
result = eval(text)
return result

def conversions(text):
temp = text.rfind(' ')
num = int(text[temp+1:])
if 'bin' in text:
return eval('bin(num)')[2:]
elif 'hex' in text:
return eval('hex(num)')[2:]
elif 'oct' in text:
return eval('oct(num)')[2:]

def trigonometry(text):
temp = text.replace('degree','')
temp = text.rfind(' ')
deg = int(text[temp+1:])
rad = (deg * math.pi) / 180
if 'sin' in text:
return round(math.sin(rad),2)
elif 'cos' in text:
return round(math.cos(rad),2)
elif 'tan' in text:
return round(math.tan(rad),2)

def factorial(n):
if n==1 or n==0: return 1
else: return n*factorial(n-1)
```

```
def logFind(text):
temp = text.rfind(' ')
num = int(text[temp+1:])
return round(math.log(num,10),2)

def isHaving(text, lst):
for word in lst:
if word in text:
return True
return False

def perform(text):
text = text.replace('math','')
if "factorial" in text: return str(factorial(int(text[text.rfind(' ')+1:])))
elif isHaving(text, ['sin','cos','tan']): return str(trigonometry(text))
elif isHaving(text, ['bin','hex','oct']): return str(conversions(text))
elif isHaving(text, ['shift','and','or','not']): return str(bitwiseOperations(text))
elif 'log' in text: return str(logFind(text))
else: return str(basicOperations(text))

# print(round(math.log(1,10),2))
```

normal_chat.py :

```
from difflib import get_close_matches
import json
from random import choice
import datetime

class DateTime:
def currentTime(self):
time = datetime.datetime.now()
```

```
x = " A.M."
if time.hour>12: x = " P.M."
time = str(time)
time = time[11:16] + x
return time

def currentDate(self):
now = datetime.datetime.now()
day = now.strftime('%A')
date = str(now)[8:10]
month = now.strftime('%B')
year = str(now.year)
result = f'{day}, {date} {month}, {year}'
return result

def wishMe():
now = datetime.datetime.now()
hr = now.hour
if hr<12:
wish="Good Morning"
elif hr>=12 and hr<16:
wish="Good Afternoon"
else:
wish="Good Evening"
return wish


def isContain(text, lst):
for word in lst:
if word in text:
return True
```

```
return False

def chat(text):
dt = DateTime()
result = ""
if isContain(text, ['good']):
result = wishMe()
elif isContain(text, ['time']):
result = "Current Time is: " + dt.currentTime()
elif isContain(text, ['date','today','day','month']):
result = dt.currentDate()

return result

data = json.load(open('assets/normal_chat.json', encoding='utf-8'))

def reply(query):
if query in data:
response =  data[query]
else:
query = get_close_matches(query, data.keys(), n=2, cutoff=0.6)
if len(query)==0: return "None"
return choice(data[query[0]])

return choice(response)

def lang_translate(text,language):
from googletrans import Translator, LANGUAGES
if language in LANGUAGES.values():
translator = Translator()
result = translator.translate(text, src='en', dest=language)
```

```
return result
else:
return "None"
```

security.py :

```
from tkinter import *
from tkinter import ttk
from PIL import Image, ImageTk
import cv2
import numpy as np
import os
from os.path import isfile, join
from threading import Thread
from .user_handler import UserData
from . import face_unlocker as FU

background, textColor = 'black', '#F6FAFB'
background, textColor = textColor, background

avatarChoosen = 0
choosedAvtrImage = None
user_name = ''
user_gender = ''

try:
face_classifier = cv2.CascadeClassifier('Cascade/haarcascade_frontalface_default.xml')
except Exception as e:
print('Cascade File is missing...')
raise SystemExit

if os.path.exists('userData')==False:
```

```
os.mkdir('userData')
if os.path.exists('userData/faceData')==False:
os.mkdir('userData/faceData')


###### ROOT1 ########
def startLogin():
try:
result = FU.startDetecting()
if result:
user = UserData()
user.extractData()
userName = user.getName().split()[0]
welcLbl['text'] = 'Hi '+userName+',\nWelcome to the world of\nScience & Technology'
loginStatus['text'] = 'UNLOCKED'
loginStatus['fg'] = 'green'
faceStatus['text']='(Logged In)'
os.system('python modules/gui_assistant.py')
else:
print('Error Occurred')

except Exception as e:
print(e)

####### ROOT2 ########
def trainFace():
data_path = 'userData/faceData/'
onlyfiles = [f for f in os.listdir(data_path) if isfile(join(data_path, f))]

Training_data = []
Labels = []
```

```
for i, files in enumerate(onlyfiles):
image_path = data_path + onlyfiles[i]
images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

Training_data.append(np.asarray(images, dtype=np.uint8))
Labels.append(i)



Labels = np.asarray(Labels, dtype=np.int32)

model = cv2.face.LBPHFaceRecognizer_create()
model.train(np.asarray(Training_data), np.asarray(Labels))

print('Model Trained Successfully !!!')
model.save('userData/trainer.yml')
print('Model Saved !!!')

def face_extractor(img):
global cropped_face
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale(gray, 1.3, 5)

if faces is ():
return None

for (x, y, w, h) in faces:
cropped_face = img[y:y+h, x:x+w]

return cropped_face
```

```
cap = None
count = 0
def startCapturing():
global count, cap
ret, frame = cap.read()
if face_extractor(frame) is not None:
count += 1
face = cv2.resize(face_extractor(frame), (200, 200))
face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

file_name_path = 'userData/faceData/img' + str(count) + '.png'
cv2.imwrite(file_name_path, face)
print(count)
progress_bar['value'] = count

cv2.putText(face, str(count), (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
else:
pass

if count==100:
progress_bar.destroy()
lmain['image'] = defaultImg2
statusLbl['text'] = '(Face added successfully)'
cap.release()
cv2.destroyAllWindows()
Thread(target=trainFace).start()
addBtn['text'] = '       Next       '
addBtn['command'] = lambda:raise_frame(root3)
return

frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
```

```
frame = cv2.flip(frame, 1)
img = Image.fromarray(frame)
imgtk = ImageTk.PhotoImage(image=img)
lmain.imgtk = imgtk
lmain.configure(image=imgtk)
lmain.after(10, startCapturing)

def Add_Face():

global cap, user_name, user_gender
user_name = nameField.get()
user_gender = r.get()
if user_name != '' and user_gender!=0:
if agr.get()==1:
cap = cv2.VideoCapture(0)
startCapturing()
progress_bar.place(x=20, y=273)
statusLbl['text'] = ''
else:
statusLbl['text'] = '(Check the Condition)'
else:
statusLbl['text'] = '(Please fill the details)'


def SuccessfullyRegistered():
if avatarChoosen != 0:
gen = 'Male'
if user_gender==2: gen = 'Female'
u = UserData()
u.updateData(user_name, gen, avatarChoosen)
usernameLbl['text'] = user_name
```

```
raise_frame(root4)

def selectAVATAR(avt=0):
global avatarChoosen, choosedAvtrImage
avatarChoosen = avt
i=1
for avtr in (avtb1,avtb2,avtb3,avtb4,avtb5,avtb6,avtb7,avtb8):
if i==avt:
avtr['state'] = 'disabled'
userPIC['image'] = avtr['image']
else: avtr['state'] = 'normal'
i+=1


################################################## GUI
################################


def raise_frame(frame):
frame.tkraise()


root = Tk()
root.title('F.R.I.D.A.Y.')
w_width, w_height = 350, 600
s_width, s_height = root.winfo_screenwidth(), root.winfo_screenheight()
x, y = (s_width/2)-(w_width/2), (s_height/2)-(w_height/2)
root.geometry('%dx%d+%d+%d' % (w_width,w_height,x,y-30)) #center location of the screen
root.configure(bg=background)
# root.attributes('-toolwindow', True)
root1 = Frame(root, bg=background)
```

```
root2 = Frame(root, bg=background)
root3 = Frame(root, bg=background)
root4 = Frame(root, bg=background)

for f in (root1, root2, root3, root4):
f.grid(row=0, column=0, sticky='news')

################################
########  MAIN SCREEN  #########
################################

image1 = Image.open('assets/images/home2.jpg')
image1 = image1.resize((300,250))
defaultImg1 = ImageTk.PhotoImage(image1)

dataFrame1 = Frame(root1, bd=10, bg=background)
dataFrame1.pack()
logo = Label(dataFrame1, width=300, height=250, image=defaultImg1)
logo.pack(padx=10, pady=10)

#welcome label
welcLbl = Label(root1, text='Hi there,\nWelcome to the world of\nScience & Technology', font=('Arial
Bold', 15), fg='#303E54', bg=background)
welcLbl.pack(padx=10, pady=20)

#add face
loginStatus = Label(root1, text='LOCKED', font=('Arial Bold', 15), bg=background, fg='red')
loginStatus.pack(pady=(40,20))

if os.path.exists('userData/trainer.yml')==False:
loginStatus['text'] = 'Your Face is not registered'
```

```
addFace = Button(root1, text='   Register Face   ', font=('Arial', 12), bg='#018384', fg='white', re-
lief=FLAT, command=lambda:raise_frame(root2))
addFace.pack(ipadx=10)
else:
Thread(target=startLogin).start()

#status of add face
faceStatus = Label(root1, text='(Face Not Detected)', font=('Arial 10'), fg=textColor, bg=background)
faceStatus.pack(pady=5)

###################################
########  FACE ADD SCREEN  #######
###################################

image2 = Image.open('assets/images/defaultFace4.png')
image2 = image2.resize((300, 250))
defaultImg2 = ImageTk.PhotoImage(image2)

dataFrame2 = Frame(root2, bd=10, bg=background)
dataFrame2.pack(fill=X)
lmain = Label(dataFrame2, width=300, height=250, image=defaultImg2)
lmain.pack(padx=10, pady=10)

#Details
detailFrame2 = Frame(root2, bd=10, bg=background)
detailFrame2.pack(fill=X)
userFrame2 = Frame(detailFrame2, bd=10, width=300, height=250, relief=FLAT, bg=background)
userFrame2.pack(padx=10, pady=10)

#progress
progress_bar = ttk.Progressbar(root2, orient=HORIZONTAL, length=303, mode='determinate')
```

```
#name
nameLbl = Label(userFrame2, text='Name', font=('Arial Bold', 12), fg='#303E54', bg=background)
nameLbl.place(x=10,y=10)
nameField = Entry(userFrame2, bd=5, font=('Arial Bold', 10), width=25, relief=FLAT, bg='#D4D5D7')
nameField.focus()
nameField.place(x=80,y=10)

genLbl = Label(userFrame2, text='Gender', font=('Arial Bold', 12), fg='#303E54', bg=background)
genLbl.place(x=10,y=50)
r = IntVar()
s = ttk.Style()
s.configure('Wild.TRadiobutton', background=background, foreground=textColor, font=('Arial Bold',
10), focuscolor=s.configure(".")["background"])
genMale = ttk.Radiobutton(userFrame2, text='Male', value=1, variable=r, style='Wild.TRadiobutton',
takefocus=False)
genMale.place(x=80,y=52)
genFemale = ttk.Radiobutton(userFrame2, text='Female', value=2, variable=r,
style='Wild.TRadiobutton', takefocus=False)
genFemale.place(x=150,y=52)

#agreement
agr = IntVar()
sc = ttk.Style()
sc.configure('Wild.TCheckbutton', background=background, foreground='#303E54', font=('Arial
Bold',10), focuscolor=sc.configure(".")["background"])
# agree = Checkbutton(userFrame2, text='I agree to use my face for Security purpose', fg=textColor,
bg=background, activebackground=background, activeforeground=textColor)
agree = ttk.Checkbutton(userFrame2, text='I agree to use my Face for Security',
style='Wild.TCheckbutton', takefocus=False, variable=agr)
agree.place(x=28, y=100)
```

```
#add face
addBtn = Button(userFrame2, text='    Add Face    ', font=('Arial Bold', 12), bg='#01933B', fg='white',
command=Add_Face, relief=FLAT)
addBtn.place(x=90, y=150)

#status of add face
statusLbl = Label(userFrame2, text='', font=('Arial 10'), fg=textColor, bg=background)
statusLbl.place(x=80, y=190)

##########################
#### AVATAR SELECTION ####
##########################

Label(root3, text="Choose Your Avatar", font=('arial', 15), bg=background, fg='#303E54').pack()

avatarContainer = Frame(root3, bg=background, width=300, height=500)
avatarContainer.pack(pady=10)
size = 100

avtr1 = Image.open('assets/images/avatars/a1.png')
avtr1 = avtr1.resize((size, size))
avtr1 = ImageTk.PhotoImage(avtr1)
avtr2 = Image.open('assets/images/avatars/a2.png')
avtr2 = avtr2.resize((size, size))
avtr2 = ImageTk.PhotoImage(avtr2)
avtr3 = Image.open('assets/images/avatars/a3.png')
avtr3 = avtr3.resize((size, size))
avtr3 = ImageTk.PhotoImage(avtr3)
avtr4 = Image.open('assets/images/avatars/a4.png')
avtr4 = avtr4.resize((size, size))
avtr4 = ImageTk.PhotoImage(avtr4)
```

```
avtr5 = Image.open('assets/images/avatars/a5.png')
avtr5 = avtr5.resize((size, size))
avtr5 = ImageTk.PhotoImage(avtr5)
avtr6 = Image.open('assets/images/avatars/a6.png')
avtr6 = avtr6.resize((size, size))
avtr6 = ImageTk.PhotoImage(avtr6)
avtr7 = Image.open('assets/images/avatars/a7.png')
avtr7 = avtr7.resize((size, size))
avtr7 = ImageTk.PhotoImage(avtr7)
avtr8 = Image.open('assets/images/avatars/a8.png')
avtr8 = avtr8.resize((size, size))
avtr8 = ImageTk.PhotoImage(avtr8)


avtb1 = Button(avatarContainer, image=avtr1, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(1))
avtb1.grid(row=0, column=0, ipadx=25, ipady=10)

avtb2 = Button(avatarContainer, image=avtr2, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(2))
avtb2.grid(row=0, column=1, ipadx=25, ipady=10)

avtb3 = Button(avatarContainer, image=avtr3, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(3))
avtb3.grid(row=1, column=0, ipadx=25, ipady=10)

avtb4 = Button(avatarContainer, image=avtr4, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(4))
avtb4.grid(row=1, column=1, ipadx=25, ipady=10)

avtb5 = Button(avatarContainer, image=avtr5, bg=background, activebackground=background, re-
```

```
lief=FLAT, bd=0, command=lambda:selectAVATAR(5))
avtb5.grid(row=2, column=0, ipadx=25, ipady=10)

avtb6 = Button(avatarContainer, image=avtr6, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(6))
avtb6.grid(row=2, column=1, ipadx=25, ipady=10)

avtb7 = Button(avatarContainer, image=avtr7, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(7))
avtb7.grid(row=3, column=0, ipadx=25, ipady=10)

avtb8 = Button(avatarContainer, image=avtr8, bg=background, activebackground=background, re-
lief=FLAT, bd=0, command=lambda:selectAVATAR(8))
avtb8.grid(row=3, column=1, ipadx=25, ipady=10)


Button(root3, text='        Submit        ', font=('Arial Bold', 15), bg='#01933B', fg='white', bd=0, re-
lief=FLAT, command=SuccessfullyRegistered).pack()

########################################
######## SUCCESSFULL REGISTRATION #######
########################################

userPIC = Label(root4, bg=background, image=avtr1)
userPIC.pack(pady=(40, 10))
usernameLbl = Label(root4, text="Roshan Kumar", font=('Arial Bold',15), bg=background,
fg='#85AD4F')
usernameLbl.pack(pady=(0, 70))

Label(root4, text="Your account has been successfully activated!", font=('Arial Bold',15),
bg=background, fg='#303E54', wraplength=300).pack(pady=10)
```

```
Label(root4, text="Launch the APP again to get started the conversation with your Personal Assistant",
font=('arial',13), bg=background, fg='#A3A5AB', wraplength=350).pack()

Button(root4, text='     OK     ', bg='#0475BB', fg='white',font=('Arial Bold', 18), bd=0, relief=FLAT,
command=lambda:quit()).pack(pady=50)

root.iconbitmap('assets/images/assistant2.ico')
raise_frame(root1)
root.mainloop()
```

user_handler.py :

```
import pickle

class UserData:
def __init__(self):
self.name = 'None'
self.gender = 'None'
self.userphoto = 0

def extractData(self):
with open('userData/userData.pck', 'rb') as file:
details = pickle.load(file)
self.name, self.gender, self.userphoto = details['name'], details['gender'], details['userphoto']

def updateData(self, name, gender, userphoto):
with open('userData/userData.pck', 'wb') as file:
details = {'name': name, 'gender': gender, 'userphoto': userphoto}
pickle.dump(details, file)

def getName(self):
return self.name
```

```
def getGender(self):
return self.gender

def getUserPhoto(self):
return self.userphoto

def UpdateUserPhoto(avatar):
u = UserData()
u.extractData()
u.updateData(u.getName(), u.getGender(), avatar)
```

web_scrapping.py :

```
import wikipedia
import webbrowser
import requests
from bs4 import BeautifulSoup
import smtplib
import urllib.request
import os
from geopy.geocoders import Nominatim
from geopy.distance import great_circle

class COVID:
def __init__(self):
self.total = 'Not Available'
self.deaths = 'Not Available'
self.recovered = 'Not Available'
self.totalIndia = 'Not Available'
self.deathsIndia = 'Not Available'
self.recoveredIndia = 'Not Available'
```

```
def covidUpdate(self):
URL = 'https://www.worldometers.info/coronavirus/'
result = requests.get(URL)
src = result.content
soup = BeautifulSoup(src, 'html.parser')

temp = []
divs = soup.find_all('div', class_='maincounter-number')
for div in divs:
temp.append(div.text.strip())
self.total, self.deaths, self.recovered = temp[0], temp[1], temp[2]

def covidUpdateIndia(self):
URL = 'https://www.worldometers.info/coronavirus/country/india/'
result = requests.get(URL)
src = result.content
soup = BeautifulSoup(src, 'html.parser')

temp = []
divs = soup.find_all('div', class_='maincounter-number')
for div in divs:
temp.append(div.text.strip())
self.totalIndia, self.deathsIndia, self.recoveredIndia = temp[0], temp[1], temp[2]

def totalCases(self,india_bool):
if india_bool: return self.totalIndia
return self.total

def totalDeaths(self,india_bool):
if india_bool: return self.deathsIndia
return self.deaths
```

```
def totalRecovery(self,india_bool):
if india_bool: return self.recoveredIndia
return self.recovered

def symptoms(self):
symt = ['1. Fever',
'2. Coughing',
'3. Shortness of breath',
'4. Trouble breathing',
'5. Fatigue',
'6. Chills, sometimes with shaking',
'7. Body aches',
'8. Headache',
'9. Sore throat',
'10. Loss of smell or taste',
'11. Nausea',
'12. Diarrhea']
return symt

def prevention(self):
prevention = ['1. Clean your hands often. Use soap and water, or an alcohol-based hand rub.',
'2. Maintain a safe distance from anyone who is coughing or sneezing.',
'3. Wear a mask when physical distancing is not possible.',
'4. Don’t touch your eyes, nose or mouth.',
'5. Cover your nose and mouth with your bent elbow or a tissue when you cough or sneeze.',
'6. Stay home if you feel unwell.',
'7. If you have a fever, cough and difficulty breathing, seek medical attention.']
return prevention

def wikiResult(query):
```

```
query = query.replace('wikipedia','')
query = query.replace('search','')
if len(query.split())==0: query = "wikipedia"
try:
return wikipedia.summary(query, sentences=2)
except Exception as e:
return "Desired Result Not Found"

class WEATHER:
def __init__(self):
#Currently in Lucknow, its 26 with Haze
self.tempValue = ''
self.city = ''
self.currCondition = ''
self.speakResult = ''

def updateWeather(self):
res = requests.get("https://ipinfo.io/")
data = res.json()
# URL = 'https://weather.com/en-IN/weather/today/l/'+data['loc']
URL = 'https://weather.com/en-IN/weather/today/'
result = requests.get(URL)
src = result.content

soup = BeautifulSoup(src, 'html.parser')

city = ""
for h in soup.find_all('h1'):
cty = h.text
cty = cty.replace('Weather','')
self.city = cty[:cty.find(',')]
```

```
break

spans = soup.find_all('span')
for span in spans:
try:
if span['data-testid']=="TemperatureValue":
self.tempValue = span.text[:-1]
break
except Exception as e:
pass

divs = soup.find_all('div', class_='CurrentConditions--phraseValue--2xXSr')
for div in divs:
self.currCondition = div.text
break

def weather(self):
from datetime import datetime
today = datetime.today().strftime('%A')
self.speakResult = "Currently in " + self.city + ", its " + self.tempValue + " degree, with a " +
self.currCondition
return [self.tempValue, self.currCondition, today, self.city, self.speakResult]

c = COVID()
w = WEATHER()

def dataUpdate():
c.covidUpdate()
c.covidUpdateIndia()
w.updateWeather()
```

```
##### WEATHER #####
def weather():
return w.weather()

### COVID ###
def covid(query):

if "india" in query: india_bool = True
else: india_bool = False

if "statistic" in query or 'report' in query:
return ["Here are the statistics...", ["Total cases: " + c.totalCases(india_bool), "Total Recovery: " +
c.totalRecovery(india_bool), "Total Deaths: " + c.totalDeaths(india_bool)]]

elif "symptom" in query:
return ["Here are the Symptoms...", c.symptoms()]

elif "prevent" in query or "measure" in query or "precaution" in query:
return ["Here are the some of preventions from COVID-19:", c.prevention()]

elif "recov" in query:
return "Total Recovery is: " + c.totalRecovery(india_bool)

elif "death" in query:
return "Total Deaths are: " + c.totalDeaths(india_bool)

else:
return "Total Cases are: " + c.totalCases(india_bool)

def latestNews(news=5):
URL = 'https://indianexpress.com/latest-news/'
```

```
result = requests.get(URL)
src = result.content

soup = BeautifulSoup(src, 'html.parser')

headlineLinks = []
headlines = []

divs = soup.find_all('div', {'class':'title'})

count=0
for div in divs:
count += 1
if count>news:
break
a_tag = div.find('a')
headlineLinks.append(a_tag.attrs['href'])
headlines.append(a_tag.text)

return headlines,headlineLinks

def maps(text):
text = text.replace('maps', '')
text = text.replace('map', '')
text = text.replace('google', '')
openWebsite('https://www.google.com/maps/place/'+text)

def giveDirections(startingPoint, destinationPoint):

geolocator = Nominatim(user_agent='assistant')
if 'current' in startingPoint:
```

```
res = requests.get("https://ipinfo.io/")
data = res.json()
startinglocation = geolocator.reverse(data['loc'])
else:
startinglocation = geolocator.geocode(startingPoint)

destinationlocation = geolocator.geocode(destinationPoint)
startingPoint = startinglocation.address.replace(' ', '+')
destinationPoint = destinationlocation.address.replace(' ', '+')

openWebsite('https://www.google.co.in/maps/dir/'+startingPoint+'/'+destinationPoint+'/')

startinglocationCoordinate = (startinglocation.latitude, startinglocation.longitude)
destinationlocationCoordinate = (destinationlocation.latitude, destinationlocation.longitude)
total_distance = great_circle(startinglocationCoordinate, destinationlocationCoordinate).km #.mile
return str(round(total_distance, 2)) + 'KM'

def openWebsite(url='https://www.google.com/'):
webbrowser.open(url)

def jokes():
URL = 'https://icanhazdadjoke.com/'
result = requests.get(URL)
src = result.content

soup = BeautifulSoup(src, 'html.parser')

try:
p = soup.find('p')
return p.text
except Exception as e:
```

```
raise e


def youtube(query):
query = query.replace('play',' ')
query = query.replace('on youtube',' ')
query = query.replace('youtube',' ')

print("Searching for videos...")
from youtubesearchpython import VideosSearch
videosSearch = VideosSearch(query, limit = 1)
results = videosSearch.result()['result']
print("Finished searching!")

webbrowser.open('https://www.youtube.com/watch?v=' + results[0]['id'])
return "Enjoy..."


def googleSearch(query):
if 'image' in query:
query += "&tbm=isch"
query = query.replace('images','')
query = query.replace('image','')
query = query.replace('search','')
query = query.replace('show','')
webbrowser.open("https://www.google.com/search?q=" + query)
return "Here you go..."

def sendWhatsapp(phone_no='',message=''):
phone_no = '+91' + str(phone_no)
webbrowser.open('https://web.whatsapp.com/send?phone='+phone_no+'&text='+message)
```

```
import time
from pynput.keyboard import Key, Controller
time.sleep(10)
k = Controller()
k.press(Key.enter)

def email(rec_email=None, text="Hello, It's F.R.I.D.A.Y. here...", sub='F.R.I.D.A.Y.'):
USERNAME = os.getenv('MAIL_USERNAME') # email address
PASSWORD = os.getenv('MAIL_PASSWORD')
if not USERNAME or not PASSWORD:
raise Exception("MAIL_USERNAME or MAIL_PASSWORD are not loaded in environment, create a
.env file and add these 2 values")

if '@gmail.com' not in rec_email: return
s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()
s.login(USERNAME, PASSWORD)
message = 'Subject: {}\n\n{}'.format(sub, text)
s.sendmail(USERNAME, rec_email, message)
print("Sent")
s.quit()


def downloadImage(query, n=4):
query = query.replace('images','')
query = query.replace('image','')
query = query.replace('search','')
query = query.replace('show','')
URL = "https://www.google.com/search?tbm=isch&q=" + query
result = requests.get(URL)
src = result.content
```

```
soup = BeautifulSoup(src, 'html.parser')
imgTags = soup.find_all('img', class_='yWs4tf') # old class name -> t0fcAb

if os.path.exists('Downloads')==False:
os.mkdir('Downloads')

count=0
for i in imgTags:
if count==n: break
try:
urllib.request.urlretrieve(i['src'], 'Downloads/' + str(count) + '.jpg')
count+=1
print('Downloaded', count)
except Exception as e:
raise e
```