PHASE 3: COMMUNICATION AND FUTURE EXPLORATION

1. ABSTRACT:

Tara stands as a groundbreaking AI assistant meticulously crafted to elevate daily productivity through seamless voice interaction and automation. At its core, Tara harnesses cutting-edge AI technologies and Python libraries to execute an extensive array of tasks. These encompass managing emails, navigating the web, system administration, and delivering real-time updates on weather, news, and beyond. By exemplifying how AI can seamlessly integrate into everyday life, this project demonstrates the potential of Tara as a reliable, efficient, and indispensable personal assistant.

`pyttsx3` falls under the category of Natural Language Processing (NLP) in the broader field of Artificial Intelligence (AI).

Natural Language Processing involves the interaction between computers and human (natural) languages. In the case of `pyttsx3`, it is specifically concerned with the generation of human-like speech from text. This process typically involves the use of machine learning algorithms and linguistic rules to convert written text into spoken words, mimicking the way humans speak.

Within NLP, text-to-speech (TTS) conversion is an important area that enables various applications such as virtual assistants, accessibility tools for visually impaired individuals, language learning platforms, and more.

So, while `pyttsx3` itself is not an AI algorithm, it provides a useful tool for developers working in the field of NLP to incorporate speech synthesis capabilities into their AI applications.

2. ARTIFICIAL INTELLIGENCE IN AI_ASSISTANT:

AI_ASSISTANT utilizes cutting-edge artificial intelligence techniques to provide a seamless user experience. Leveraging natural language processing (NLP), speech recognition, and machine learning, AI_ASSISTANT can understand and respond to user queries in real-time. Its core functionalities include voice interaction, task automation, information retrieval, and personalized user support. By continuously learning from user interactions, AI_ASSISTANT improves its accuracy and efficiency over time.

3. FRAMEWORK FOR AI_ASSISTANT:

The framework for AI_ASSISTANT is designed to be modular and scalable, consisting of several key components:

- **Voice Processing Module**: Handles speech recognition and synthesis using the pyttsx3 and speech_recognition libraries.
- **Task Automation Module**: Automates routine tasks such as taking screenshots, monitoring CPU usage, and fetching the weather.
- **Information Retrieval Module**: Integrates with APIs to provide real-time information like news updates and Wikipedia summaries.
- **User Interface Module**: Provides a graphical interface using tkinter, allowing users to interact with the assistant via text input or voice commands.

4. SYSTEMATIC MODEL:

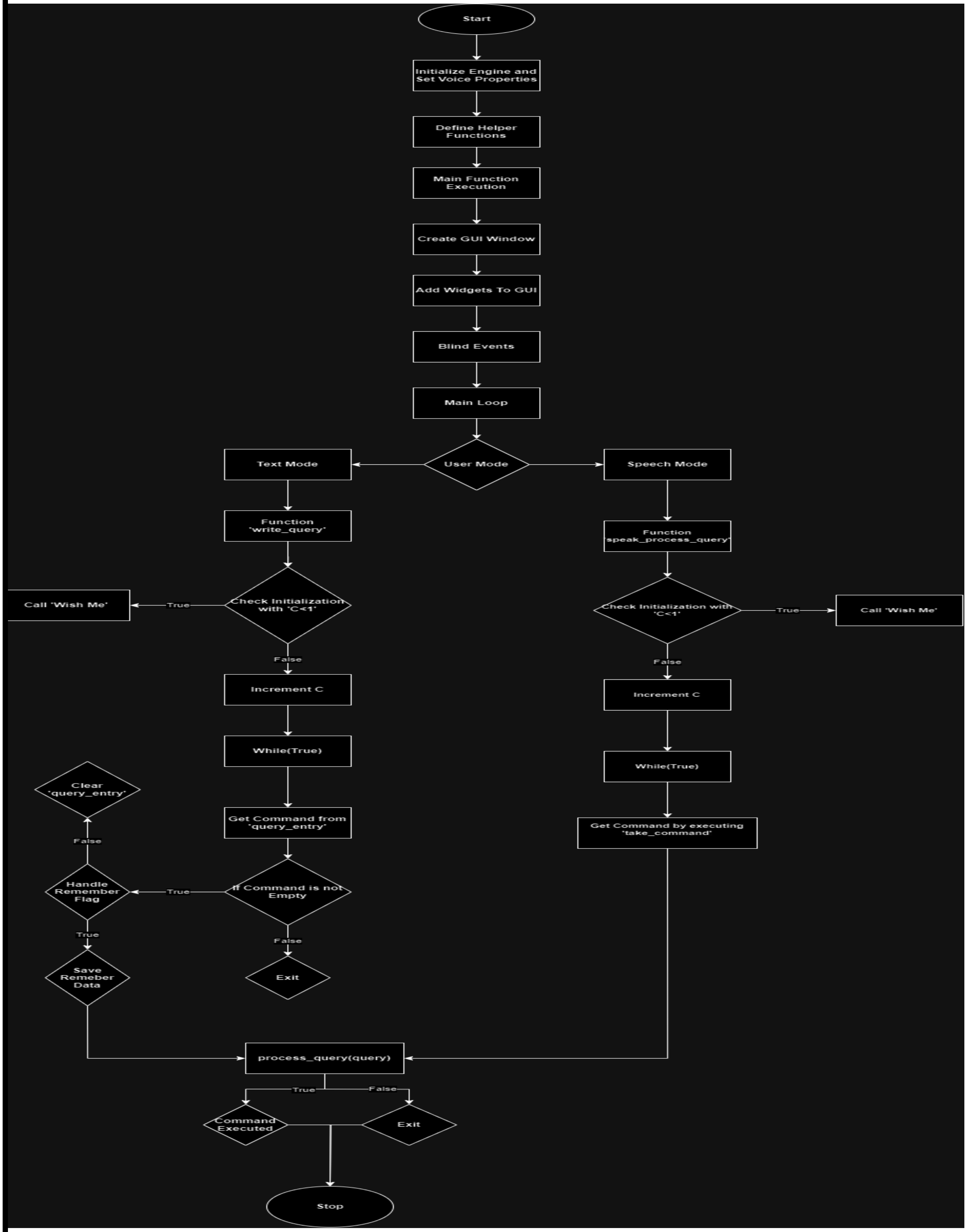
The systematic model for AI_ASSISTANT involves the following steps:

1. **Initialization**: Set up the speech engine and voice properties.
2. **Greeting**: The assistant greets the user based on the time of day.
3. **Listening**: The assistant listens for voice commands or accepts text input.
4. **Processing**: The assistant processes the user's query and determines the appropriate action.
5. **Execution**: The assistant executes the command, such as fetching information, performing a task, or providing a response.
6. **Feedback**: The assistant provides feedback to the user through speech or text, ensuring a responsive interaction.

5. Hardware Requirements:

- **Processor: Intel Core i5 or equivalent**
- **RAM: Minimum 4 GB (8 GB recommended)**
- **Storage: At least 500 MB of free disk space**
- **Microphone: For voice command input**
- **Speakers: For audio output**

6.FLOWCHART:


Start
Initialize Engine and Set Voice Properties
Define Helper Functions
Main Function Execution
Create GUI Window
Add Widgets To GUI
Blind Events
Main Loop
User Mode
Text Mode
Speech Mode
Function 'write_query'
Function 'speak_process_query'
Check Initialization with 'C<1'
True
Call 'Wish Me'
False
Increment C
While(True)
Get Command from 'query_entry'
Get Command by executing 'take_command'
If Command is not Empty
Handle Remember Flag
Clear 'query_entry'
Save Remeber Data
Exit
process_query(query)
Command Executed
Stop

7. Software Requirements:

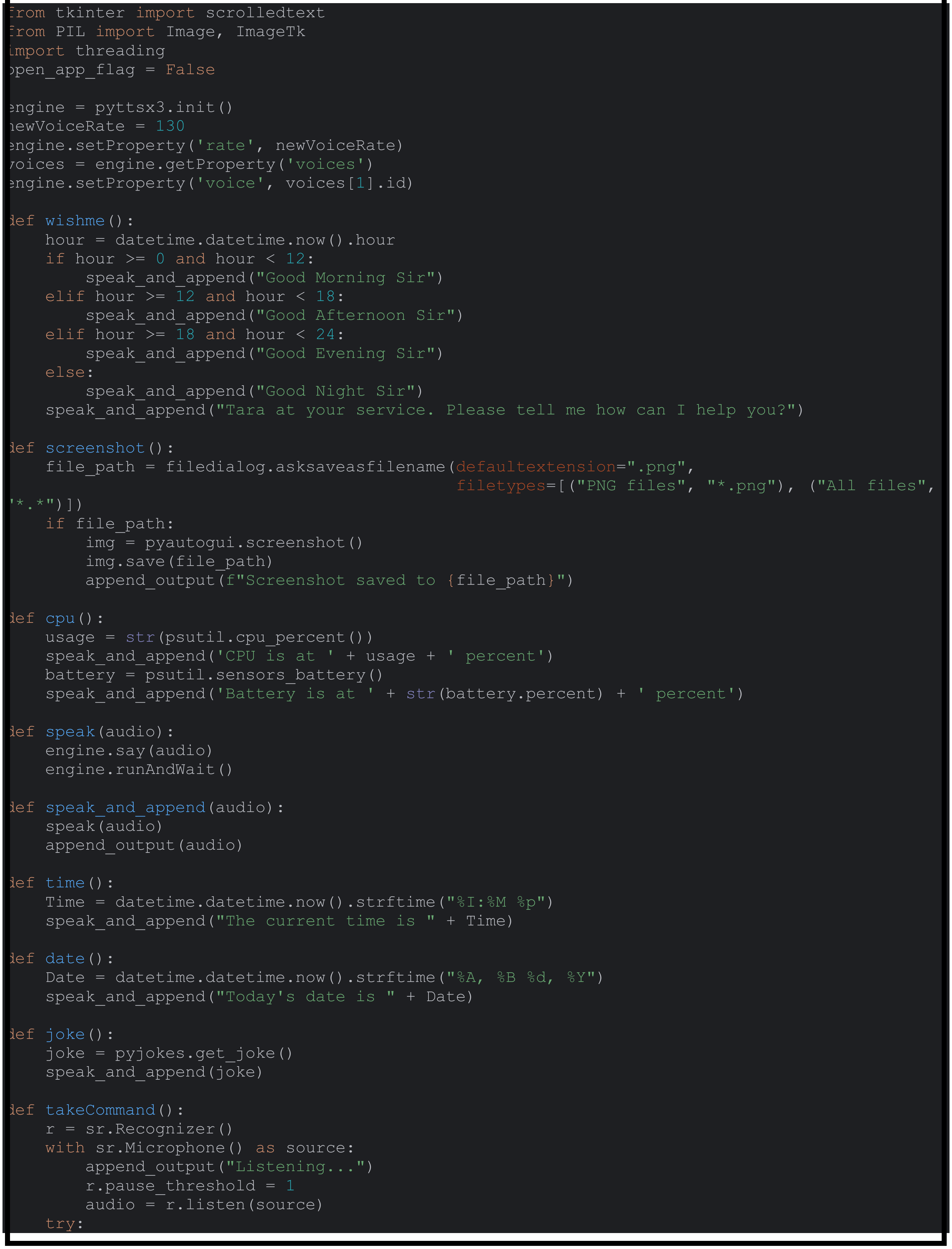
- **Operating System: Windows 10 or later**
- **Python: Version 3.7 or later**
- **Internet Connection: Required for web-based functionalities like email, weather updates, and news retrieval.**

8. Tools and Versions:

- **Python**: Version 3.7+
- **pyttsx3**: 2.90
- **datetime**: Standard Python library
- **speech_recognition**: 3.8.1
- **wikipedia**: 1.4.0
- **smtplib**: Standard Python library
- **webbrowser**: Standard Python library
- **os**: Standard Python library
- **pyautogui**: 0.9.53
- **psutil**: 5.9.0
- **pyjokes**: 0.6.0
- **requests**: 2.25.1
- **json**: Standard Python library

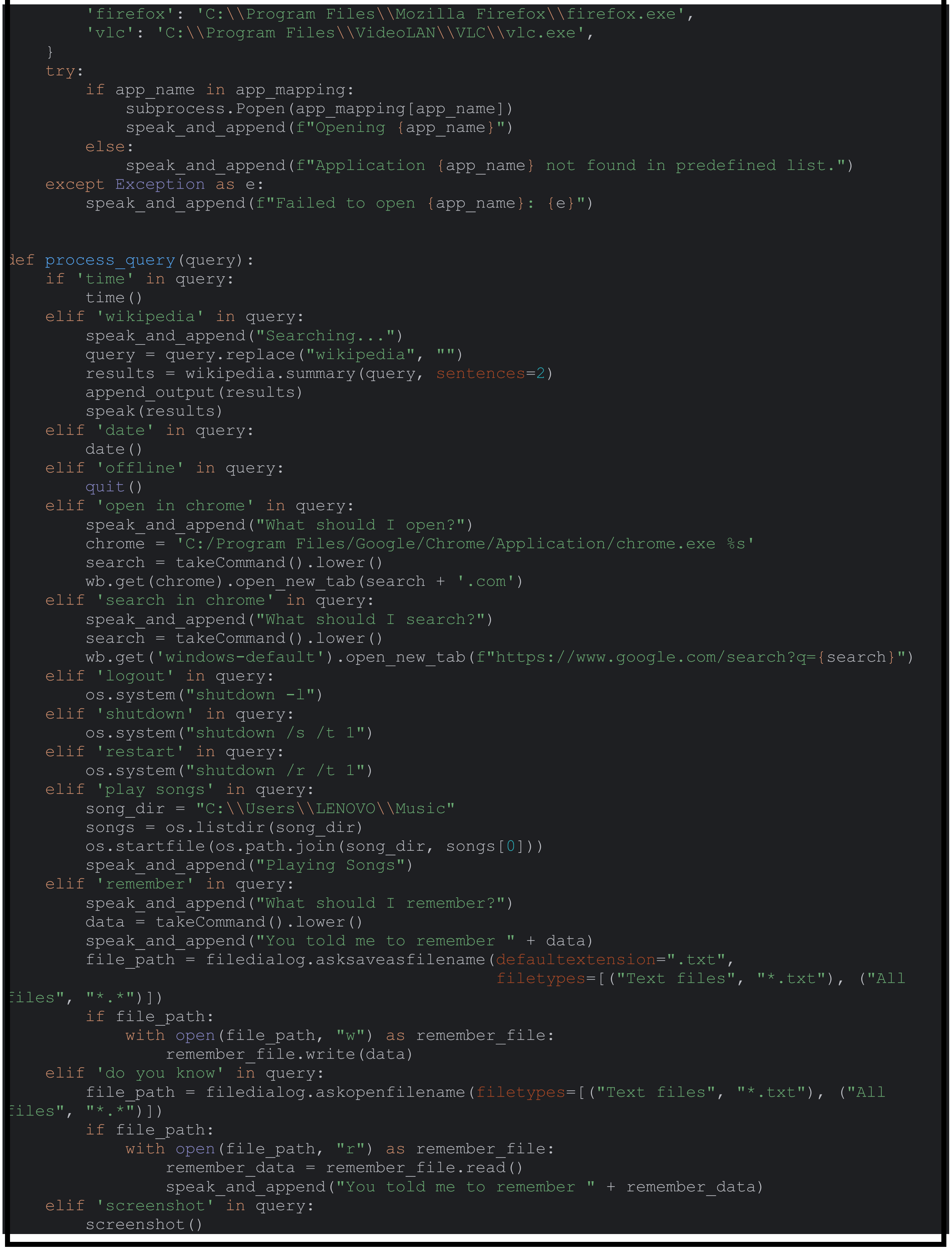
CODE IMPLEMENTATION:

```
import pyttimport pyttsx3
import datetime
import speech_recognition as sr
import wikipedia
import time
import webbrowser as wb
import os
import pyautogui
import psutil
import pyjokes
import requests
import subprocess
from tkinter import  filedialog
import tkinter as tk
```

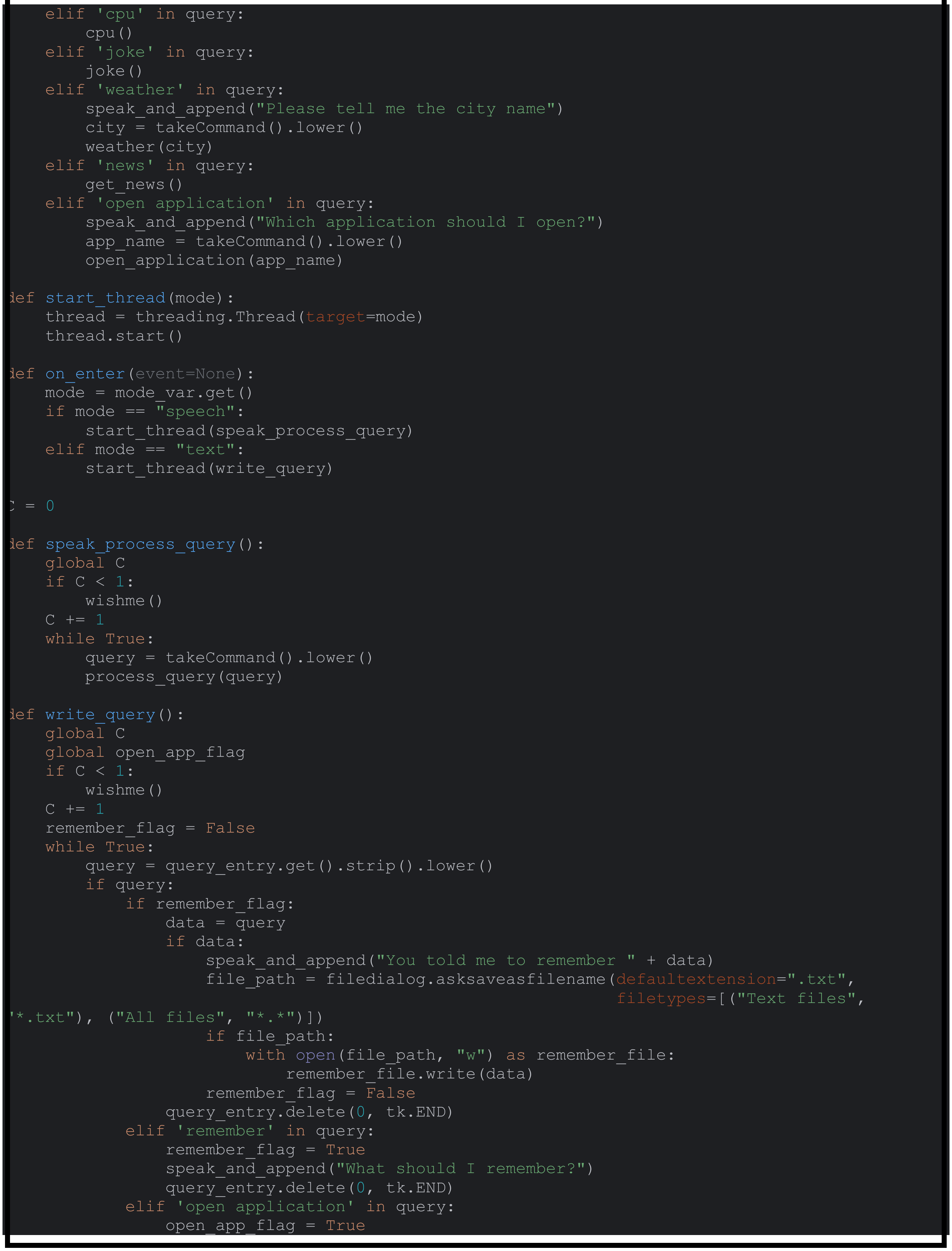


```
from tkinter import scrolledtext
from PIL import Image, ImageTk
import threading
pen_app_flag = False

ngine = pyttsx3.init()
ewVoiceRate = 130
ngine.setProperty('rate', newVoiceRate)
oices = engine.getProperty('voices')
ngine.setProperty('voice', voices[1].id)

ef wishme():
    hour = datetime.datetime.now().hour
    if hour >= 0 and hour < 12:
        speak_and_append("Good Morning Sir")
    elif hour >= 12 and hour < 18:
        speak_and_append("Good Afternoon Sir")
    elif hour >= 18 and hour < 24:
        speak_and_append("Good Evening Sir")
    else:
        speak_and_append("Good Night Sir")
    speak_and_append("Tara at your service. Please tell me how can I help you?")

ef screenshot():
    file_path = filedialog.asksaveasfilename(defaultextension=".png",
                                             filetypes=[("PNG files", "*.png"), ("All files",
"*.*")])
    if file_path:
        img = pyautogui.screenshot()
        img.save(file_path)
        append_output(f"Screenshot saved to {file_path}")

ef cpu():
    usage = str(psutil.cpu_percent())
    speak_and_append('CPU is at ' + usage + ' percent')
    battery = psutil.sensors_battery()
    speak_and_append('Battery is at ' + str(battery.percent) + ' percent')

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

ef speak_and_append(audio):
    speak(audio)
    append_output(audio)

ef time():
    Time = datetime.datetime.now().strftime("%I:%M %p")
    speak_and_append("The current time is " + Time)

ef date():
    Date = datetime.datetime.now().strftime("%A, %B %d, %Y")
    speak_and_append("Today's date is " + Date)

ef joke():
    joke = pyjokes.get_joke()
    speak_and_append(joke)

ef takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        append_output("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)
    try:
```

```python
        append_output("Recognizing...")
        query = r.recognize_google(audio, language='en-in')
        append_output(f" {query}\n")
    except Exception as e:
        append_output("Error: " + str(e))
        speak_and_append("Say that again please...")
        return "None"
    return query

def weather(city):
    api_key = "b52c66bcd330f1661de28426f176faac"
    base_url = "http://api.openweathermap.org/data/2.5/weather?"
    complete_url = base_url + "appid=" + api_key + "&q=" + city
    response = requests.get(complete_url)
    data = response.json()
    if data["cod"] != "404":
        main = data["main"]
        weather_desc = data["weather"][0]["description"]
        temp = main["temp"]
        temp_celsius = temp - 273.15
        result = f"The temperature in {city} is {temp_celsius:.2f} degrees Celsius with
{weather_desc}."
        speak_and_append(result)
    else:
        speak_and_append("City not found")

def get_news():
    api_key = "81a16fa94dc54006bd497762913c248a"
    base_url = "https://newsapi.org/v2/top-headlines"
    params = {
        "apiKey": api_key,
        "country": "us",
        "pageSize": 5
    }
    response = requests.get(base_url, params=params)
    if response.status_code == 200:
        news_data = response.json()
        articles = news_data.get("articles", [])
        if articles:
            speak_and_append("Here are the top news headlines")
            for article in articles[:5]:
                title = article["title"]
                description = article["description"]
                news = f"{title}\n{description}\n"
                speak_and_append(news)
        else:
            speak_and_append("No articles found")
    else:
        speak_and_append("Failed to fetch news data")

def open_application(app_name):
    app_mapping = {
        'notepad': 'notepad.exe',
        'calculator': 'calc.exe',
        'chrome': 'C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe',
        'word': 'C:\\Program Files\\Microsoft Office\\root\\Office16\\WINWORD.EXE',
        'excel': 'C:\\Program Files\\Microsoft Office\\root\\Office16\\EXCEL.EXE',
        'powerpoint': 'C:\\Program Files\\Microsoft Office\\root\\Office16\\POWERPNT.EXE',
        'paint': 'mspaint.exe',
        'file explorer': 'explorer.exe',
        'task manager': 'taskmgr.exe',
        'photos': 'C:\\Program Files\\Windows Photo Viewer\\PhotoViewer.dll',
        'calendar': 'C:\\Program Files\\Windows Calendar\\wincal.exe',
        'media player': 'C:\\Program Files\\Windows Media Player\\wmplayer.exe',
        'edge': 'C:\\Program Files (x86)\\Microsoft\\Edge\\Application\\msedge.exe',
```



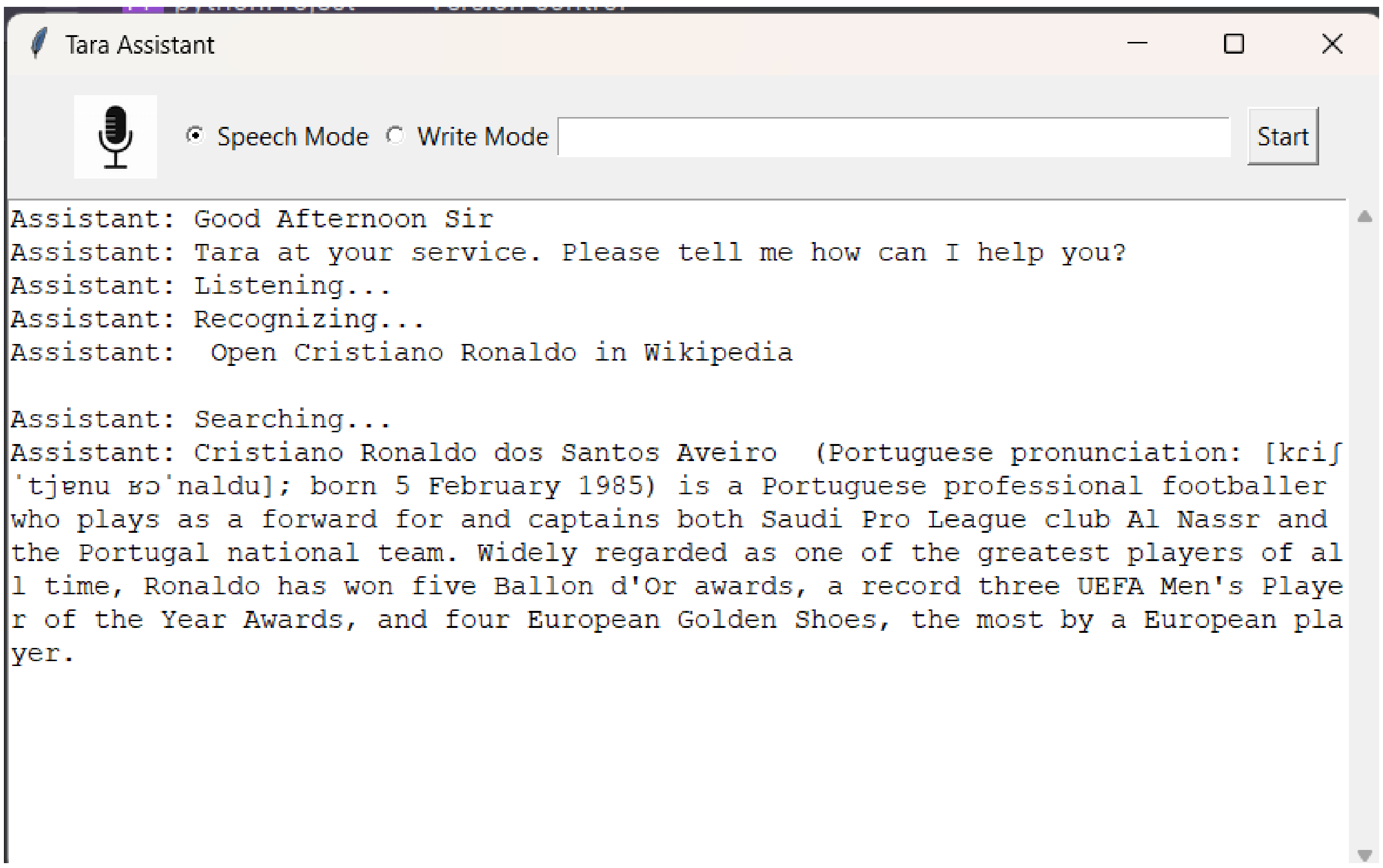
```
        'firefox': 'C:\\Program Files\\Mozilla Firefox\\firefox.exe',
        'vlc': 'C:\\Program Files\\VideoLAN\\VLC\\vlc.exe',
    }
    try:
        if app_name in app_mapping:
            subprocess.Popen(app_mapping[app_name])
            speak_and_append(f"Opening {app_name}")
        else:
            speak_and_append(f"Application {app_name} not found in predefined list.")
    except Exception as e:
        speak_and_append(f"Failed to open {app_name}: {e}")


def process_query(query):
    if 'time' in query:
        time()
    elif 'wikipedia' in query:
        speak_and_append("Searching...")
        query = query.replace("wikipedia", "")
        results = wikipedia.summary(query, sentences=2)
        append_output(results)
        speak(results)
    elif 'date' in query:
        date()
    elif 'offline' in query:
        quit()
    elif 'open in chrome' in query:
        speak_and_append("What should I open?")
        chrome = 'C:/Program Files/Google/Chrome/Application/chrome.exe %s'
        search = takeCommand().lower()
        wb.get(chrome).open_new_tab(search + '.com')
    elif 'search in chrome' in query:
        speak_and_append("What should I search?")
        search = takeCommand().lower()
        wb.get('windows-default').open_new_tab(f"https://www.google.com/search?q={search}")
    elif 'logout' in query:
        os.system("shutdown -l")
    elif 'shutdown' in query:
        os.system("shutdown /s /t 1")
    elif 'restart' in query:
        os.system("shutdown /r /t 1")
    elif 'play songs' in query:
        song_dir = "C:\\Users\\LENOVO\\Music"
        songs = os.listdir(song_dir)
        os.startfile(os.path.join(song_dir, songs[0]))
        speak_and_append("Playing Songs")
    elif 'remember' in query:
        speak_and_append("What should I remember?")
        data = takeCommand().lower()
        speak_and_append("You told me to remember " + data)
        file_path = filedialog.asksaveasfilename(defaultextension=".txt",
                                                 filetypes=[("Text files", "*.txt"), ("All
files", "*.*")])
        if file_path:
            with open(file_path, "w") as remember_file:
                remember_file.write(data)
    elif 'do you know' in query:
        file_path = filedialog.askopenfilename(filetypes=[("Text files", "*.txt"), ("All
files", "*.*")])
        if file_path:
            with open(file_path, "r") as remember_file:
                remember_data = remember_file.read()
                speak_and_append("You told me to remember " + remember_data)
    elif 'screenshot' in query:
        screenshot()
```



```python
    elif 'cpu' in query:
        cpu()
    elif 'joke' in query:
        joke()
    elif 'weather' in query:
        speak_and_append("Please tell me the city name")
        city = takeCommand().lower()
        weather(city)
    elif 'news' in query:
        get_news()
    elif 'open application' in query:
        speak_and_append("Which application should I open?")
        app_name = takeCommand().lower()
        open_application(app_name)

def start_thread(mode):
    thread = threading.Thread(target=mode)
    thread.start()

def on_enter(event=None):
    mode = mode_var.get()
    if mode == "speech":
        start_thread(speak_process_query)
    elif mode == "text":
        start_thread(write_query)

C = 0

def speak_process_query():
    global C
    if C < 1:
        wishme()
    C += 1
    while True:
        query = takeCommand().lower()
        process_query(query)

def write_query():
    global C
    global open_app_flag
    if C < 1:
        wishme()
    C += 1
    remember_flag = False
    while True:
        query = query_entry.get().strip().lower()
        if query:
            if remember_flag:
                data = query
                if data:
                    speak_and_append("You told me to remember " + data)
                    file_path = filedialog.asksaveasfilename(defaultextension=".txt",
                                                              filetypes=[("Text files",
"*.txt"), ("All files", "*.*")])
                    if file_path:
                        with open(file_path, "w") as remember_file:
                            remember_file.write(data)
                    remember_flag = False
                query_entry.delete(0, tk.END)
            elif 'remember' in query:
                remember_flag = True
                speak_and_append("What should I remember?")
                query_entry.delete(0, tk.END)
            elif 'open application' in query:
                open_app_flag = True
                speak_and_append("Which application should I open?")
                query_entry.delete(0, tk.END)
                time.sleep(3)
            elif open_app_flag:
                app_to_open = query
                open_application(app_to_open)
                open_app_flag = False
                query_entry.delete(0, tk.END)


        else:
            process_query(query)
            query_entry.delete(0, tk.END)
        root.update()
def reset_app():
    global remember_flag, open_app_flag
    remember_flag = False
    open_app_flag = False
    query_entry.delete(0, tk.END)
def append_output(text):
    text_area.insert(tk.END, "Assistant: " + text + "\n")
    text_area.see(tk.END)

if __name__ == "__main__":
    root = tk.Tk()
    root.title("Tara AI Assistant")
    frame = tk.Frame(root)
    frame.pack(pady=10)
    microphone_image = Image.open("D:\\Mic.png")
    microphone_image = microphone_image.resize((50, 50), Image.LANCZOS)
    microphone_icon = ImageTk.PhotoImage(microphone_image)
    microphone_label = tk.Label(frame, image=microphone_icon)
    microphone_label.pack(side=tk.LEFT, padx=10)
    mode_var = tk.StringVar(value="speech")
    speech_mode_button = tk.Radiobutton(frame, text="Speech Mode", variable=mode_var,
value="speech")
    speech_mode_button.pack(side=tk.LEFT)
    write_mode_button = tk.Radiobutton(frame, text="Write Mode", variable=mode_var,
value="text")
    write_mode_button.pack(side=tk.LEFT)
    query_entry = tk.Entry(frame, width=50)
    query_entry.pack(side=tk.LEFT)
    button = tk.Button(frame, text="Start", command=on_enter)
    button.pack(side=tk.RIGHT, padx=10)
    root.bind("<Return>", on_enter)
    text_area = scrolledtext.ScrolledText(root, height=20, width=80)
    text_area.pack()
    root.mainloop()
```

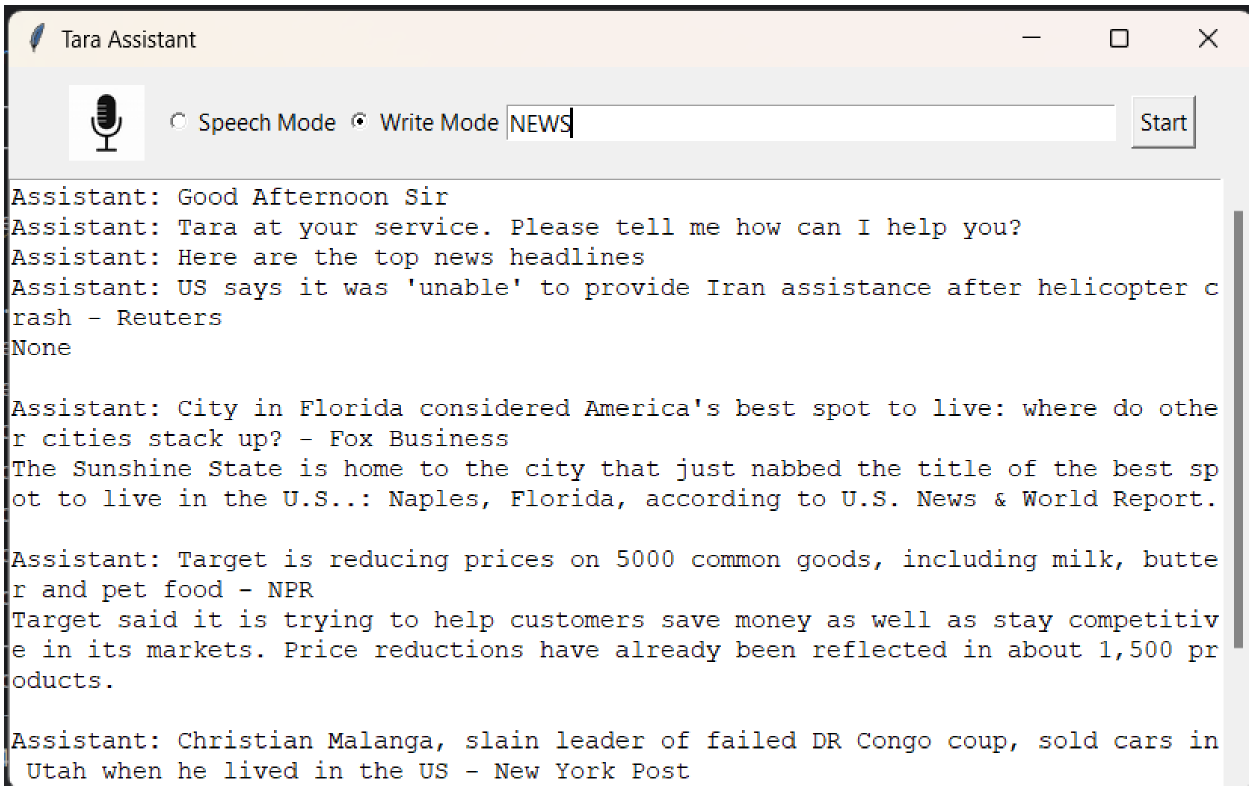
OUTPUT(SCREEN SHOTS):

1.SPEECH MODE:



```
Assistant: Good Afternoon Sir
Assistant: Tara at your service. Please tell me how can I help you?
Assistant: Listening...
Assistant: Recognizing...
Assistant:  Open Cristiano Ronaldo in Wikipedia

Assistant: Searching...
Assistant: Cristiano Ronaldo dos Santos Aveiro  (Portuguese pronunciation: [kɾiʃ
ˈtjɐnu ʁɔˈnaldu]; born 5 February 1985) is a Portuguese professional footballer
who plays as a forward for and captains both Saudi Pro League club Al Nassr and
the Portugal national team. Widely regarded as one of the greatest players of al
l time, Ronaldo has won five Ballon d'Or awards, a record three UEFA Men's Playe
r of the Year Awards, and four European Golden Shoes, the most by a European pla
yer.
```

2.WRITE MODE:



```
Assistant: Good Afternoon Sir
Assistant: Tara at your service. Please tell me how can I help you?
Assistant: Here are the top news headlines
Assistant: US says it was 'unable' to provide Iran assistance after helicopter c
rash - Reuters
None

Assistant: City in Florida considered America's best spot to live: where do othe
r cities stack up? - Fox Business
The Sunshine State is home to the city that just nabbed the title of the best sp
ot to live in the U.S..: Naples, Florida, according to U.S. News & World Report.

Assistant: Target is reducing prices on 5000 common goods, including milk, butte
r and pet food - NPR
Target said it is trying to help customers save money as well as stay competitiv
e in its markets. Price reductions have already been reflected in about 1,500 pr
oducts.

Assistant: Christian Malanga, slain leader of failed DR Congo coup, sold cars in
 Utah when he lived in the US - New York Post
```

CONCLUSION AND FUTURE SCOPE:

AI_ASSISTANT represents a significant advancement in AI-driven communication and task automation. As AI technology continues to evolve, future improvements can include more sophisticated natural language understanding, integration with more diverse data sources, and enhanced user personalization. Potential future developments also encompass advanced machine learning algorithms for better context awareness and predictive capabilities, making AI_ASSISTANT an even more valuable tool in personal and professional settings.