CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Project Name: AI Voice Assistant

AI Voice Assistant is a desktop application that combines voice command functionality with a graphical interface using Tkinter**,** offering users a dual-mode interaction—either by speaking or typing queries. It leverages a range of Python libraries to handle speech, automation, system operations, and online tasks.

Submitted By:

Name: REKHA RANI

UID: 24MCI10261

Class: 24MAM3/B

Submitted To:

Dr: Javed Alam

TABLE OF CONTENT

Acknowledgement

I would like to express my heartfelt gratitude to all those who supported me throughout the development of this project, **“**AI Voice Assistant.”

First and foremost, I would like to thank Dr. Javed Alam, my respected guide, for their valuable guidance, continuous encouragement, and insightful suggestions that helped shape this project from start to finish.

I am also grateful to my institution and faculty members for providing the necessary infrastructure and resources that made the successful completion of this project possible.

Special thanks to my friends and classmates for their motivation, feedback, and moral support throughout the development process.

Last but not least, I would like to thank my family for their unwavering support and patience during the countless hours I spent working on this project.

This project has been a great learning experience, and I am sincerely thankful to everyone who contributed directly or indirectly to its completion.

Aim:

The aim of this project is to design and develop an AI-powered voice assistant integrated with a graphical user interface (GUI) using Python, capable of understanding user commands through voice or text input and performing a wide range of automated tasks to enhance user interaction, convenience, and productivity on a desktop environment.

Objective

The primary objective of this project is to develop a multifunctional AI voice assistant that allows users to interact with their computer system through voice and text commands using a visually appealing and user-friendly graphical user interface (GUI)**.**

1. **Enable Voice Interaction**
   To implement speech recognition and text-to-speech functionalities, allowing the assistant to understand and respond to spoken commands.
2. **Dual Input Mode**
   To allow users to interact with the assistant via both voice input and typed text, making it accessible in noisy environments or for users with different preferences.
3. **Perform Smart Tasks**
   To automate a variety of useful tasks such as:
   - Searching the web and YouTube
   - Fetching summaries from Wikipedia
   - Sending WhatsApp messages
   - Opening common applications (Notepad, Calculator, File Explorer)
   - Taking screenshots
   - Controlling system functions (shutdown, lock, restart)
4. **Build an Interactive GUI**
   To design a Tkinter-based GUI that displays the conversation, accepts typed commands, and allows button-based interaction with the assistant.
5. **Enhance User Convenience**
   To create a helpful desktop assistant that boosts productivity by simplifying routine tasks through automation and voice control.
6. **Learn and Apply Key Concepts**
   To gain hands-on experience in areas such as:
   - Python programming
   - Voice processing
   - GUI development
   - System-level scripting
   - Threading and multitasking

Introduction

In today's rapidly evolving digital age, voice-controlled systems have become an integral part of how humans interact with machines. From smartphones to smart homes, virtual assistants like Siri, Alexa, and Google Assistant have transformed the way users access information and perform tasks. Inspired by these advancements, this project, "**AI Voice Assistant**," aims to build a simplified yet effective voice-activated assistant that can understand user commands, respond intelligently, and perform a variety of tasks.

This voice assistant is designed using Python, one of the most versatile and beginner-friendly programming languages. It combines speech recognition, text-to-speech synthesis, graphical user interface (GUI) capabilities, and task automation to provide a hands-free, interactive experience for users.

What makes this project unique is the integration of both voice and text input, allowing users to interact using whichever method they prefer. The assistant can respond to questions, search the web, open system applications, send WhatsApp messages, take screenshots, and control basic system functions like locking or shutting down the PC.

The project also includes a Tkinter-based GUI, offering a user-friendly interface where users can view the conversation history, enter typed commands, and interact with the assistant visually.

Overall, this project serves as an excellent learning experience in areas such as artificial intelligence basics, GUI programming, API usage, automation, and real-time input/output handling, making it a solid foundation for more advanced AI projects in the future.

Technology Used

This project utilizes a combination of Python libraries, system utilities, and open-source tools to create a fully functional AI voice assistant with a graphical interface. Below are the key technologies used:

Programming Language

- **Python**
  Python was chosen due to its simplicity, readability, and wide support for machine learning, speech processing, and GUI development.

Python Libraries and Modules

1. **SpeechRecognition**
   For capturing and converting voice input into text using the Google Speech API.
2. **pyttsx3**
   A text-to-speech conversion library in Python for generating spoken responses.
3. **Tkinter**
   Python's built-in library used to design the Graphical User Interface (GUI).
4. **Wikipedia API**
   Used to fetch summaries and information based on voice queries.
5. **pywhatkit**
   Allows the assistant to send WhatsApp messages, play YouTube videos, and perform Google searches.
6. **webbrowser**
   Used to open websites like Google, YouTube, etc., directly from voice commands.
7. **datetime**
   For retrieving and formatting the current date and time.
8. **os** and **subprocess**
   Used to interact with the operating system (e.g., open Notepad, Calculator, File Explorer, shutdown, restart, etc.).
9. **ctypes**
   For system-level operations like locking the PC.
10. **pyautogui**
    Used for taking screenshots and automating keypress actions (e.g., mute/unmute).
11. **pyjokes**
    For generating and telling computer-related jokes.
12. **threading**
    To maintain GUI responsiveness during voice command processing.

Implementation

```
import speech_recognition as sr
import pyttsx3
import datetime
import wikipedia
import webbrowser
import pyjokes
import pywhatkit
import tkinter as tk
from tkinter import simpledialog, scrolledtext
import threading
import os
import subprocess
import ctypes
import pyautogui

# Initialize the speech engine
engine = pyttsx3.init()
engine.setProperty('rate', 150)

# Speak function with lock
speak_lock = threading.Lock()

def speak(text):
    chat_log.insert(tk.END, f"Assistant: {text}\n")
    chat_log.see(tk.END)
    with speak_lock:
        engine.say(text)
        engine.runAndWait()

# Greet the user
def greet():
    hour = datetime.datetime.now().hour
    if 0 <= hour < 12:
        speak("Good morning!")
    elif 12 <= hour < 18:
        speak("Good afternoon!")
    else:
        speak("Good evening!")
    speak("I am your assistant. How can I help you today?")

# Take voice input
def take_command():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        speak("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)
```

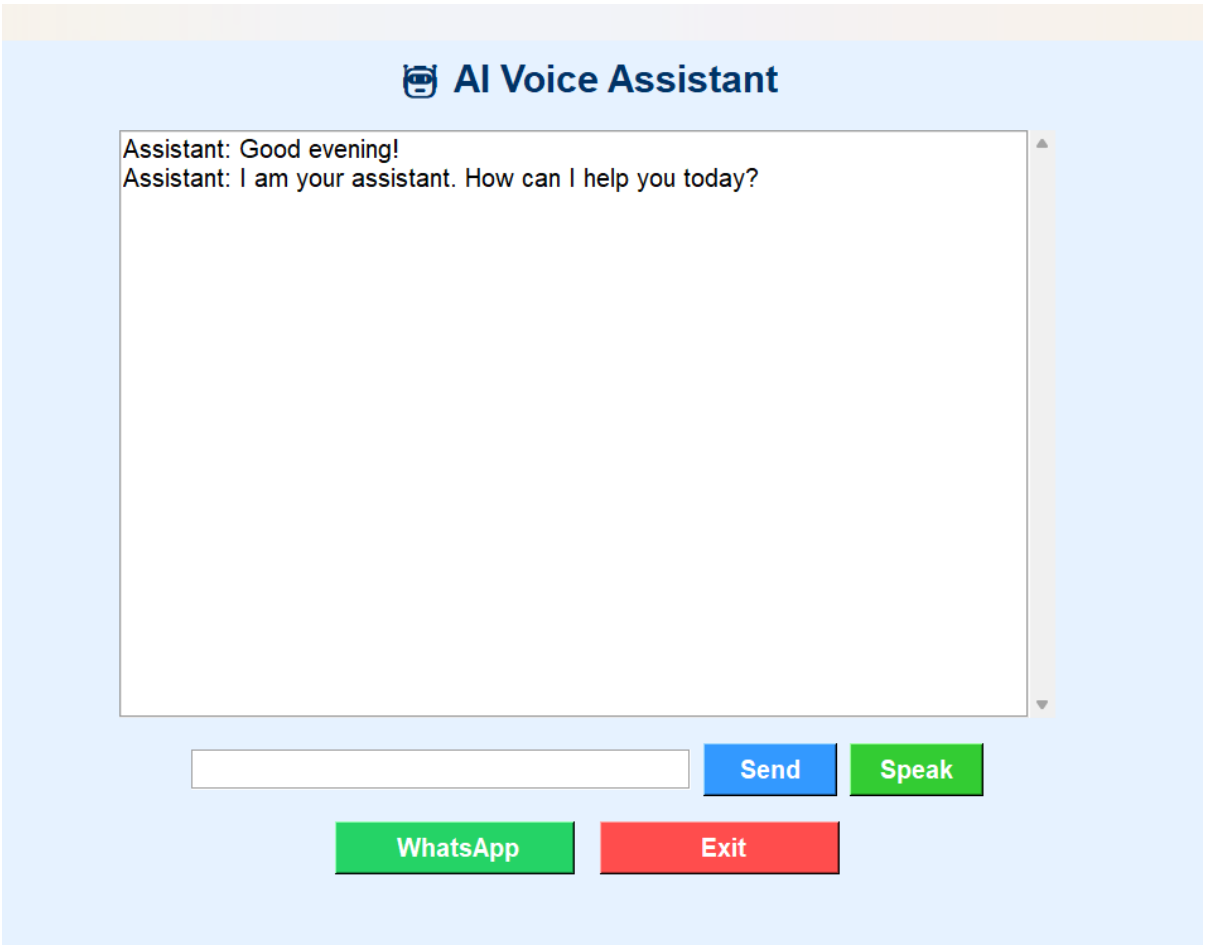
```
 try:
        query = r.recognize_google(audio, language="en-in")
        chat_log.insert(tk.END, f"You (voice): {query}\n")
        chat_log.see(tk.END)
    except Exception:
        speak("Sorry, I didn't catch that.")
        return None
    return query.lower()

# Send WhatsApp message via GUI
def send_whatsapp_gui():
    number = simpledialog.askstring("Phone Number", "Enter number with country code (e.g.,
+911234567890):")
    message = simpledialog.askstring("Message", "Enter the message:")
    if number and message:
        try:
            speak(f"Sending WhatsApp message to {number}")
            pywhatkit.sendwhatmsg_instantly(number, message, wait_time=10)
        except Exception as e:
            speak("Something went wrong.")
            print(e)
    else:
        speak("Number or message not entered.")

# Process user command
def process_query(query):
    if query is None:
        return

    if "time" in query:
        strTime = datetime.datetime.now().strftime("%H:%M")
        speak(f"The time is {strTime}")

    elif "date" in query:
        today = datetime.datetime.now().strftime("%A, %d %B %Y")
        speak(f"Today is {today}")

    elif "wikipedia" in query:
        speak("Searching Wikipedia...")
        query = query.replace("wikipedia", "")
        try:
            results = wikipedia.summary(query, sentences=2)
            speak("According to Wikipedia:")
            speak(results)
        except:
            speak("Couldn't find anything on Wikipedia.")

    elif "open youtube" in query:
        webbrowser.open("https://youtube.com")

    elif "open google" in query:
        webbrowser.open("https://google.com")
```

```
elif "search" in query:
        query = query.replace("search", "").strip()
        speak(f"Searching Google for {query}")
        webbrowser.open(f"https://www.google.com/search?q={query}")

    elif "youtube search" in query or "play" in query:
        query = query.replace("youtube search", "").replace("play", "").strip()
        speak(f"Searching YouTube for {query}")
        webbrowser.open(f"https://www.youtube.com/results?search_query={query}")

    elif "joke" in query:
        joke = pyjokes.get_joke()
        speak(joke)

    elif "how are you" in query:
        speak("I'm great! Ready to help you.")

    elif "who are you" in query:
        speak("I am your personal assistant, created with Python.")

    elif "version control" in query:
        speak("My version control is managed by you.")

    elif "send whatsapp message" in query:
        send_whatsapp_gui()

    elif "open file explorer" in query:
        speak("Opening File Explorer")
        subprocess.Popen("explorer")

    elif "open notepad" in query:
        speak("Opening Notepad")
        subprocess.Popen("notepad.exe")

    elif "open calculator" in query:
        speak("Opening  Calculator")
        subprocess.Popen("calc.exe")

    elif "lock the pc" in query or "lock computer" in query:
        speak("Locking the computer")
        ctypes.windll.user32.LockWorkStation()

    elif "shut down" in query or "shutdown" in query:
        speak("Shutting down the computer")
        os.system("shutdown /s /t 1")

    elif "restart" in query:
        speak("Restarting the computer")
        os.system("shutdown /r /t 1")

    elif "sleep" in query:
        speak("Putting the computer to sleep")
        os.system("rundll32.exe powrprof.dll,SetSuspendState 0,1,0")
```

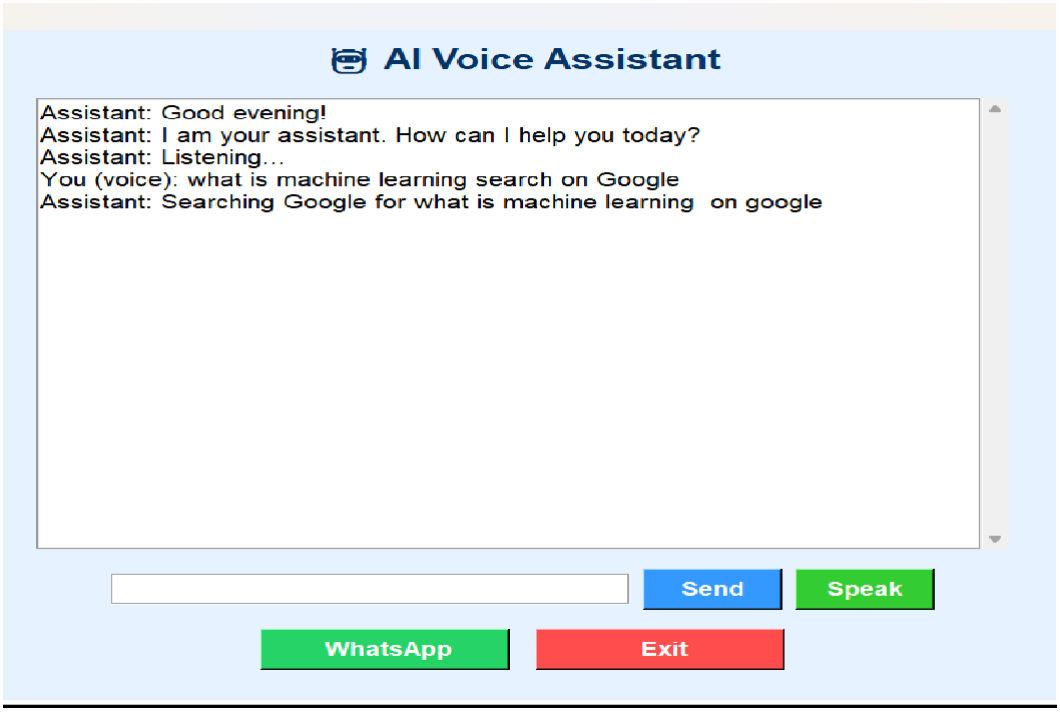
```
elif "take screenshot" in query:
        speak("Taking screenshot")
        screenshot = pyautogui.screenshot()
        path = "screenshot.png"
        screenshot.save(path)
        speak(f"Screenshot saved as {path}")

    elif "mute" in query:
        speak("Muting volume")
        pyautogui.press("volumemute")

    elif "unmute" in query:
        speak("Unmuting volume")
        pyautogui.press("volumemute")

    elif "help" in query:
        speak("Here are some things you can ask me: search Google, play music on YouTube,
open Notepad, lock PC, take screenshot, or tell a joke.")

    elif "stop" in query or "exit" in query or "bye" in query:
        speak("Goodbye! Have a great day.")
        app.destroy()

    else:
        speak("Sorry, I didn't understand that.")

# Voice command in background thread
def handle_voice_command():
    threading.Thread(target=lambda: process_query(take_command()), daemon=True).start()

# Text command handler
def handle_text_command():
    query = text_entry.get()
    if query:
        chat_log.insert(tk.END, f"You (typed): {query}\n")
        chat_log.see(tk.END)
        text_entry.delete(0, tk.END)
        process_query(query.lower())


# --- GUI Setup ---
app = tk.Tk()
app.title("My Personal AI Voice Assistant")
app.geometry("550x600")
app.configure(bg="#e6f2ff")


# Title Label
title_label = tk.Label(app, text=" AI Voice Assistant", font=("Helvetica", 18, "bold"),
bg="#e6f2ff", fg="#003366")
title_label.pack(pady=10)
```

```
# Chat log
chat_log = scrolledtext.ScrolledText(app, wrap=tk.WORD, font=("Arial", 12), height=20,
width=65, bg="white", bd=2, relief=tk.GROOVE)
chat_log.pack(padx=10, pady=10)

# Input frame
frame = tk.Frame(app, bg="#e6f2ff")
frame.pack(pady=10)

text_entry = tk.Entry(frame, font=("Arial", 14), width=30, bd=2, relief=tk.GROOVE)
text_entry.grid(row=0, column=0, padx=5)

send_btn = tk.Button(frame, text="Send", command=handle_text_command, font=("Arial",
12, "bold"), bg="#3399ff", fg="white", width=8)
send_btn.grid(row=0, column=1, padx=5)

voice_btn = tk.Button(frame, text="Speak", command=handle_voice_command,
font=("Arial", 12, "bold"), bg="#33cc33", fg="white", width=8)
voice_btn.grid(row=0, column=2, padx=5)

# Action Buttons Frame
action_frame = tk.Frame(app, bg="#e6f2ff")
action_frame.pack(pady=10)

whatsapp_btn = tk.Button(action_frame, text="📨 WhatsApp",
command=send_whatsapp_gui, font=("Arial", 12, "bold"), bg="#25D366", fg="white",
width=15)
whatsapp_btn.grid(row=0, column=0, padx=10)

exit_btn = tk.Button(action_frame, text=" Exit", command=app.destroy, font=("Arial", 12,
"bold"), bg="#ff4d4d", fg="white", width=15)
exit_btn.grid(row=0, column=1, padx=10)

# Start assistant
greet()
app.mainloop()
```

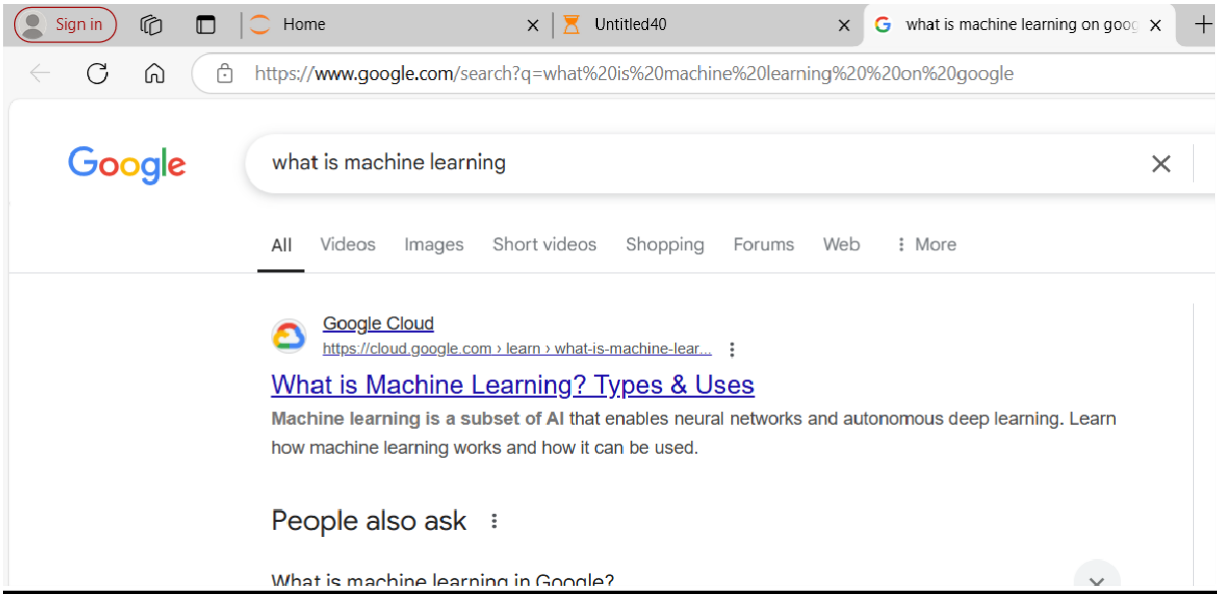
Output


AI Voice Assistant
Assistant: Good evening!
Assistant: I am your assistant. How can I help you today?
Send
Speak
WhatsApp
Exit

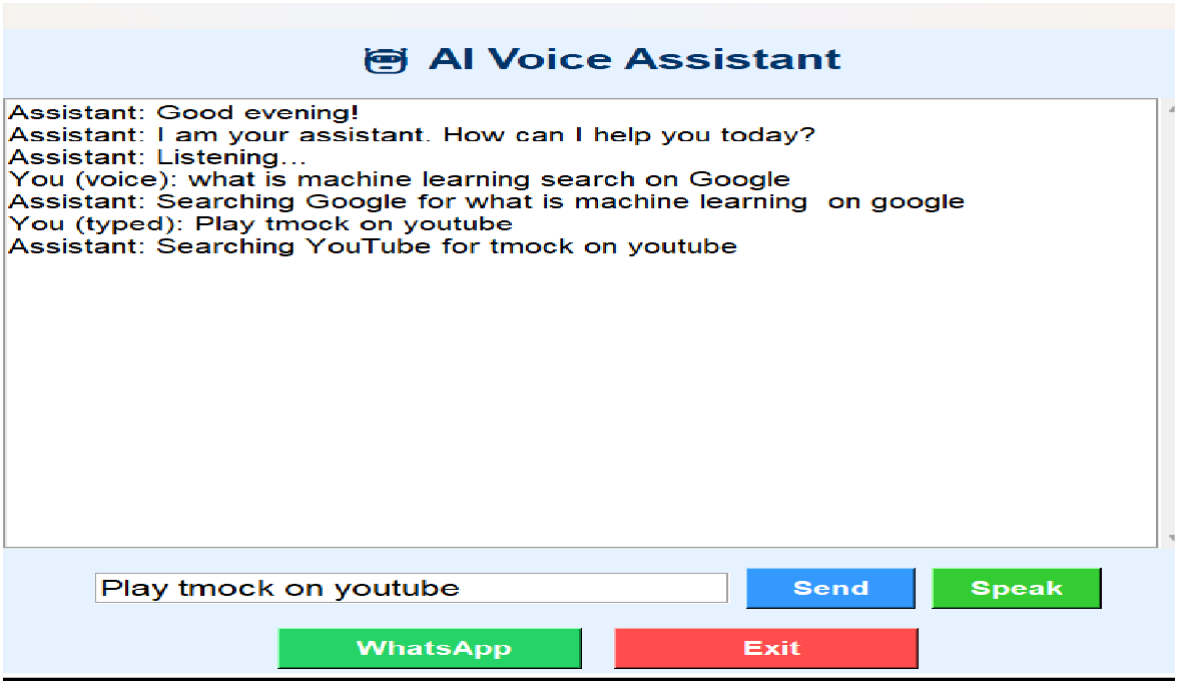
Send command (Speak)



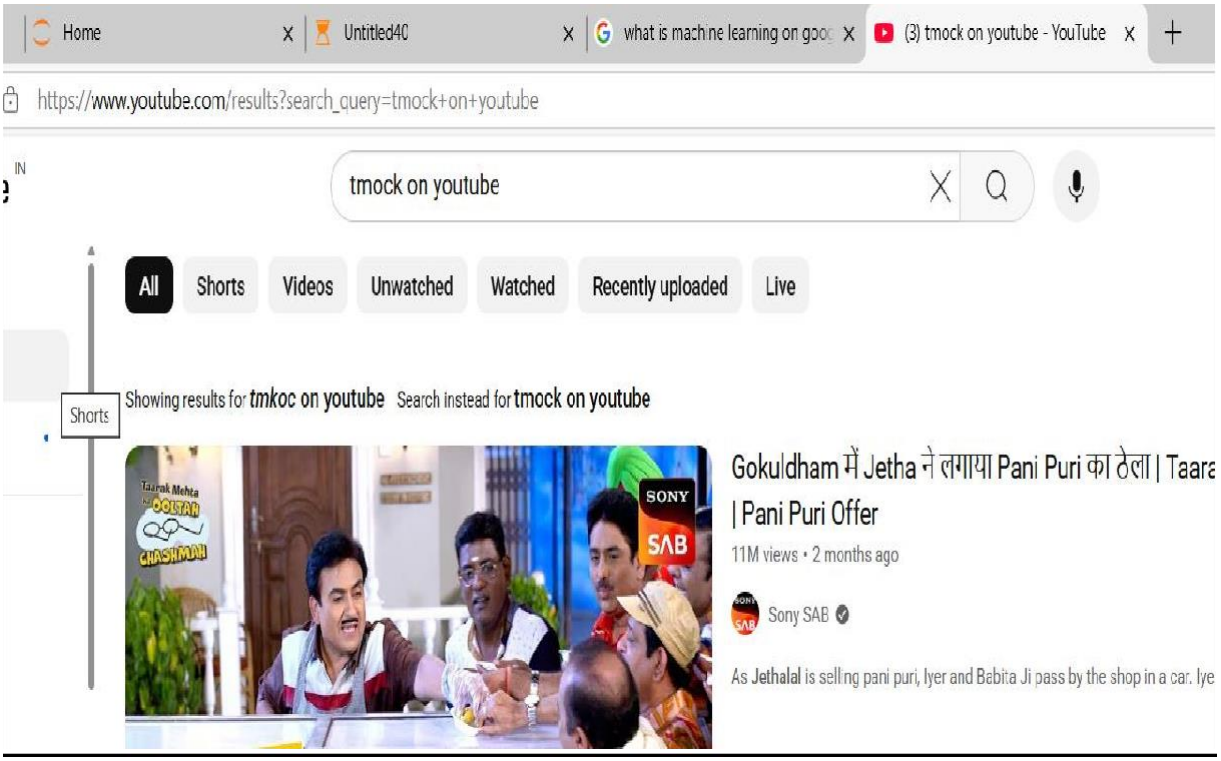
It redirect the google page:



Send command via Text:

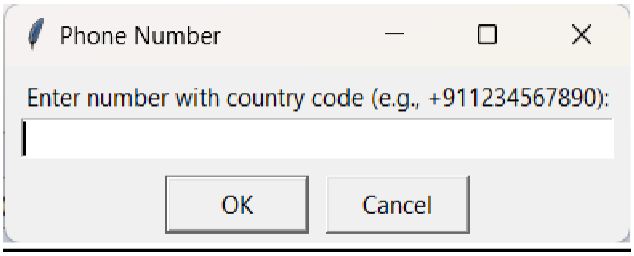


It redirect the Youtube :

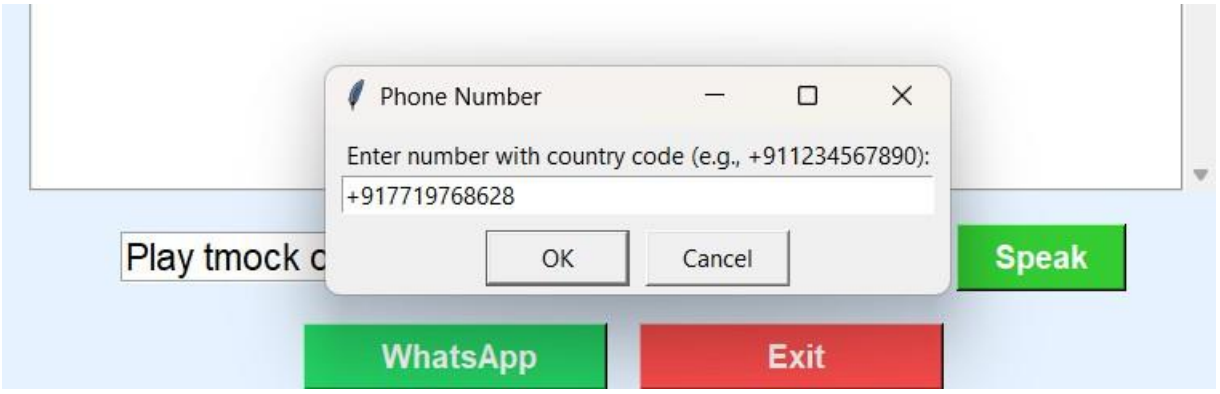


Send Whatsapp Message:

It take number from user with country code:

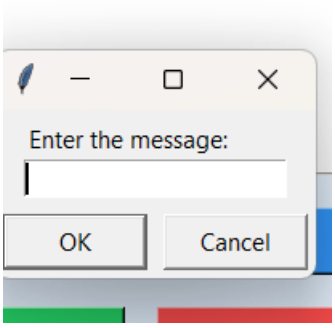


Step1: Enter user number with county code

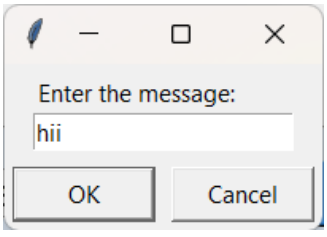


Click on ok.

Step2:
Display a message box

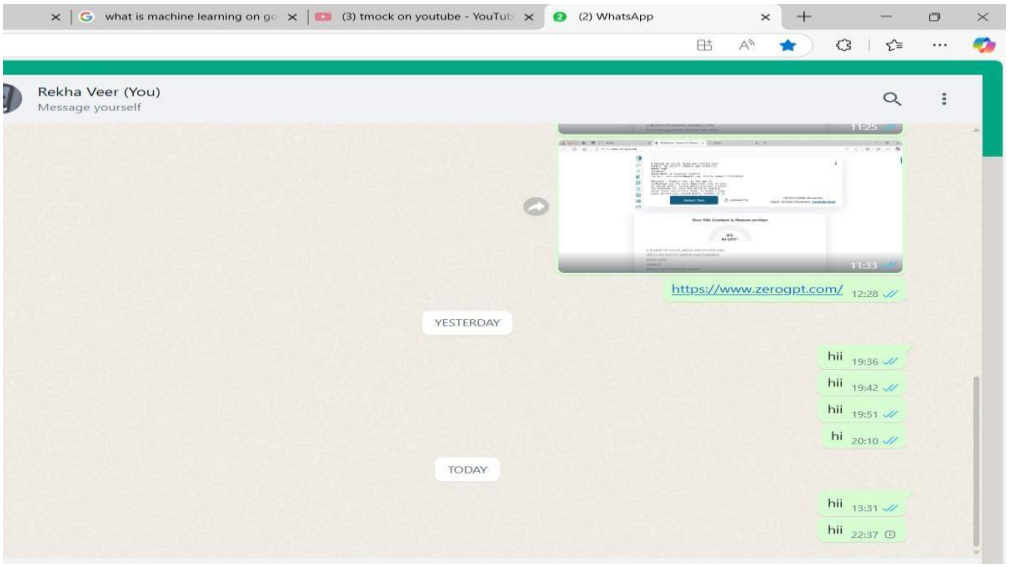


Here we enter message .

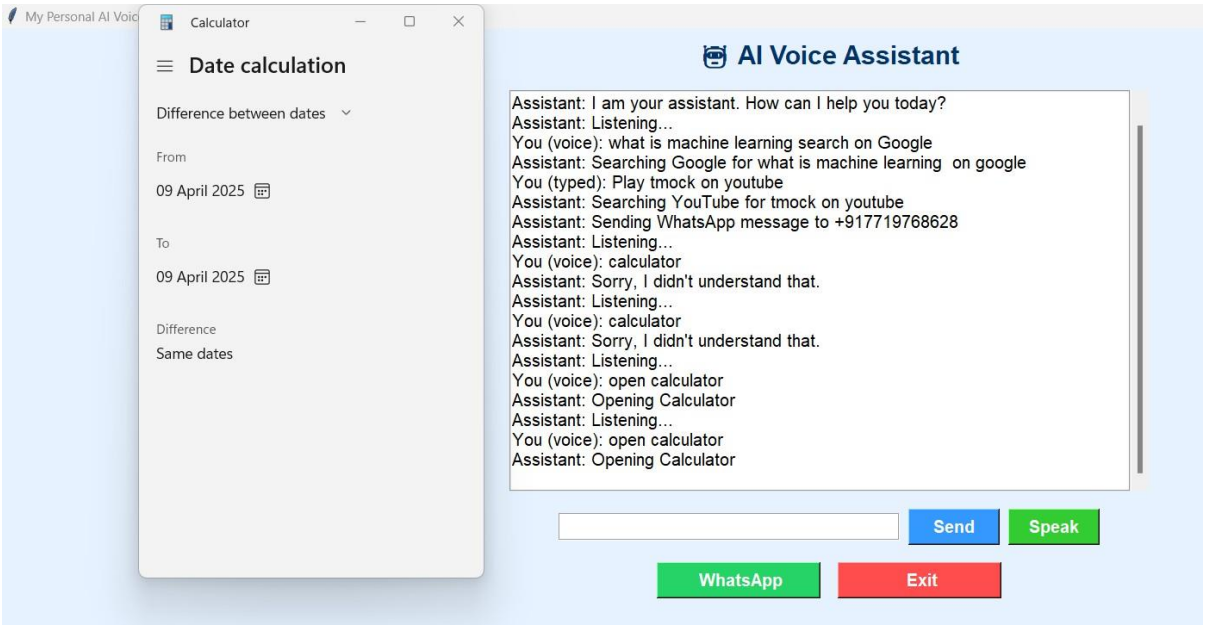


Click on ok.

Step3: what'sapp opened and message should be sent.



Send command like PC controls (open file exp., open calculator , open notepad, restart, shutdown pc Etc)



Conclusion

The development of this project, **"AI Voice Assistant,"** has successfully demonstrated how voice-controlled automation can enhance user interaction with desktop systems. By integrating speech recognition, text-to-speech, GUI design, and system-level control, the assistant is capable of performing a wide range of tasks such as searching the web, opening applications, sending WhatsApp messages, controlling system functions, and more.

Through this project, a user-friendly and efficient personal assistant was created, offering both **voice and text-based interaction** within a graphical interface. The use of Python's diverse libraries made the implementation flexible, modular, and easy to expand in the future.

This project not only strengthened practical knowledge of Python programming, GUI development, and API integration but also provided insights into real-time user interaction, voice processing, and task automation.

In conclusion, the assistant serves as a valuable step toward building more advanced and intelligent virtual assistants, and it lays a strong foundation for further improvements such as adding natural language understanding, voice feedback personalization, and integration with IoT or cloud services.

Github Link

https://github.com/re123-rani/AI-voice-Assistant