**Logo

Description automatically generated**

***Artificial Intelligence Lab***

***Project Report***

### *Course Name:* Artificial Intelligence Lab

### *Course Code:*AL - 2002

### *Instructor:* Sir. Osaid

***Group Members:***

1. Muhammad Taha Majeed Khan (21K-3316)
2. Muhammad Samamah (21K-3205)

***Table of Content:***

**Table of Content** **Page**

**Introduction 3**

**Project Overview 3**

**Technical Implementation 3**

**Feature and Functionalities 10**

**Challenges and Solutions 11**

**Outputs 12**

**Future Enhancements 14**

**Conclusion 14**

***Project Report***

***Virtual Desktop Assistant Based On Voice Recognition***

# ***Introduction:***

Virtual assistants are getting more and more popular as artificial intelligence (AI) and natural language processing (NLP) continue to grow quickly. By using voice commands to aid users with a variety of tasks, these assistants improve the intuitiveness of gadget interactions. The goal of this project is to develop a voice-activated virtual desktop assistant that can respond to user commands.

# ***Project Overview:***

In the "Virtual Desktop Assistant Based on Voice Recognition in Python" project, a desktop program that can comprehend and carry out user commands is being developed. The assistant is capable of doing a number of things, like playing music, opening apps, and retrieving weather data.

# ***Technical Implementations:***

## ***GUI Development:***

The Python GUI library Tkinter is used to generate the graphical user interface (GUI). The UI consists of:

1. A text field where the user and assistant's conversation is shown.
2. Text entry widget by manual input.
3. Buttons to start voice input, end the interaction, and issue orders.

### ***GUI.py:***

from tkinter import \*

from PIL import Image, ImageTk

import action

import spech\_to\_text

def User\_send():

    send = entry1.get()

    bot = action.Action(send)

    text.insert(END, "You --> " + send + "\n")

    if bot is not None:

        text.insert(END, "AI Assistant <-- " + str(bot) + "\n")

    if bot == "ok sir":

        root.destroy()

def ask():

    ask\_val = spech\_to\_text.spech\_to\_text()

    bot\_val = action.Action(ask\_val)

    text.insert(END, "You --> " + ask\_val + "\n")

    if bot\_val is not None:

        text.insert(END, "AI Assistant <-- " + str(bot\_val) + "\n")

    if bot\_val == "ok sir":

        root.destroy()

def delete\_text():

    text.delete("1.0", "end")

root = Tk()

root.geometry("550x675")

root.title("AI Assistant")

root.resizable(False, False)

root.config(bg="#FFFFFF")

Main\_frame = LabelFrame(root, padx=50, pady=10, borderwidth=3, relief="raised", bg="#FFFFFF")

Main\_frame.grid(row=0, column=0, padx=10, pady=10, sticky="nsew")

Text\_label = Label(Main\_frame, text="AI Assistant", font=("Helvetica", 16, "bold"), bg="#FFFFFF")

Text\_label.grid(row=0, column=1, padx=20, pady=10)

Display\_Image = ImageTk.PhotoImage(Image.open("image/images.jpg"))

Image\_Label = Label(Main\_frame, image=Display\_Image, bg="#FFFFFF")

Image\_Label.grid(row=1, column=0, pady=30, padx= 30 , sticky="nsew")

text = Text(root, font=('Arial', 10), bg="#EFEFEF")

text.place(x=100, y=375, width=375, height=175)

entry1 = Entry(root, justify=CENTER)

entry1.place(x=100, y=560, width=350, height=30)

button1 = Button(root, text="Speak", bg="#4CAF50", fg="#FFFFFF", pady=10, padx=20, relief=FLAT, command=ask)

button1.place(x=100, y=610)

button2 = Button(root, text="Send", bg="#4CAF50", fg="#FFFFFF", pady=10, padx=20, relief=FLAT, command=User\_send)

button2.place(x=250, y=610)

button3 = Button(root, text="Clear", bg="#4CAF50", fg="#FFFFFF", pady=10, padx=20, relief=FLAT, command=delete\_text)

button3.place(x=400, y=610)

root.mainloop()

## ***Voice Recognition:***

The Speech Recognition library is used to implement voice recognition. When the assistant hears speech input, it translates to text and is processed.

### ***Speech\_to\_Text.py:***

import speech\_recognition as sr

from requests\_html import HTMLSession

import speak

def spech\_to\_text():

    r =  sr.Recognizer()

    with sr.Microphone() as source:

      audio = r.listen(source)

      voice\_data = ''

      try:

        voice\_data = r.recognize\_google(audio)

        return voice\_data

      except sr.UnknownValueError:

             speak.speak("sorry")

      except sr.RequestError:

            speak.speak('No internet connect please turn on you internet')

## ***Action Handling:***

The action.py script maps recognized phrases to specified actions, so it can handle a variety of commands. Among the actions are starting programs, obtaining meteorological information, carrying out computations, etc.

### ***Action.py:***

import datetime

import speak

import webbrowser

import weather

import os

import calendar

import subprocess

import speech\_recognition as sr

def get\_calendar\_events():

    cal = calendar.Calendar()

    events = cal.monthdays2calendar(datetime.datetime.now().year, datetime.datetime.now().month)

    return events

def open\_application(application\_name):

    try:

        subprocess.Popen(application\_name)

        speak.speak(f"{application\_name} is now open")

    except Exception as e:

        speak.speak(f"Unable to open {application\_name}: {str(e)}")

def calculate\_expression(expression):

    try:

        result = eval(expression)

        speak.speak(f"The result of {expression} is {result}")

        return f"The result of {expression} is {result}"

    except Exception as e:

        speak.speak(f"Unable to calculate the expression: {str(e)}")

        return "Unable to calculate the expression"

def listen\_for\_expression():

    recognizer = sr.Recognizer()

    with sr.Microphone() as source:

        print("Listening for expression...")

        audio = recognizer.listen(source)

    try:

        print("Recognizing expression...")

        expression = recognizer.recognize\_google(audio)

        print(f"User said: {expression}")

        return expression

    except sr.UnknownValueError:

        print("Could not understand audio")

        return None

    except sr.RequestError as e:

        print(f"Error retrieving recognition results from Google Speech Recognition service: {e}")

        return None

def Action(send):

    data\_btn = send.lower()

    if "what is your name?" in data\_btn:

        speak.speak("my name is virtual Assistant")

        return "my name is virtual Assistant"

    elif "hello" in data\_btn or "hi" in data\_btn:

        speak.speak("Hey sir, How i can  help you !")

        return "Hey sir, How i can  help you !"

    elif "how are you" in data\_btn:

        speak.speak("I am doing great these days sir")

        return "I am doing great these days sir"

    elif "thank you" in data\_btn or "thank" in data\_btn:

        speak.speak("its my pleasure sir to stay with you")

        return "its my pleasure sir to stay with you"

    elif "good morning" in data\_btn:

        speak.speak("Good morning sir, i think you might need some help")

        return "Good morning sir, i think you might need some help"

    elif "time now" in data\_btn or "current time" in data\_btn:

        current\_time = datetime.datetime.now()

        Time = (str)(current\_time.hour) + " Hour : ", (str)(current\_time.minute) + " Minute"

        speak.speak(Time)

        return str(Time)

    elif "shutdown" in data\_btn or "quit" in data\_btn:

        speak.speak("ok sir")

        return "ok sir"

    elif "open spotify" in data\_btn or "spotify" in data\_btn:

        webbrowser.open("https://spotify.com/")

        speak.speak("spotify.com is now ready for you, enjoy your music")

        return "spotify.com is now ready for you, enjoy your music"

    elif 'open google' in data\_btn or 'google' in data\_btn:

        open\_application("chrome")

        return "Opening Google Chrome"

    elif 'open facebook' in data\_btn or 'facebook' in data\_btn:

        open\_application("firefox")

        return "Opening Mozilla Firefox"

    elif 'youtube' in data\_btn or "open youtube" in data\_btn:

        open\_application("firefox https://youtube.com/")

        return "Opening YouTube in Microsoft Edge"

    elif 'play songs' in data\_btn or "songs" in data\_btn:

        speak.speak("What song do you want to play?")

        song\_name = listen\_for\_expression()

        if song\_name:

            webbrowser.open(f"https://www.youtube.com/results?search\_query={song\_name.replace(' ', '+')}")

            speak.speak(f"Playing {song\_name}")

            return f"Playing {song\_name}"

        else:

            return "Unable to recognize song name. Please try again."

    elif 'weather' in data\_btn:

        ans = weather.Weather()

        speak.speak(ans)

        return ans

    elif 'music from my laptop' in data\_btn:

        url = 'D:\\Songs'

        songs = os.listdir(url)

        os.startfile(os.path.join(url, songs[0]))

        speak.speak("songs playing...")

        return "songs playing..."

    elif 'calendar events' in data\_btn:

        events = get\_calendar\_events()

        speak.speak("Here are your upcoming events:")

        for week in events:

            for day, event in week:

                if event:

                    speak.speak(f"{day}: {event}")

        return "Here are your upcoming events"

    elif 'calculate' in data\_btn:

        speak.speak("Do you want to enter the expression or speak it?")

        response = listen\_for\_expression()

        if response:

            return calculate\_expression(response)

        else:

            return "Unable to recognize expression. Please try again."

    else:

        speak.speak("I'm unable to understand!")

        return "I'm unable to understand!"

## ***Speech Synthesis:***

The assistant uses the pyttsx3 library to provide spoken responses.

### ***Speak.py:***

import pyttsx3

def speak(text):

    engine = pyttsx3.init()

    rate = engine.getProperty('rate')

    engine.setProperty('rate', rate-70)

    engine.say(text)

    engine.runAndWait()

## ***Weather Information:***

Web scraping is used by the weather module to retrieve weather data.

### ***Weather.py:***

from requests\_html import HTMLSession

def Weather():

    try:

        s = HTMLSession()

        query = "karachi"

        url = f'https://www.google.com/search?q=weather+{query}'

        r = s.get(url, headers={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36'})

        temp\_element = r.html.find('span#wob\_tm', first=True)

        unit\_element = r.html.find('div.vk\_bk.wob-unit span.wob\_t', first=True)

        desc\_element = r.html.find('span#wob\_dc', first=True)

        if temp\_element and unit\_element and desc\_element:

            temp = temp\_element.text

            unit = unit\_element.text

            desc = desc\_element.text

            return f"{temp} {unit} {desc}"

        else:

            return "Weather information not found"

    except Exception as e:

        return f"Error fetching weather information: {str(e)}"

# ***Feature and Functionalities:***

## ***Voice Recognition:***

Uses Google's speech recognition API to translate spoken words into text.

## ***Text Command Processing:***

This system interprets commands and accepts text input.

***Application Control:***

Opens websites and apps that are designated.

## ***Weather Information:***

Offers up-to-date meteorological data.

## ***Music Playback:***

Plays music from locally stored files or YouTube.

## ***Calendar Events:***

This function pulls up and displays calendar events.

## ***Mathematical Calculations:***

Assesses statements in mathematics.

# ***Challenges and Solutions:***

## ***Voice Recognition Accuracy:***

By utilizing Google's extensive voice recognition API, early difficulties in identifying accents and other speech patterns were lessened.

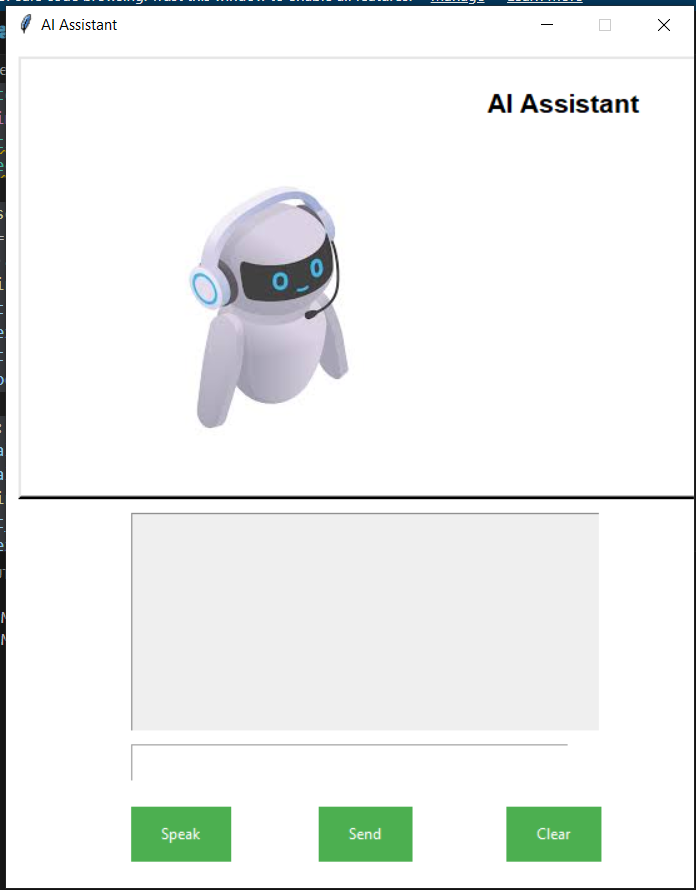
## ***Action Handling:***

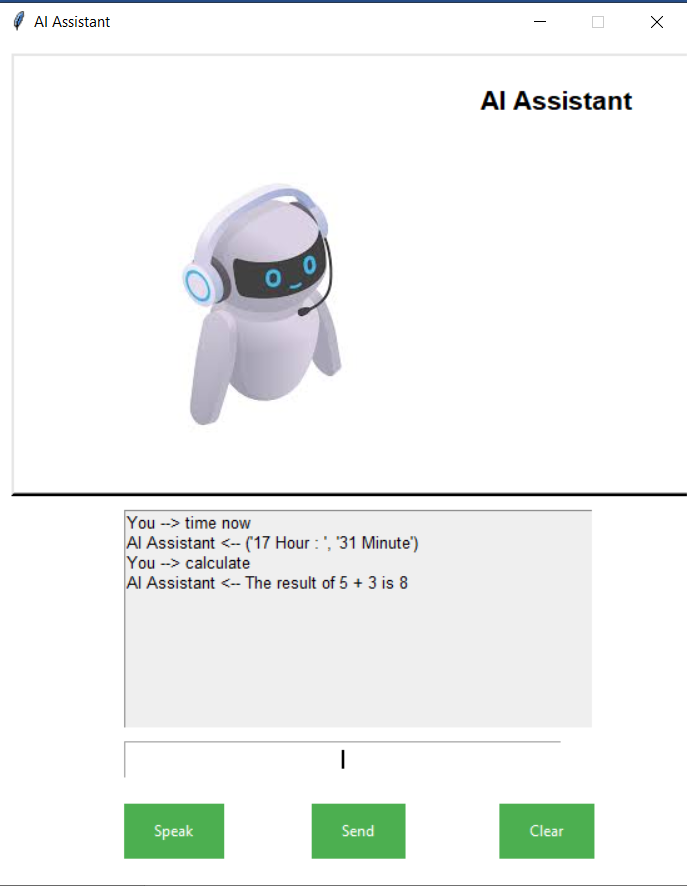
Extensive testing and edge case management were necessary to guarantee that the assistant accurately mapped commands to actions.

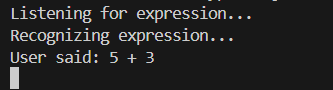
## ***GUI Responsiveness:***

By streamlining the code and employing threading where required, Tkinter's event loop—which occasionally introduced lag—was reduced.

# ***Output:***







# ***Future Enhancements:***

## ***Enhanced NLP:***

Applying cutting-edge NLP methods to comprehend and handle complicated commands more effectively.

## ***Custom Wake phrase:***

Using a unique wake phrase to turn on the helper.

## ***Smart Device Integration:***

Adding more features to manage smart home appliances.

## ***User Profiles:***

Including the ability to create several user profiles, each with their own preferences and settings.

# ***Conclusion:***

The Virtual Desktop Assistant project effectively illustrates how natural language processing and speech recognition can be used to produce an interactive and useful assistant. The project covers a range of functionality from basic searches to more complicated tasks like application control and web scraping. It was constructed using Python and other modules. This assistant can be made even more functional and user-friendly with more improvements.