**Mini Project**

**"Smart Voice Assistant"**

**Name : Shreya N. Pandharikar (22007035)**

**AIM:** To implement a voice-controlled personal assistant named "Sofia" using Python, integrating features like voice recognition, web scraping, email handling, and system commands.

**INTRODUCTION:**

In this practical, we develop a Python-based personal assistant named "Sofia" that interacts with users through speech or text input. Sofia can perform various tasks such as opening applications, searching the web, sending emails, managing notes, and providing information sourced from Wikipedia and Google Maps.

**OBJECTIVES:**

* Develop a voice-controlled personal assistant.
* Implement functionalities for opening applications, searching the web, and sending emails.
* Utilize web scraping to fetch information from Wikipedia and Google Maps.
* Enhance user interaction through speech synthesis and recognition.
* Ensure robustness and reliability of the assistant's functionalities.

**FEATURES:**

* Voice-controlled interaction using speech recognition.
* Integration with web services like Wikipedia and Google Maps for information retrieval.
* Ability to perform system commands such as opening applications and managing notes.
* Customizable input mode (speech or text) based on user preference.
* Modular code structure facilitating easy addition of new features.

**IMPLEMENTATION:**

The implementation involves integrating various Python libraries such as speech\_recognition, pyttsx3, wikipedia, BeautifulSoup, and webbrowser to enable voice recognition, text-to-speech conversion, web scraping, and browser automation functionalities. The program utilizes functions for different tasks like opening applications, searching Wikipedia, sending emails, setting reminders, and more**.**

**FUTURE ENHANCEMENTS:**

* Implement natural language processing for improved user interactions.
* Integrate machine learning algorithms to personalize user experience and enhance task automation.
* Extend functionality to support more third-party services and APIs.
* Enhance security features, such as user authentication and data encryption.
* Improve error handling and feedback mechanisms for better user experience.

**SOURCE CODE:**

import ctypes

import datetime

import os

import subprocess

import time

import webbrowser

from tkinter import \*

import cv2

import pyautogui

import pyjokes

import pyttsx3

import requests

import speech\_recognition as sr

import wikipedia

from bs4 import BeautifulSoup

engine = pyttsx3.init()

voices = engine.getProperty('voices')

engine.setProperty('voice', voices[1].id)

PASSWORD = "chingu" # Set your password here

def speak(audio):

engine.say(audio)

print(audio)

engine.runAndWait()

def takecommand():

r = sr.Recognizer()

mic\_index = 1 # Modify this with the correct device index

with sr.Microphone() as source:

print("Listening...")

r.pause\_threshold = 1

audio = r.listen(source, timeout=5, phrase\_time\_limit=5)

try:

print("Recognizing...")

query = r.recognize\_google(audio)

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

except Exception as e:

print(e)

speak("Say that again please...")

return "None"

return query.lower()

def authenticate():

global root

root = Tk()

root.title("Sofia - Personal Assistant")

root.geometry("500x500")

label = Label(root, text="Welcome to Sofia", font=("Helvetica", 16, "bold"), pady=10)

label.pack()

label\_pass = Label(root, text="Enter Password:", font=("Helvetica", 14))

label\_pass.pack()

password\_entry = Entry(root, show="\*", font=("Helvetica", 14))

password\_entry.pack(pady=5)

def check\_password():

password = password\_entry.get()

if password == PASSWORD:

root.destroy()

else:

speak("Incorrect password. Please try again.")

password\_entry.delete(0, END)

button = Button(root, text="Authenticate", command=check\_password, font=("Helvetica", 14))

button.pack(pady=10)

root.mainloop()

def sendEmail():

sendEmail= "<https://mail.google.com/mail/u/0/#inbox?compose=new>"

os.startfile(sendEmail)

speak("Opening Email")

def wish():

hour = datetime.datetime.now().hour

current\_time = datetime.datetime.now().strftime("%I:%M %p")

if 0 <= hour < 12:

speak("Good Morning")

elif 12 <= hour < 18:

speak("Good Afternoon")

else:

speak("Good Evening")

speak(f"It's {current\_time}. I am Sofia. How can I assist you?")

def opennotepad():

notepad\_path = "C:\\Windows\\System32\\notepad.exe"

os.startfile(notepad\_path)

speak("Opening Notepad")

def run(self):

speak("please say wakeup to continue")

while True:

self.query = self.takecommand()

def opencommandprompt():

os.system("start cmd")

speak("Opening Command Prompt")

def opencamera():

cap = cv2.VideoCapture(0)

while True:

ret, img = cap.read()

cv2.imshow("webcam", img)

k = cv2.waitKey(1)

if k == 27:

break

cap.release()

cv2.destroyAllWindows()

speak("Closing Camera")

def searchwikipedia():

speak("What do you want to search on Wikipedia?")

query = takecommand()

speak("Searching Wikipedia")

try:

results = wikipedia.summary(query, sentences=2)

speak("According to Wikipedia")

speak(results)

except wikipedia.exceptions.DisambiguationError as e:

speak("It seems there are multiple possible pages matching your query. Please specify.")

for option in e.options[:5]: # Limiting to the first 5 options for clarity

speak(option)

speak("Please refine your query.")

except wikipedia.exceptions.PageError:

speak("Sorry, I couldn't find any information related to your query.")

def openchrome():

chrome\_path = "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe"

os.startfile(chrome\_path)

speak("Opening Chrome")

def openspotify():

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

speak("Opening Spotify")

def openyoutube():

webbrowser.open("https://www.youtube.com/")

def opengoogle(query):

webbrowser.open(f"https://www.google.com/search?q={query}")

def switchtabs():

pyautogui.keyDown('alt')

pyautogui.press('tab')

time.sleep(1)

pyautogui.keyUp('alt')

def getcurrenttime():

current\_time = datetime.datetime.now().strftime("%I:%M %p")

speak(f"The current time is {current\_time}")

def taskexecution(input\_mode="speech"):

wish()

while True:

if input\_mode == "speech":

query = takecommand().lower()

else:

query = input("Enter your command: ").lower()

print("User query:", query)

if "hi sofia" in query or "hello sofia" in query:

speak("Hello! How can I assist you today?")

elif "open notepad" in query:

speak("Opening Notepad")

opennotepad()

elif "open command prompt" in query:

opencommandprompt()

elif "open camera" in query:

opencamera()

elif "wikipedia" in query:

searchwikipedia()

elif "open chrome" in query:

openchrome()

elif "open spotify" in query:

openspotify()

elif "open youtube" in query:

openyoutube()

elif "open google" in query:

speak("What should I search for?")

if input\_mode == "speech":

search\_query = takecommand()

else:

search\_query = input("Enter search query: ")

opengoogle(search\_query)

elif "switch tabs" in query:

switchtabs()

elif "current time" in query:

getcurrenttime()

elif "hello" in query or "hey" in query:

speak(" Hello , I'm Sofia")

speak(" How can I help you?")

elif "how are you" in query:

speak(" i am fine,what about you?")

elif "also good" in query or "fine" in query:

speak(" that's good to hear from you")

elif "thank you" in query or "thanks" in query:

speak(" thank you,its my pleasure")

elif "temperature" in query:

search = "temperature in amravati"

url = f"https://www.google.com/search?q=temperature+in+amravati&rlz=1C1ONGR\_enIN1096IN1096&oq=tempreature+in+amaravti&gs\_lcrp=EgZjaHJvbWUqCQgBEAAYDRiABDIGCAAQRRg5MgkIARAAGA0YgAQyCQgCEAAYDRiABDIJCAMQABgNGIAEMgsIBBAAGAoYDRiABDIICAUQABgNGB4yCAgGEAAYDRgeMggIBxAAGA0YHjIICAgQABgNGB4yCAgJEAAYDRge0gEJMTQwNzhqMGo3qAIAsAIA&sourceid=chrome&ie=UTF-8"

r = requests.get(url)

data = BeautifulSoup(r.text, "html.parser")

temp = data.find("div", class\_="BNeawe").text

speak(f"current {search} is {temp}")

elif "restart" in query:

subprocess.call(["shutdown", "/r"])

elif "hibernate" in query or "sleep" in query:

speak("Hibernating")

subprocess.call("shutdown / h")

elif "log off" in query or "sign out" in query:

speak("Make sure all the application are closed before sign-out")

time.sleep(5)

subprocess.call(["shutdown", "/l"])

elif "write a note" in query:

speak("What should i write?")

note = takecommand()

file = open('sophia.txt', 'w')

speak("Should i include date and time")

snfm = takecommand()

if 'yes' in snfm or 'sure' in snfm:

strtime = datetime.datetime.now().strftime("%H:%M:%S")

file.write(strtime)

file.write(" :- ")

file.write(note)

else:

file.write(note)

elif "show note" in query:

speak("Showing Notes")

file = open("sophia.txt", "r")

print(file.read())

speak(file.read(6))

# elif "set reminder" in query:

# set\_reminder()

elif "bye" in query or "goodbye" in query:

speak("Goodbye!")

break # Exit the while loop

elif "where is" in query:

query = query.replace("where is", "").strip()

location = query

speak("User asked to Locate")

speak(location)

show\_location(location)

elif "don't listen" in query or "stop listening" in query:

speak("for how much time you want to stop sophia from listening commands")

a = int(takecommand())

time.sleep(a)

print(a)

elif "change background" in query:

ctypes.windll.user32.SystemParametersInfoW(20, 0, "Location of wallpaper", 0)

speak("Background changed successfully")

elif "joke" in query:

speak(pyjokes.get\_joke())

elif "who made you" in query or "who created you" in query:

speak("I have been created by SHREYA-PRANJALI.")

elif "send a mail" in query:

try:

speak("What should I say?")

content = takecommand()

speak("whom should i send?")

to = input()

sendEmail(to, content)

speak("Email has been sent !")

except Exception as e:

print(e)

speak("I am not able to send this email")

elif "search places in google maps" in query:

speak("Sure, please specify the place you want to search for.")

place\_query = takecommand()

search\_google\_maps(place\_query)

elif "set reminder" in query:

set\_reminder()

def show\_location(location):

search\_query = f"{location} map"

url = f"https://www.google.com/maps/search/{search\_query}"

webbrowser.open(url)

def search\_google\_maps(query):

search\_query = query.replace(" ", "+")

url = f"https://www.google.com/maps/search/{search\_query}"

webbrowser.open(url)

def set\_reminder():

speak("What would you like me to remind you about?")

reminder\_text = takecommand()

speak("When should I remind you?")

speak("Please specify the time in the format HH:MM (24-hour format)")

reminder\_time = input("Enter the time in HH:MM format: ")

current\_time = datetime.datetime.now()

reminder\_hour, reminder\_minute = map(int, reminder\_time.split(":"))

# Calculate the time difference for the reminder

reminder\_datetime = current\_time.replace(hour=reminder\_hour, minute=reminder\_minute, second=0, microsecond=0)

time\_diff = (reminder\_datetime - current\_time).total\_seconds()

# Wait until it's time for the reminder

if time\_diff > 0:

speak(f"Reminder set for {reminder\_time}. I will remind you about '{reminder\_text}' then.")

time.sleep(time\_diff)

speak(f"Reminder: {reminder\_text}")

else:

speak("Invalid time. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

authenticate()

speak(" Hi ,Do you want to interact via speech or text? Say 'speech' or 'text'.")

mode = takecommand().lower() # Get the input mode from the user

while mode not in ['speech', 'text']:

speak("Please say 'speech' or 'text' to select the input mode.")

mode = takecommand().lower()

taskexecution(mode)

**OUTPUT :**



**CONCLUSION:**

In conclusion, the development of the "Smart Voice Assistant Project" represents a significant step forward in harnessing the power of voice recognition and synthesis technologies to create a more intuitive and efficient human-computer interaction experience. Through the integration of various Python libraries and functionalities, this project showcases the potential for voice-controlled assistants to streamline tasks, access information, and provide assistance in a wide range of applications. With further enhancements and refinements, such as natural language processing and machine learning integration, the Smart Voice Assistant has the potential to revolutionize how users interact with technology, paving the way for more seamless and personalized experiences in the future.