Report On

Voice Assistance

v

Submitted in partial fulfillment of the requirements of the Course project in Semester IV of Second Year Computer Engineering

by

Dakshith Shetty (Roll No. 38)

Kaushal Tare (Roll No. 47)

Supervisor

Prof. Sneha Mhatre

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**



**(2023-24)**

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**

**CERTIFICATE**

This is to certify that the project entitled “Voice Assistance” is a Bonafide work of Dakshith Shetty (Roll No. 38) Kaushal Tare (Roll No. 47) submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Computer Engineering.

**Supervisor**

Prof. Sneha Mhatre

Internal Examiner External Examiner

Dr Megha Trivedi

Head of Department

Dr. H.V. Vankudre

Principal

**Index**

1. Abstract
2. Problem Statement
3. Module Description
4. Block Diagram

5.Code

6.Output

7.Conclusion

8.Reference

9.Code

**Abstract:**

This mini project aims to develop a basic voice assistant in Python utilizing speech recognition and text-to-speech conversion libraries. It will involve implementing speech recognition to understand user commands, natural language processing to extract intents, and task execution to perform corresponding actions or provide responses. By integrating these components, the project will culminate in the creation of a functional voice assistant capable of interacting with users through spoken commands and responses

1

**Problem Statement:**

"Develop a voice assistant program that can perform basic tasks such as answering questions, providing weather updates, setting reminders, and playing music upon voice command. The program should utilize speech recognition and synthesis libraries in Python to understand spoken commands and respond appropriately."

**Module Description:**

To develop a voice assistant capable of understanding and executing user commands spoken in natural language.

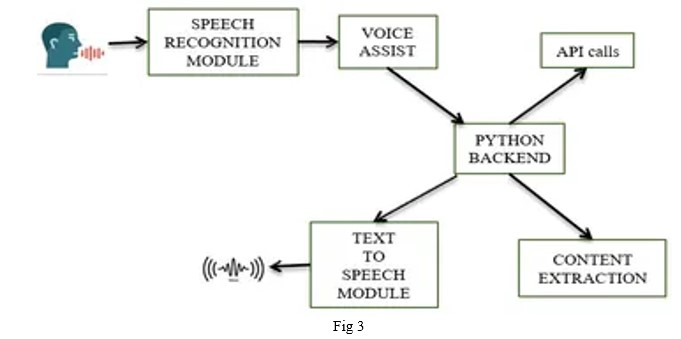
Key Components:

1. Speech Recognition: Implement a module to recognize voice commands using libraries such as SpeechRecognition.
2. Natural Language Understanding: Develop a mechanism to understand user intent through natural language processing.
3. Task Execution: Execute tasks based on recognized commands, such as retrieving information, setting reminders, and providing updates.
4. Text-to-Speech Conversion: Convert assistant responses into spoken words using libraries like pyttsx3 or gTTS.
5. User Interaction: Design an intuitive interface for users to interact with the assistant using voice commands.

Outcome: A functional voice assistant enabling seamless interaction between users and the system through spoken commands and responses.

3

**Block Diagram**



Block Diagram: Voice Assistance

Code:-

import pyttsx3

import datetime

import speech\_recognition as sr

import wikipedia

import os

import webbrowser

import pyjokes

import pywhatkit as kit

import time

from plyer import notification

import tkinter as tk

from tkinter import ttk

from tkinter import LEFT, BOTH, SUNKEN

from PIL import Image, ImageTk

from threading import Thread

# Constants for custom styling

BG\_COLOR = "#D2C6E2"

BUTTON\_COLOR = "#F9F4F2"

BUTTON\_FONT = ("Arial", 14, "bold")

BUTTON\_FOREGROUND = "black"

HEADING\_FONT = ("white", 24, "bold")

INSTRUCTION\_FONT = ("Helvetica", 14)

engine = pyttsx3.init('sapi5')

voices = engine.getProperty('voices')

engine.setProperty('voice', voices[0].id)

def speak(audio):

engine.say(audio)

engine.runAndWait()

entry = None

stop\_flag = False # Define the stop\_flag variable at the top of the script

def wish\_time():

global entry

x = entry.get()

hour = int(datetime.datetime.now().hour)

if 0 <= hour < 6:

speak('Good night! Sleep tight.')

elif 6 <= hour < 12:

speak('Good morning!')

elif 12 <= hour < 18:

speak('Good afternoon!')

else:

speak('Good evening!')

speak(f"{x} How can I help you?")

def take\_command():

recognizer = sr.Recognizer()

with sr.Microphone() as source:

print("Say something:")

speak("say something")

recognizer.adjust\_for\_ambient\_noise(source, duration=1.0) # Adjust for 1 second of ambient noise

recognizer.pause\_threshold = 0.5

audio = recognizer.listen(source)

try:

print("Recognizing...")

speak("recognizing")

query = recognizer.recognize\_google(audio, language='en-in')

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

except Exception as e:

# print(e)

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

return "None"

return query

def perform\_task():

global stop\_flag

while not stop\_flag:

query = take\_command().lower() # Converting user query into lower case

if 'wikipedia' in query:

speak('Searching Wikipedia...')

query = query.replace("wikipedia", "")

try:

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

speak("According to Wikipedia")

print(results)

speak(results)

except wikipedia.exceptions.DisambiguationError as e:

# Handle disambiguation error (when the search term has multiple possible meanings)

print(f"There are multiple meanings for '{query}'. Please be more specific.")

speak(f"There are multiple meanings for '{query}'. Please be more specific.")

except wikipedia.exceptions.PageError as e:

# Handle page not found error (when the search term does not match any Wikipedia page)

print(f"'{query}' does not match any Wikipedia page. Please try again.")

speak(f"'{query}' does not match any Wikipedia page. Please try again.")

elif 'play' in query:

song = query.replace('play', "")

speak("Playing " + song)

kit.playonyt(song)

elif 'open youtube' in query:

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

elif 'open google' in query:

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

elif 'search' in query:

s = query.replace('search', '')

kit.search(s)

elif 'the time' in query:

str\_time = datetime.datetime.now().strftime("%H:%M:%S")

speak(f"Sir, the time is {str\_time}")

elif 'open code' in query:

code\_path = "C:\\Users\\DELL\\AppData\\Local\\Programs\\Microsoft VS Code\\Code.exe"

os.startfile(code\_path)

elif 'joke' in query:

speak(pyjokes.get\_joke())

elif "where is" in query:

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

location = query

speak("User asked to Locate")

speak(location)

webbrowser.open("https://www.google.nl/maps/place/" + location.replace(" ", "+"))

elif 'exit' in query:

speak("thanks for giving your time")

stop\_voice\_assistant()

def stop\_voice\_assistant():

global stop\_flag

speak("Stopping the Voice Assistant.")

stop\_flag = True

def start\_voice\_assistant():

global stop\_flag

wish\_time()

perform\_task()

stop\_flag = False # Reset the flag to False when starting the voice assistant

def main():

# Create the main GUI window

root = tk.Tk()

root.title("Voice Assistant")

root.geometry("500x700")

root.configure(bg=BG\_COLOR)

def on\_button\_click():

global stop\_flag

if not stop\_flag:

stop\_flag = False # Reset the flag to False when starting the voice assistant

Thread(target=start\_voice\_assistant).start()

else:

stop\_voice\_assistant()

# Load and set the background image

background\_image = Image.open(

"wallpaperflare.com\_wallpaper.jpg") # Replace "path/to/your/background\_image.jpg" with the actual image file path

background\_photo = ImageTk.PhotoImage(background\_image)

background\_label = ttk.Label(root, image=background\_photo)

background\_label.place(x=0, y=0, relwidth=1, relheight=1)

f1 = ttk.Frame(root)

f1.pack(pady=100) # Add some padding to the frame to center it vertically

image2 = Image.open("p.jpg") # Replace "path\_to\_image2.jpg" with the actual path to your image

resized\_image = image2.resize((120, 120))

p2 = ImageTk.PhotoImage(resized\_image)

l2 = ttk.Label(f1, image=p2, relief=SUNKEN)

l2.pack(side="top", fill="both")

# Heading

heading\_label = ttk.Label(root, text="Voice Assistant", font=HEADING\_FONT, background=BG\_COLOR)

heading\_label.pack(pady=20)

global entry

f1 = ttk.Frame(root)

f1.pack()

l1 = ttk.Label(f1, text="Enter Your Name", font=INSTRUCTION\_FONT, background=BG\_COLOR)

l1.pack(side=LEFT, fill=BOTH)

entry = ttk.Entry(f1, width=30)

entry.pack(pady=10)

# Instruction

instruction\_label = ttk.Label(root, text="Click the button below to start the Voice Assistant.",

font=INSTRUCTION\_FONT, background=BG\_COLOR)

instruction\_label.pack(pady=10)

# Create and place a button on the GUI

button = ttk.Button(root, text="Start Voice Assistant", command=on\_button\_click,

style="VoiceAssistant.TButton")

button.pack(pady=20)

# Style the button

style = ttk.Style(root)

style.configure("VoiceAssistant.TButton", font=BUTTON\_FONT, background=BUTTON\_COLOR, foreground=BUTTON\_FOREGROUND)

# Run the GUI main loop

root.mainloop()

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

main()

Results:-



**Conclusion:**

In conclusion, the development of the Voice Assistant Mini Project in Python offers a practical demonstration of integrating speech recognition and natural language processing to create an interactive user experience. By leveraging these technologies, we have successfully implemented a voice assistant capable of understanding and executing user commands spoken in natural language. The project underscores the importance of user-centric design, enabling users to interact with the system effortlessly through voice commands. Moving forward, further enhancements could be made to expand the assistant's capabilities and improve its accuracy and responsiveness, thereby advancing the field of voice-enabled applications in Python. Overall, the Voice Assistant Mini Project serves as a valuable learning experience in the intersection of artificial intelligence, human-computer interaction, and software development.

**References:**

Stackoverflow, tutorialpoints , wikepidiea, Openai