International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 06 | June - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

Python Application to Shut-Down PC using Voice Command

Prof. Dr. Thiyagarajan
Assistant Professor
J.Mary Ratana Manjari, Aravind Kanagandula,
D.Aravind Sagar, D. Akhilesh Goud, Y.Arun, A.Aryan Reddy
BACHELOR OF TECHNOLGY IN COMPUTER SCIENCE & ENGINEERING (AI & ML)
Malla Reddy University ,Hyderabad,Telangana,India.

1.INTRODUCTION

Voice recognition based Computer Shut-Down Python program is an application that allows a user to shut down their computer by using their voice. With this program, users can save time and effort by simply giving a voice command to turn off their computer instead of going through multiple steps to do so manually. The program uses speech recognition technology to identify specific voice commands and execute the corresponding action. Voice recognition technology has advanced significantly in recent years, and it is now possible to create accurate and reliable voice recognition programs using Python. This type of program can be very useful in situations where a user's hands are occupied, or they are unable to use the computer's mouse or keyboard due to physical limitations. In this application, the user can customize the voice command to shut down the computer according to their preferences. They can also add other voice commands to perform other actions, such as hibernating or restarting the computer. Overall, the voice recognition based shutdown program is a convenient and practical tool that can enhance the user's computing experience. In this program, the computer's microphone will listen to your voice and convert it into text using sophisticated algorithms. The text is then analyzed to identify the command and execute it. It's a fascinating technology that opens up new possibilities for computer interaction and automation. It's a powerful tool that can save you time and hassle by eliminating the need for manual shutdown.

2.ABSTRACT

ABSTRACT This application is a simple voice-controlled program that allows a user to shut down their computer using voice commands. It creates a voice assistant that can recognize voice commands to shut down a computer. It uses the speech_recognition module to listen to the user's voice input and recognize the command using the Google API. The pyttsx3 module is used to give the assistant the ability to speak.. The program creates a GUI using the Tkinter library, with a "RUN" button that starts the voice assistant when pressed. Once the button is pressed, the program initializes the microphone and waits for the user to say "yes" or "no" to the question of whether they want to shut down their computer. If the user says "no," the program responds with a message indicating that the computer will not be shut down, and the program exits. If the user says "yes," the program responds with a message indicating that the computer is shutting down, and then initiates a 30-second countdown to shutdown. The countdown is displayed on the GUI, and an "Abort" button is displayed that allows the user to cancel the shutdown if desired. The program uses multi-threading to ensure that the GUI remains responsive while the voice assistant is running. When the user says "yes," the program starts a new thread that executes the shutdown sequence, while the main thread continues to update the GUI and respond to user input. The shutdown sequence itself is implemented using the os library, which allows the program to execute a shutdown command on the operating system.

3. LITERATURE REVIEW

The code imports the required modules: **pyttsx3**, **speech_recognition**, and **os**. The **take_commands()** function is defined to capture audio input from the user using the microphone and recognize it using the Google API. If there are any errors during the recognition process, it returns "None". Otherwise, it returns the recognized query. The **Speak()** function is defined to initialize the **pyttsx3** module, convert the input text into speech, and speak it using the system's audio output. The program starts by asking the user if they want to shut down their computer. It uses the **Speak()** function to ask the question. Inside the while loop, it calls the **take_commands()** function to capture the user's response. If the response contains the word "no", it uses the **Speak()** function to acknowledge the response and breaks the loop. If the response contains the word "yes", it uses the **Speak()** function to acknowledge the response, initiates a shutdown command using the **os.system()** function, and breaks the loop. If the response does not match either condition, it asks the user to repeat the command.

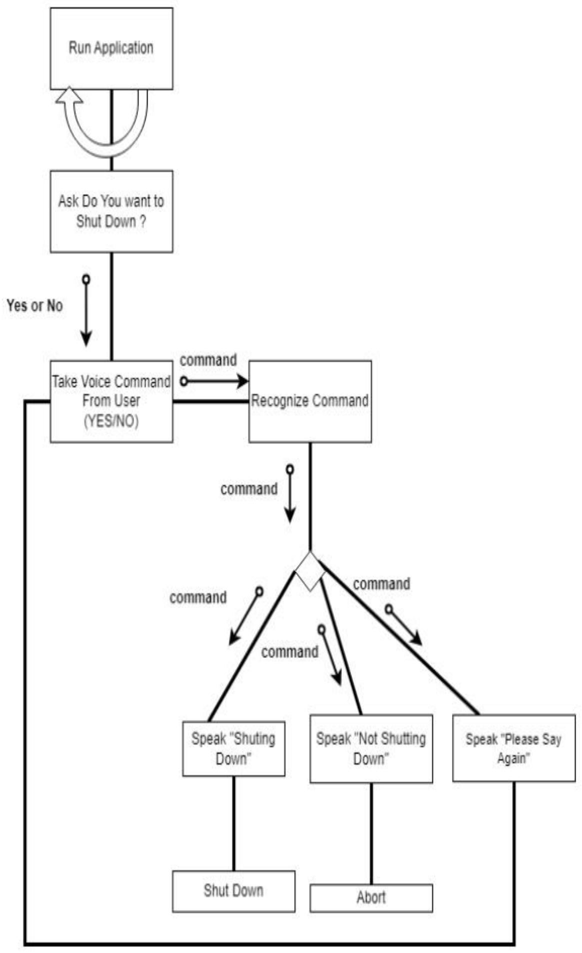
4. PROBLEM STATEMENT

Develop a voice assistant program that can take voice commands from the user, recognize them using the Google API, and perform actions based on the commands. The program should provide the functionality to shut down the computer if the user confirms the shutdown request. The voice assistant should be able to handle different responses from the user, provide appropriate feedback, and execute the shutdown command when requested. The program should ensure a smooth and user-friendly interaction with the voice assistant, handling any errors or exceptions gracefully. Additionally, the code should prioritize security and obtain proper user consent before executing system commands

5.METHODOLOGY

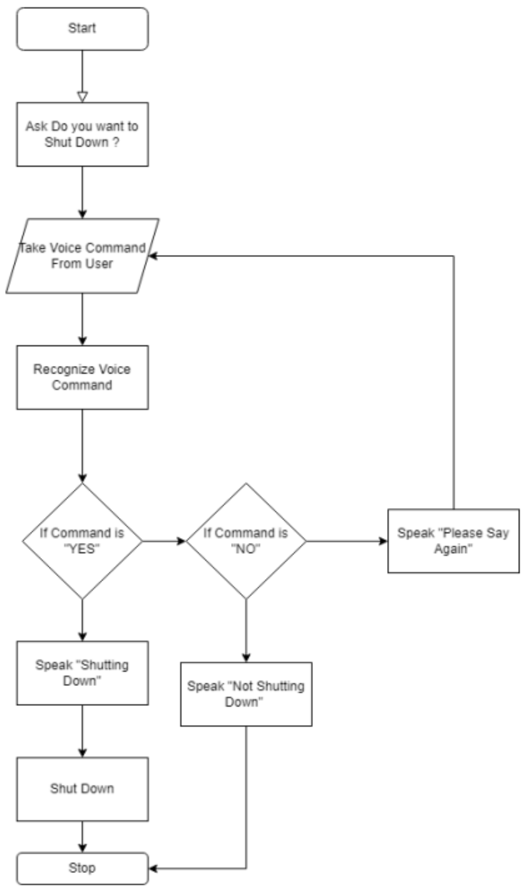
The Simple multi-vendor ecommerce website methodology has included the architecture.

ARCHITECTURE



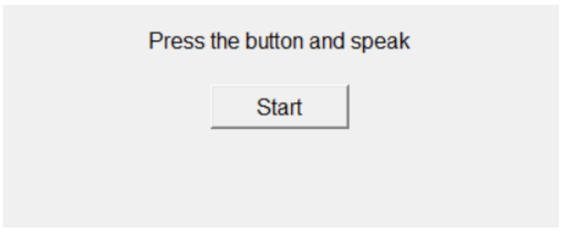
The architecture of the program combines the GUI, voice recognition, text-to-speech, and system action components to create an interactive voice assistant application that provides step-by-step feedback to the user.

DATA FLOW DIAGRAM

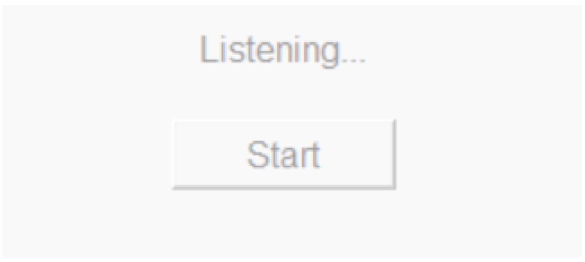


6.EXPERIMENTAL RESULTS

Code uses the Tkinter library to create a basic GUI window. The window includes a label to display instructions and a button that triggers the voice assistant functionality. When the button is clicked, the **take_commands()** function is called to capture the user's voice command.

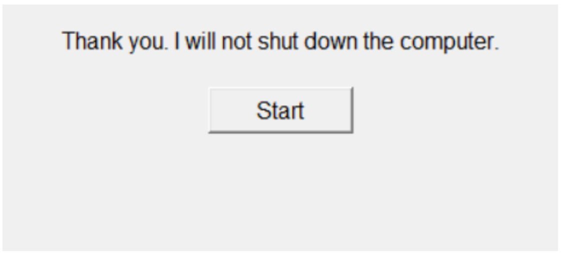

Press the button and speak

Inside the **button_click()** function, before calling **take_commands()**, I added **label.config(text="Listening...")** to display "Listening..." on the label. This informs the user that the microphone is active and ready to listen to their voice command.

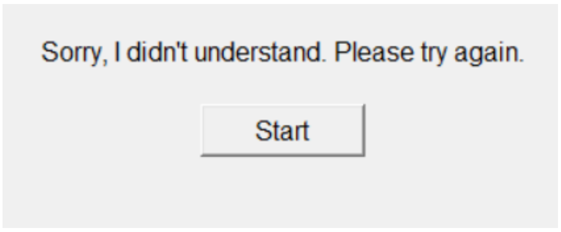

Listening...

After calling **take_commands()**, I updated the label to display "Recognizing..." using **label.config(text="Recognizing...")**. This provides feedback to the user that the query is being recognized.

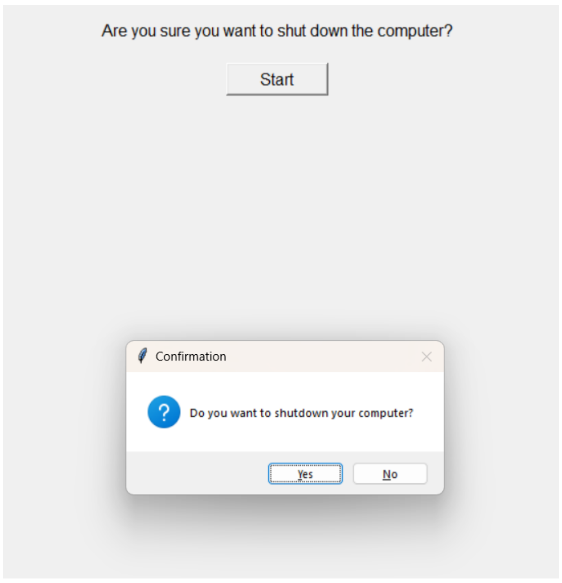
if "no" in command:
It will display ("Thank you. I will not shut down the computer.")


Thank you. I will not shut down the computer.

If it do not understand the query it will display "Sorry, I didn't understand. Please try again"


Sorry, I didn't understand. Please try again.

If "yes" is detected, a confirmation dialog appears using the **messagebox** module from Tkinter, asking the user to confirm the shutdown. If the user confirms, the **shutdown_computer()** function is called to initiate the shutdown process.


Are you sure you want to shut down the computer?
Start
Confirmation
Do you want to shutdown your computer?
Yes
No

On clicking “yes” the system will shutdown

7.CONCLUSION

The program allows users to interact with the voice assistant by speaking voice commands and receiving spoken responses.The code utilizes the **pyttsx3** library for text-to-speech conversion and the **speech_recognition** library for voice recognition using the Google API. The Tkinter library is used to create the GUI window, consisting of a label to display step-by-step feedback and a button to trigger the voice assistant functionality.

The application follows a sequential flow where the user clicks the "Start" button, and the voice assistant listens for voice commands. The program captures the user's voice input, recognizes the command, and responds accordingly. If the recognized command indicates a desire to shut down the computer, a confirmation dialog is displayed, allowing the user to confirm or cancel the shutdown action.

The GUI provides real-time feedback, displaying "Listening..." while the microphone is active and "Recognizing..." during the speech recognition process. Spoken responses from the voice assistant are also displayed on the GUI label, enhancing the user experience and providing clear communication.The code can serve as a starting point for developing more sophisticated voice assistant applications with additional features and functionalities. Further improvements can be made to enhance the user interface, handle more complex voice commands, integrate with other APIs or services, and expand the range of actions the voice assistant can perform.Overall, the code demonstrates the integration of voice recognition, text-to-speech, and GUI components to create an interactive voice assistant application that can understand user commands and respond accordingly, making it a useful and engaging tool for users.

8.FEATURE ENHANCEMENTS

Natural Language Understanding: Incorporate natural language understanding (NLU) capabilities to enable the assistant to understand and respond to more complex queries and user instructions. NLU can help in extracting relevant information from the user's commands and performing appropriate actions based on the context.

Multi-Language Support: Add support for multiple languages to make the assistant accessible to a broader user base. This would involve integrating additional language recognition models and enabling language switching functionality in the GUI.

Contextual Awareness: Enhance the assistant's ability to maintain context across multiple commands or interactions. This can involve storing user preferences, remembering previous interactions, and using that information to provide more personalized and contextually relevant responses.

Text-Based Interaction: Enable the assistant to accept text-based input in addition to voice commands. This would allow users to type their queries instead of speaking, providing flexibility and accommodating users who prefer typing over voice input.

Error Handling and Recovery: Implement robust error handling and recovery mechanisms to gracefully handle errors and unexpected situations. This could include providing error messages or prompts for clarification when the assistant doesn't understand a command or encounters an issue

Voice Command Validation: Implement a validation mechanism to ensure that the voice commands provided by the user are recognized correctly. This can involve checking the command against a predefined set of valid commands or using natural language processing techniques to analyze and understand the user's intent more accurately.

9.REFERENCES

Tkinter Documentation: Tkinter is the standard GUI toolkit for Python at: https://docs.python.org/3/library/tkinter.html

SpeechRecognition Documentation: It provides easy-to-use APIs for working with various speech recognition APIs, including Google, IBM Watson, and more.
Refered from the link https://pypi.org/project/SpeechRecognition/

pyttsx3 Documentation: The documentation provides information on installation, basic usage, advanced features, and customization options. refered from the link https://github.com/nateshmbhat/pyttsx3Google Speech Recognition API:
Refered to the official Google Speech Recognition API documentation for more information on the API usage, available configuration options, and best practices at: https://cloud.google.com/speech-to-text/docs/reference/rest/v1/speech

Python GUI Programming with Tkinter: This ***book by Alan D. Moore*** provides a comprehensive guide to GUI programming with Tkinter.

Python Text-to-Speech Conversion with pyttsx3: The pyttsx3 library has its own documentation that provides detailed information on how to use the library for text-to-speech conversion.Refered by the link: https://pyttsx3.readthedocs.io/