“Aura” – Your Personal Voice Assistant

Overview

This Python-based voice assistant, named "Aura," uses speech recognition and text-to-speech capabilities to interact with users. It also provides a graphical user interface (GUI) for enhanced user interaction. Aura can execute various tasks, such as opening applications, searching online, toggling light/dark modes, and more.

Key Features

1. **Speech Recognition**:
    - Converts spoken commands to text using the ***speech_recognition*** library.
2. **Text-to-Speech**:
    - Provides audible feedback using the ***pyttsx3*** library.
3. **Graphical Interface**:
    - Designed using ***tkinter*** for a user-friendly experience.
4. **Task Execution**:
    - Performs tasks like opening YouTube, Google, Notepad, and Calculator.
5. **Theme Support**:
    - Toggles between dark and light modes.
6. **Music Playback**:
    - Plays random .mp3 files from a specified folder.

Code Explanation

1. Libraries Used

- **tkinter**: For creating the GUI.
- **speech_recognition**: For converting spoken input into text.
- **pyttsx3**: For text-to-speech functionalities.
- **datetime**: To fetch and display the current time.

- **webbrowser**: For opening web pages.
- **os**: For executing system-level commands and accessing files.
- **random**: For selecting random files (e.g., music playback).

---

2. Components

Speech Recognition

- Function: ***listen()***
- Uses ***speech_recognition.Recognizer*** and ***sr.Microphone*** to capture and process audio.
- Returns the command in lowercase for easier processing.

Text-to-Speech

- Function: speak(text)
- Converts text into audible speech using pyttsx3.

Command Execution

- Function: execute_command(command)
- Matches recognized commands to predefined tasks:
    - **Basic Greetings**: Responds to "hello."
    - **Time**: Announces the current time.
    - **Web Browsing**: Opens YouTube, Google, or searches for user queries.
    - **System Apps**: Opens Notepad or Calculator.
    - **Music**: Plays random .mp3 files from a specified folder.
    - **Themes**: Toggles between dark and light modes.
    - **Exit**: Quits the application.

GUI

- Built using tkinter:
    - **Labels**: Display status updates, commands, and instructions.
    - **Buttons**: Start listening or toggle modes.

- **Text Box**: Displays the recognized commands.

Dark/Light Mode

- Function: toggle_mode(mode=None)
- Adjusts the GUI theme based on user preference.

Installation

Requirements

1. Python 3.7+
2. Libraries:
    - speech_recognition
    - pyttsx3
    - tkinter
    - datetime
    - webbrowser
    - os
    - random

Installation Steps

1. Install Python: Download Python
2. Install required libraries
3. Save the script in a .py file.

Usage

1. **Run the Script**:
    - Execute the script: python voice_assistant.py.
2. **Interacting with Aura**:
    - Click "Start Listening."
    - Speak commands such as:

- "Hello"
- "Time"
- "Open YouTube"
- "Search for [query]"
- "Dark Mode"
- "Exit"

Customization

1. **Music Folder**:
    - Update the ***music_folder*** path to your music directory.

```
music_folder = "C:/music"
```

2. **Voice Settings**:
    - Modify speech rate, volume, or voice:

```
engine.setProperty('rate', 150)  # Speed of speech
engine.setProperty('volume', 1)  # Volume level
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)  # Female voice
```

3. **Add New Commands**:
    - Extend ***execute_command(command)*** with additional functionality:

```
elif 'new command' in command:
    # Your custom logic
    speak("Custom response")
```

Known Limitations

1. **Internet Dependency**:
    - Requires an active internet connection for recognizing commands.
2. **Fixed Music Folder**:
    - Music playback is limited to the specified folder.

Potential Enhancements

1. **Add More Commands**:
    - Expand the assistant's capabilities with additional system and web functionalities.
2. **Natural Language Processing**:
    - Integrate advanced NLP techniques for better command understanding.
3. **Voice Activation**:
    - Enable hands-free operation with a wake word.

Conclusion

Aura is a functional voice assistant designed to enhance user convenience. Its modular architecture allows for easy customization and feature expansion, making it a versatile tool for personal or educational projects.

References:

- GeeksforGeeks. (2022, July 12). *Voice Assistant using python*. GeeksforGeeks. https://www.geeksforgeeks.org/voice-assistant-using-python/

- SayantanI. (n.d.). *GitHub - 01-SayantanI/Assistant: This Python Voice Assistant with GUI uses Tkinter to enable users to interact through voice commands. It performs tasks like Wikipedia searches,google searches, YouTube music playback, website opening, providing a fun and interactive voice-based experience.* GitHub. https://github.com/01-SayantanI/Assistant?tab=readme-ov-file

- JakeEh. (2023, August 13). *Make a Voice Assistant with Python* [Video]. YouTube. https://www.youtube.com/watch?v=iwVaAAEE4fo