Voice Recorder Documentation

==========

This documentation provides a detailed explanation of the Voice Recorder application code.

Table of Contents:

---

1. Introduction:

The Voice Recorder application is a simple GUI-based program built using Tkinter library in Python. It allows the user to record audio for a specified duration and saves the recording as a WAV file.

2. Required Libraries:

The following libraries are required to run the Voice Recorder application:

- Tkinter: Provides the graphical user interface components.

- messagebox from tkinter: Used to display messages and alerts.

- sounddevice: Enables audio recording functionality.

- scipy.io.wavfile: Used to write the recorded audio as a WAV file.

- time: Provides time-related functions.

- wavio: A library for reading and writing WAV files.

- datetime: Used to generate a timestamp for unique filenames.

3. Application Overview:

The Voice Recorder application consists of a graphical user interface (GUI) window built using Tkinter. It includes the following components:

- Application window with a fixed size and title.

- Logo image displayed at the top.

- "Voice Recorder" title displayed in a large font.

- Entry box to input the recording duration in seconds.

- Label to indicate the purpose of the entry box.

- Record button to start the recording.

- Countdown label to show the remaining recording time.

4. Code Explanation:

4.1. Application Setup:

- The necessary libraries are imported at the beginning of the code.

- The Tkinter root window is created with a fixed size of 600x700 pixels, positioned at (400, 80) on the screen.

- The window is configured to be non-resizable and has a title of "Voice Recorder".

- The background color of the window is set to "#4a4a4a".

4.2. Record Function:

- The `Record` function is called when the user clicks on the Record button.

- It retrieves the duration of the recording from the entry box and converts it to an integer value.

- The sounddevice library is used to record audio for the specified duration.

- Inside a try-except block, the function attempts to convert the duration input to an integer. If it fails, a message is printed.

- A while loop is used to implement a countdown timer. The loop updates the GUI window, sleeps for one second, and decreases the timer value by 1.

- When the timer reaches zero, a message box is displayed to notify the user that the recording time is up.

- The sound.wait() function is used to ensure that the recording is completed before proceeding.

- A timestamp is generated using the current date and time, formatted as "%Y%m%d_%H%M%S".

- The recording is saved as a WAV file with a unique filename based on the timestamp using the scipy.io.wavfile.write() function.

4.3. Timer Implementation:

- The timer implementation is done using a while loop and the time.sleep() function.

- Inside the loop, the GUI window is updated using the root.update() function to ensure responsiveness.

- The loop sleeps for one second using the time.sleep(1) function.

- The timer value is decreased by 1 each iteration until it reaches zero.

4.4. Saving the Recording:

- The current date and time are obtained using the datetime.now() function.

- The timestamp is formatted as "%Y%m%d_%H%M%S" to represent the year, month, day, hour, minute, and second.

- A filename is generated by appending the timestamp to the string "recording_" and the extension ".wav".

- The recording is saved using the scipy.io.wavfile.write() function with the generated filename, frequency, and recorded data.

4.5. GUI Design:

- The application includes a logo image displayed at the top using the PhotoImage() function and the Label() widget.

- The "Voice Recorder" title is displayed in a large font using the Label() widget.

- An entry box is provided for the user to enter the recording duration in seconds using the Entry() widget.

- A label is displayed to indicate the purpose of the entry box using the Label() widget.

- The Record button is created using the Button() widget with a custom font, text, background color, and command to call the Record function.

- The countdown label is created using the Label() widget and initially placed at the coordinates (240, 590).

- The application window is displayed using the root.mainloop() function.

5. Conclusion:

The Voice Recorder application provides a simple and user-friendly interface for recording audio. It allows users to specify the recording duration and saves the recordings as separate WAV files with unique filenames based on the current date and time.

Please note that the code provided is for demonstration purposes and may require additional error handling and input validation to ensure robustness and user-friendly behavior