

|  |  |
| --- | --- |
| **Submitted By:** |  |
| **Name:** | **Zunaira Abdul Shakoor** |
| **Roll No.:** | **su92-Bsdsm-f23-034** |
| **Section:** | **3A** |
| **Date:** | **29 November,2024** |
| **Project Name** | **Music Player Assistant** |

### ****Project:****

### ****Music player Assistant****

**It has three files:**

* pymusic\_player.py
* main.py
* test.py.

### ****pymusic\_player.py: Music Player GUI****

This file contains the main music player logic and GUI components.

#### ****Key Components:****

**1.Imports**:

* pygame: A library used to handle audio playback.
* mutagen: Used for reading metadata of MP3 files.
* tkinter: Python’s standard library for GUI.
* ttkthemes: Provides styled themes for Tkinter.

**2.Global Variables**:

* root, listbox, index, etc., are used to manage the state of the application, including the song list, current song index, and volume.

**3.Music Player GUI Initialization**:

* root: The main Tkinter window is created with a theme (radiance), a background color, and window size restrictions.
* listbox: A widget that displays the list of available songs. Users can select a song from here.
* **Volume Control**: The ttk.Scale widget is used to create a volume slider, which controls the music volume.
* **Buttons**: Buttons are added for different functionalities like Play, Pause, Next, Previous, Add Song, and Delete Song.

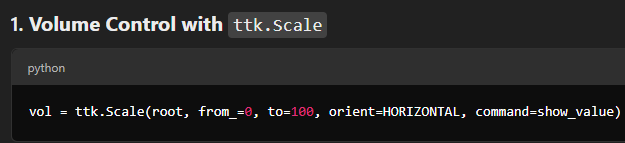
**4.Functions**:

* **show\_value(val)**: Controls the volume based on slider input.
* **on\_closing()**: Handles the quit action for closing the application.
* **play\_songs()**: Loads and plays songs from the directory.
* **pausesong(event)**: Pauses/Unpauses the current song.
* **nextsong(event)**: Plays the next song in the playlist.
* **previoussong(event)**: Plays the previous song in the playlist.
* **loop\_songs()**: Ensures that when a song finishes, the next song starts playing automatically.
* **upcoming\_song()**: Displays the upcoming song in the GUI.
* **del\_music(self)**: Deletes selected songs from the playlist.
* **add\_music(self)**: Adds a new song to the playlist by selecting it from the file system.

1. **Event Bindings**:

* Buttons are linked to the respective functions using the .bind() method.

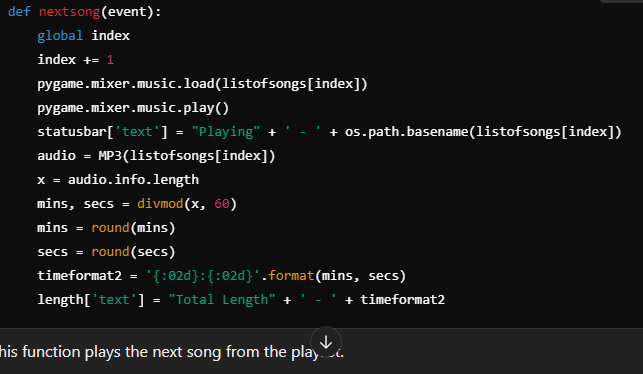
**1. **Volume Control with** ttk.Scale**



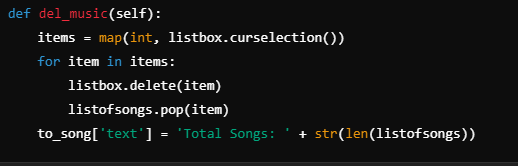
1. **Play, Pause, Next, Previous Button Functions:**



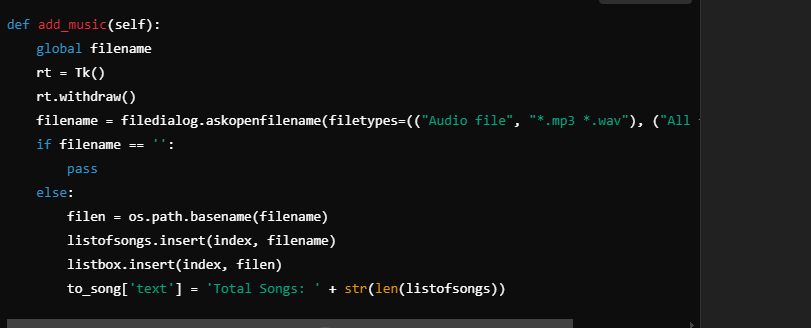
**Play Next Song**



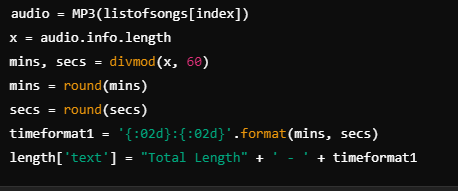
1. ****Dynamic Playlist (Add/Delete Songs):****



**Add Song to Playlist:**



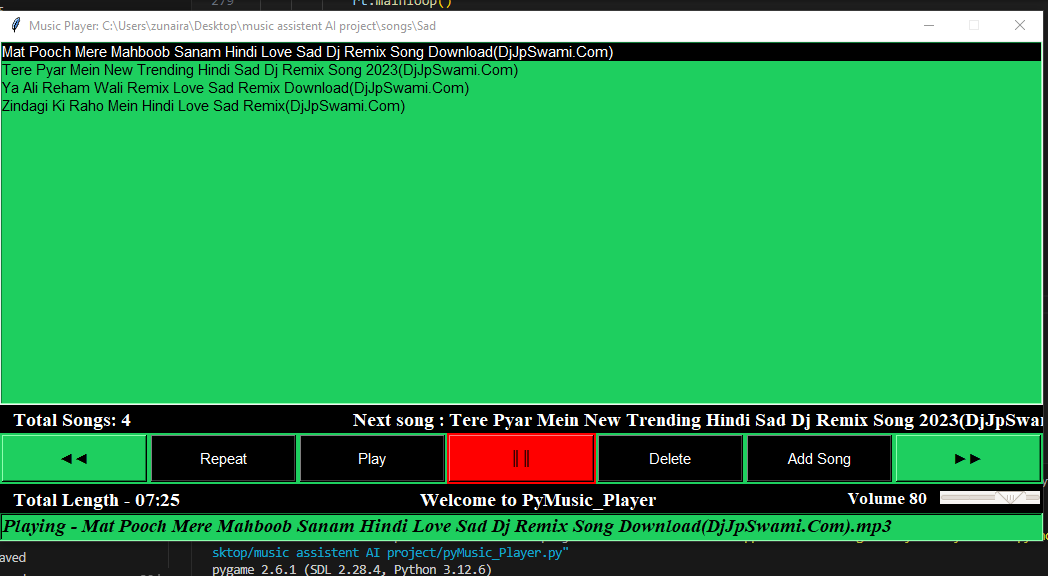
1. **Playback Status Update:**



1. **GUI Elements (Buttons and Labels):**



**GUI DISPLAY:**



### ****main.py: Integration with Voice Commands****

This script integrates **voice commands** to control the music player using the pyttsx3 library for speech synthesis and speech\_recognition for recognizing voice commands.

#### ****Key Components:****

**1.pyttsx3**:

* Used to provide voice feedback. The speak function is responsible for converting text to speech using the sapi5 engine (Microsoft's speech synthesis engine).

**2.speech\_recognition**:

* Used to capture and recognize voice commands from the user.
* recognize\_google() is used to convert speech into text.

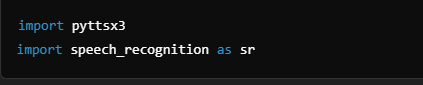
**3.Tkinter:**

* Although Tkinter is imported, it’s not actively used in this code snippet, though it might be useful for the main GUI if extended.

**4.Voice Command Integration**:

* **speak(audio)**: This function takes text input and speaks it out loud.
* **listen()**: Captures the user's voice through the microphone and processes it using Google’s speech recognition API.
* **Commands for controlling music**:
* The user can give voice commands like "play", "pause", "next", "previous", etc., to control the music player.
* The commands are recognized and then executed by calling corresponding functions from pymusic\_player.py.

****6.Speech Recognition Integration (In the Second Code Block):****



### **test.py**

The test.py file is likely meant for testing purposes, ensuring that individual components of the music player work as expected. You might use this to simulate user interactions, test speech commands, or check the player’s behavior with different types of music files.

#### General Use Cases:

* **Unit Testing**: Testing functions like add\_music, del\_music, and the playback functionality.
* **Mocking**: Simulating user input, like selecting a song or changing the volume, to see how the music player responds.

### Possible Enhancements or Features:

1. **Voice Commands**: Integrate voice commands for music control (e.g., "Play", "Pause", "Next song", "Previous song").
2. **Playlist Saving**: Allow users to save their playlist so they can load it later.
3. **Shuffle and Repeat Mode**: Add options for shuffling the songs or repeating a song.
4. **Advanced File Formats**: Support for other file types like .wav or .flac.

### ****How It Works Together:****

**1.Main Flow:**

* main.py acts as the entry point for the voice-based control system.
* Once main.py is executed, it listens for voice commands from the user. For example, the user could say “Play music” to start the music player.
* Based on the voice command, the program triggers appropriate actions in the pymusic\_player.py, like playing music, pausing it, going to the next song, etc.

**2.GUI (pymusic\_player.py)**:

* Once the music player is initialized through pymusic\_player.py, the GUI appears, showing the list of available songs, volume control, and buttons for manual control.
* Voice commands act as an alternative input method to control the player.

**3.Voice and Manual Control**:

* Users can interact with the music player in two ways:
* **Voice commands**: Start playing, pause, skip to the next or previous song, add a song, or delete a song.
* **Manual control**: Clicking buttons in the GUI.

**test.py :**

is used to test the functionalities of pymusic\_player.py to ensure everything is working smoothly

#### Suggestions for Improvement:

#### The main.py file could be expanded to make the speech recognition interact more seamlessly with the GUI.

#### You might want to handle edge cases, such as the absence of any MP3 files in the selected directory, or the case where the song list is empty.