Project Report On:

"Music Player"

Submitted By: Lakshay Joshi

University Roll No.: 220112227

Class Roll No.: 40

Section: F1

Date Of Submission: 18.01.2025

Department Of Computer Science

And Engineering

Graphic Era Hill University, Dehradun

Certificate

This is to certify that Lakshay Joshi, University Roll No.: 2219015 have successfully completed the project demonstrating “Music Player” at Graphic Era Hill University, Dehradun in the fulfilment of requirements of Fifth semester, Bachlor of Technology(Computer Science and Engineering).

ACKNOWLEDGEMENT

I would like to express our gratitude to the almighty Lord Shiva, the most beneficent and the most merciful, for completion of project.

I wish to thank our parents for their continuing support and encouragement. I also wish to thank them for providing us with the opportunity to reach this far in our studies.

I would like to thank particularly our project Co-ordinator _____for his patience, support and encouragement throughout the completion of this project and having faith in us.

At Last, but not the least I greatly indebted to all other persons who directly or indirectly helped me during this work.

Mr. Lakshay Joshi

Roll No:- 2219015

CSE-F1-V Sem

GEHU Dehradun

CONTENTS

INTRODUCTION

This project presents an Enhanced Music Player application developed using Python. The application uses several libraries such as pygame, tkinter, speech_recognition, mutagen, and gTTS to provide an interactive and user-friendly interface for playing and controlling music. In addition to standard controls like play, pause, and stop, the player supports voice commands for hands-free interaction. The player also provides additional features like volume control, and real-time song time tracking.

Purpose

The main objective of this project is to create an intuitive music player that:

1. Allows users to control music playback through both graphical interface and voice commands.
2. Provides real-time feedback such as the song's current position and total length.
3. Supports shuffling and navigation between songs.

The application aims to deliver a seamless, accessible, and interactive experience for users, particularly those who prefer voice-assisted control.

RESOURCES USED:

- **pygame.mixer**: Used for handling music playback, volume control, and song management.
- **tkinter**: Utilized to build the graphical user interface (GUI) for the music player.
- **speech_recognition**: Used to recognize voice commands, allowing users to interact with the player hands-free.
- **mutagen**: Provides MP3 metadata extraction to retrieve song duration and album art.
- **gTTS (Google Text-to-Speech)**: Converts text-based messages (such as song names and status updates) into spoken feedback.

Key Features

1. Music Playback Controls

- **Play**: Starts playing the selected song from the playlist.
- **Pause**: Pauses the currently playing song.
- **Stop**: Stops the playback of the song.
- **Resume**: Resumes playback of the currently paused song.
- **Next**: Skips to the next song in the playlist.
- **Previous**: Goes back to the previous song in the playlist.
- **Shuffle**: Randomly selects a song from the playlist and starts playback.

2. Voice Command Integration

The player supports voice commands for the following actions:

- **Play**: Starts playback of the selected song.
- **Pause**: Pauses the music.
- **Stop**: Stops the music.
- **Next**: Moves to the next song.
- **Previous**: Moves to the previous song.
- **Shuffle**: Shuffles the playlist and plays a random song.

- **Load**: Opens a file dialog to load a directory of songs into the playlist.

These commands are recognized by the speech_recognition library and executed accordingly.

3. Playlist Management

The player allows users to load a directory of MP3 files into the playlist using a file dialog. Songs can be selected from the list for playback.

4. Real-Time Song Time Tracking

The player displays the current playtime and total duration of the song in the format MM:SS / MM:SS. This real-time tracking is updated every second while the song is playing.

1. GUI Design

The graphical user interface (GUI) is developed using tkinter. The GUI consists of:

- **Song Information**: A label displaying the currently playing song's name.
- **Controls**: Buttons for play, pause, stop, next, previous, shuffle, and volume control.
- **Playlist**: A listbox displaying the available songs in the selected directory.
- **Album Art**: A section of the GUI that displays the album art of the current song (if available).
- **Time Display**: A status bar at the bottom displaying the song’s current time and total duration.

2. Voice Command Processing

The speech_recognition library listens for voice commands in real-time. When a command is recognized, the application checks the command's content (e.g., "play", "pause", "next") and invokes the corresponding function to perform the action. The speech recognition runs on a separate thread to ensure the GUI remains responsive.

3. Speech Feedback

The gTTS (Google Text-to-Speech) library is used to generate spoken feedback for actions like "Playing music", "Song paused", and "Now playing [song name]". This feedback is generated in real-time and played using playsound.

4. Song Metadata Extraction

Using the mutagen library, the application extracts metadata from MP3 files, such as song duration and album art. If album art is embedded in the MP3 file, it is displayed in the GUI using the Pillow library for image manipulation.

5. Error Handling

The application handles errors in several places:

- **File Loading**: If there is an error loading or playing a song, an error message is displayed using messagebox.showerror().
- **Time and Album Art Updates**: If there is an issue retrieving song metadata or updating album art, it falls back to default values or empty states.

LIMITATIONS

1. **Voice Command Accuracy**: While the voice recognition feature is functional, its accuracy may be affected by background noise or unclear commands. Using a more advanced speech recognition system or training a custom model could improve performance.
2. **Delay in Speech Feedback**: The time.sleep(2) delay after the "Playing music" announcement could cause a brief pause before playback begins. Reducing this delay or using non-blocking speech synthesis could improve the user experience.
3. **Album Art Handling**: The application expects album art to be embedded in the MP3 file's metadata. If the metadata is missing or corrupted, the album art display may fail. Adding a fallback image (e.g., a default "no album art" icon) could address this.

Future Improvements:

1. **Enhanced Voice Command Processing:**
    - Voice commands could be expanded to support more sophisticated features like volume control by voice, or skipping to a specific track by name.
2. **Better Error Messages:**
    - Error handling can be improved with more informative messages to guide users in resolving issues (e.g., handling missing files or unsupported formats).

3. **User Authentication (Optional):**
    - Implementing user authentication for personalized music playlists and preferences could enhance the user experience.
4. **Cross-Platform Support:**
    - Ensuring better cross-platform compatibility for different operating systems (Windows, macOS, Linux) can be considered.

CONCLUSION

This enhanced music player application effectively combines audio control, voice command integration, and multimedia features to deliver a modern and interactive music listening experience. The use of speech recognition for hands-free control, along with real-time feedback and album art display, makes the application both functional and accessible. Despite a few limitations, the application demonstrates a robust approach to developing an interactive music player with Python.

REFERENCES

- https://www.pygame.org/docs/
- https://www.w3schools.com/python/python_threading.asp
- https://gtts.readthedocs.io/en/latest/
- www.youtube.com
- Mutagen docs
- Python docs
- gemini.google.com