MUSIC PLAYLIST MANAGER

JAVA PROJECT REPORT

SUBMITTED TO

Dr. Mohammad Shifat-E-Rabbi

Assistant Professor

Course: CSE215; Section: 07

SUBMITTED BY

Group No. 07

Team Members:

Shaikh Adiba Nur (2411268642)

Sudipto Sarkar (2321173042)

Noushin Tabasum Katha (2021350642)

Amattullah Shajida Elma (2322157042)

Abstract

The **Music Playlist Manager** is a Java-based application designed to simplify music organization and playback. It allows users to create and manage playlists, add or remove songs, and save playlists for future use. The application features a user-friendly interface, smooth music playback using multithreading, and file input/output for saving data. With its intuitive design and essential functionalities, the Music Playlist Manager offers an efficient solution for managing personal music collections.

Table Of Contents

Introduction

Managing music collections can be time-consuming, especially for users with large libraries. The Music Playlist Manager was developed to address this need by providing an easy-to-use application for organizing and playing music.

The application is built using Java, leveraging technologies like Swing for the graphical user interface and libraries like JLayer for audio playback. It allows users to create playlists, add or remove songs, and save their playlists for future use. Smooth playback is ensured through multithreading, and the integration of file I/O enables users to maintain their playlists across sessions. The project demonstrates the practical application of programming concepts like object-oriented design, multithreading, and file handling, offering a functional and engaging tool for music lovers.

This report outlines the features, implementation, and outcomes of the project, showcasing its successful completion and potential for future enhancements.

Project Overview

The Music Playlist Manager is a Java-based desktop application designed to enable users to manage, organize, and play their music playlists efficiently. The application allows users to create playlists, add and remove songs, and save/load playlists for future sessions. By incorporating multithreading for smooth playback and file I/O for persistent data storage, this application offers a seamless music management experience.

Objectives

- To develop a user-friendly interface for managing music playlists.
- To implement smooth music playback using multithreading.
- To provide functionality for saving and loading playlists with file I/O operations.
- To showcase the ability to handle audio metadata and provide basic music player controls.

Features

1. **Playlist Management**:
    - Users can create new playlists and manage existing ones.
    - Easily add and remove songs from playlists.
2. **Smooth Music Playback**:
    - Music plays smoothly while the user interacts with other features of the application, thanks to multithreading.
    - Supports basic playback controls like play, pause, stop, skip, and back.
3. **File I/O**:
    - Playlists can be saved to files and reloaded, allowing users to continue where they left off.
4. **User-Friendly Interface**:
    - The GUI provides buttons to control playback, displays song metadata (title, artist, duration), and includes a progress slider to track the current song.

Technologies Used

- **Java Swing:** For creating the graphical user interface (GUI).
- **Java Audio Libraries:**
    - **JLayer**: For MP3 audio playback.
    - **mp3agic**: For reading and editing MP3 metadata.
    - **jaudiotagger**: For managing audio file metadata (e.g., title, artist).
- **Multithreading:** Ensures smooth music playback while interacting with the application.
- **File I/O:** For saving and loading playlists to/from files.

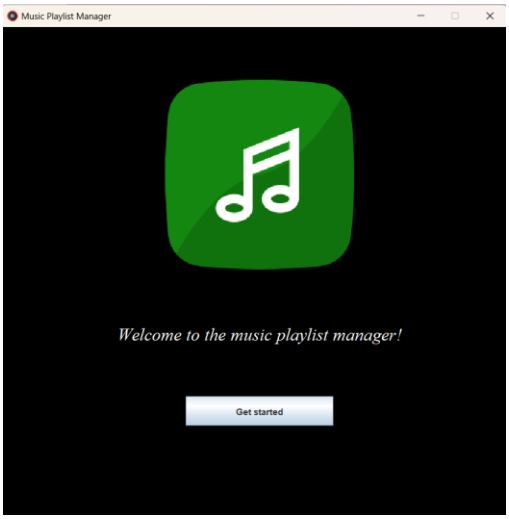
Implementation Details

1. **Song Class**:
    - Represents a single song in the system, holding attributes like title, artist, and file path.
    - Uses external libraries (**mp3agic** and **jaudiotagger**) to read and write metadata from audio files.
    - Fully functional and tested.
2. **MusicPlayer Class**:
    - Controls the playback of music, providing methods to play, pause, skip, and navigate through songs.
    - Implements multithreading to separate the playback functionality from the main GUI, ensuring a smooth user experience.
    - Fully implemented and integrated with other classes.
3. **MusicPlayerGUI Class**:
    - Provides the main graphical interface where users can interact with the application.
    - Includes buttons for controlling playback, a display area for song metadata, and a progress slider for the current song's position.
    - Fully implemented and integrates with the **MusicPlayer** class to provide a seamless user interface.
4. **MusicPlaylistDialog Class**:
    - Allows users to create, edit, and manage playlists.
    - Users can add or remove songs from playlists, and save/load playlists to/from files.
    - Fully implemented, ensuring users can manage their music collections efficiently.
5. **LaunchPage Class**:
    - The entry point to the application, featuring a welcome screen with the application logo and a button to transition to the main interface.
    - Fully implemented and integrated into the application workflow.

Implementation Result:

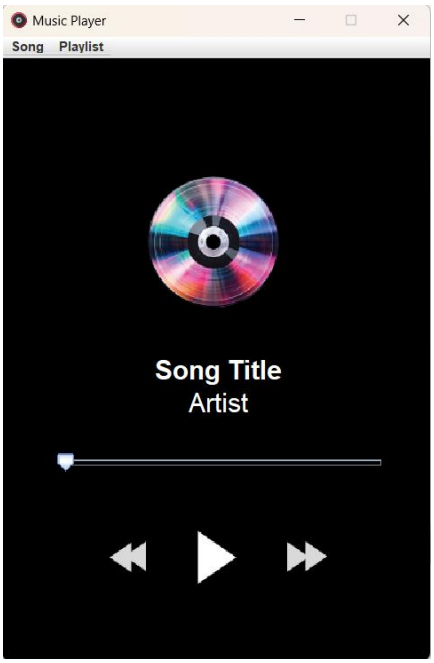
Launch Page:

- Title: *Welcome Screen*
- Description: This is the starting point of the application, featuring a logo and a button to proceed to the main interface.



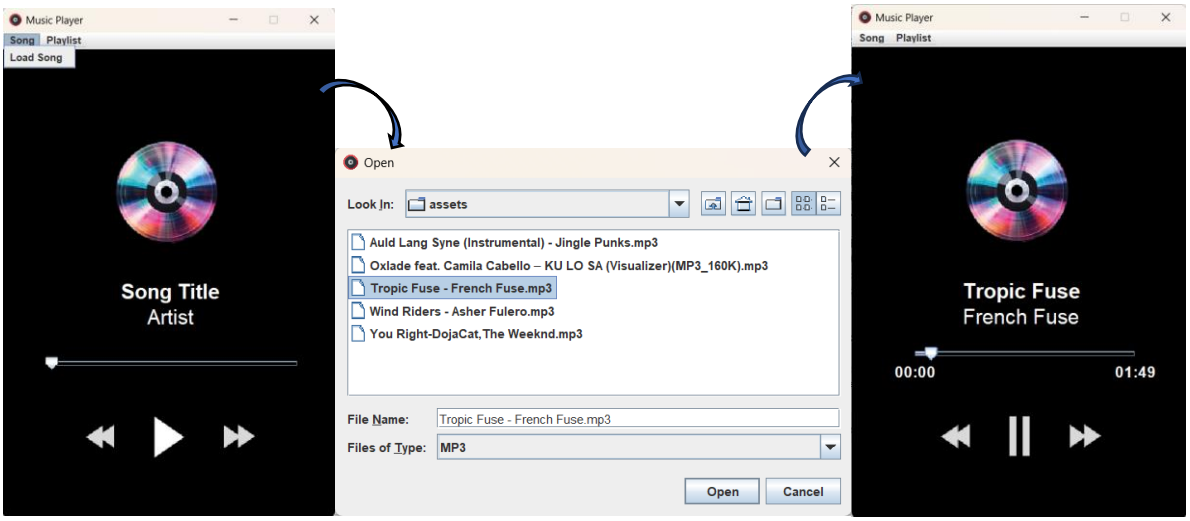
Music Player GUI:

- Title: *Music Player Interface*
- Description: The main interface where users can play, pause, and navigate songs. It includes song metadata, playback controls, and a progress bar.



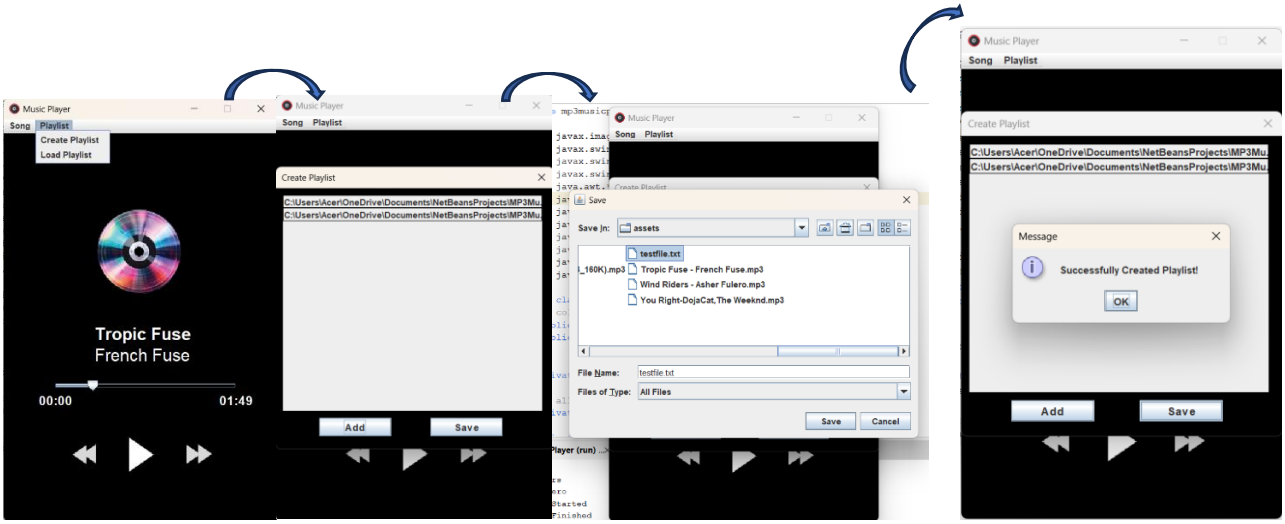
Playback in Progress:

- Title: *Playback Screen*
- Description: After loading a song, this window shows the song currently playing, with playback controls and a progress bar for the track.



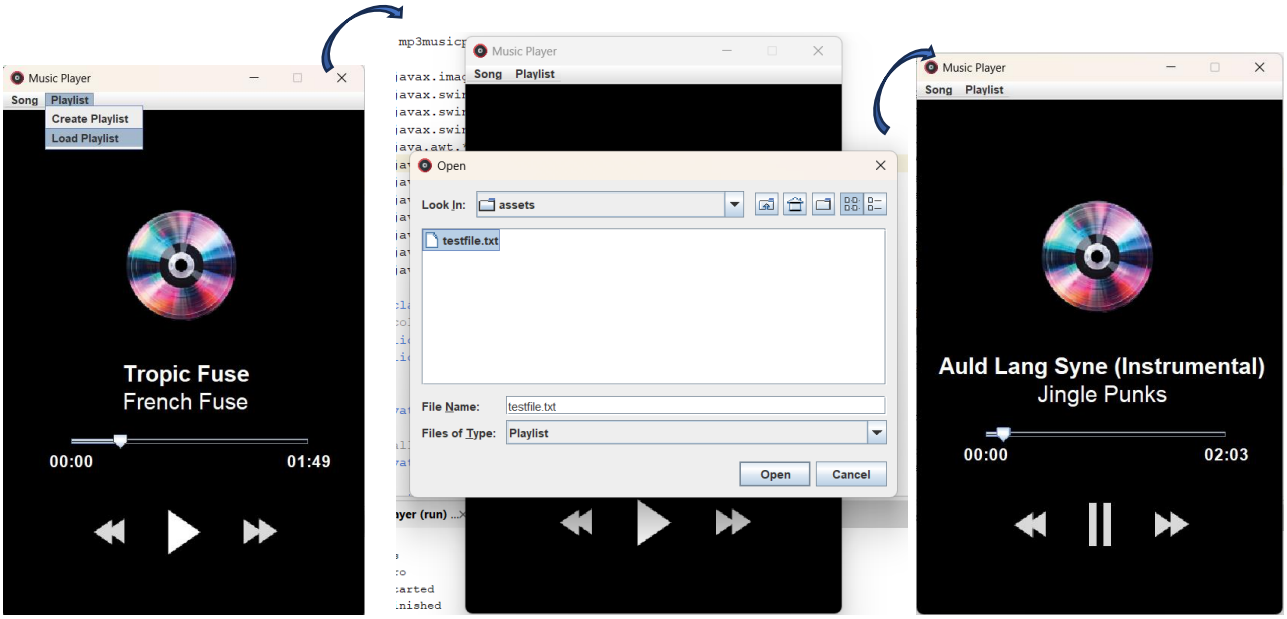
Playlist Creation:

The steps for creating a playlist (adding .mp3 files to a .txt file) are shown below:

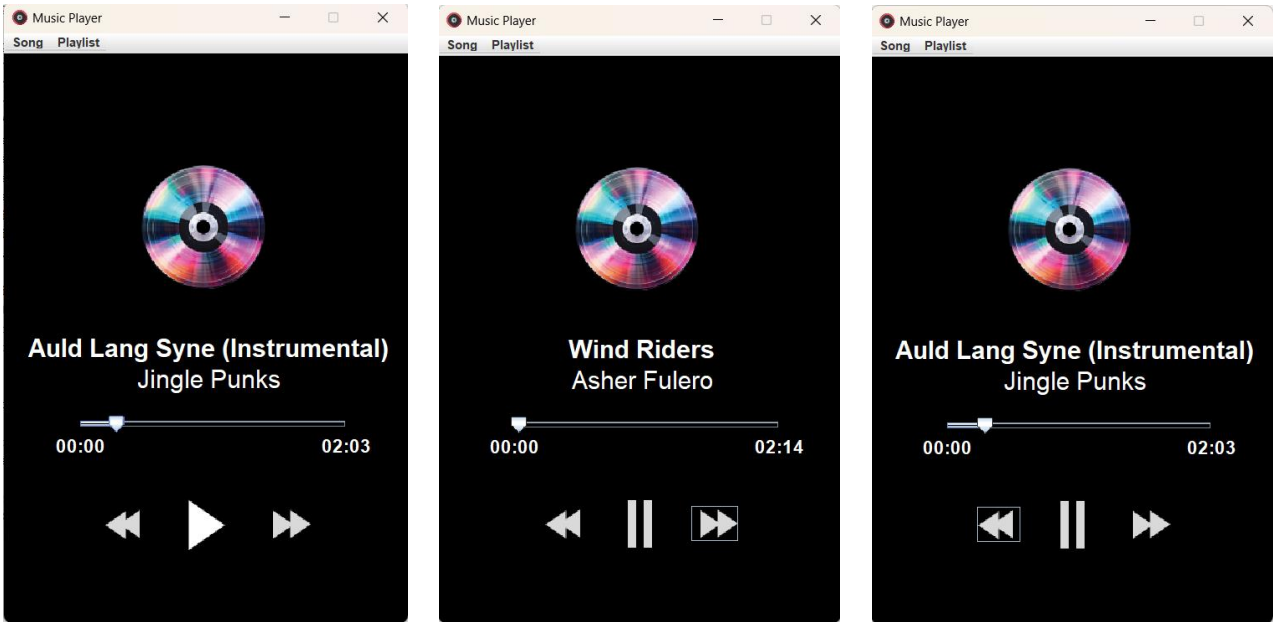


Loading The Playlist:

After creating a playlist if we want to play the songs of it then we might need to follow the following steps-



Playback Control:



Paused Song *Forward Button (Next Song)* *Backward Button (Previous Song)*

Challenges Faced and Solutions

1. **Multithreading for Playback**:
    - **Challenge**: Ensuring music playback continues smoothly while the GUI remains responsive.
    - **Solution**: We used multithreading to separate the music playback process from the GUI updates. This allows the user to control the app while listening to music without interruptions.
2. **File I/O for Playlist Persistence**:
    - **Challenge**: Managing playlist data and ensuring that songs can be added, saved, and loaded correctly from files.
    - **Solution**: We implemented file I/O to save playlist data in a structured format and used error handling to ensure robust file operations.
3. **UI Design and User Experience**:
    - **Challenge**: Designing a user-friendly interface that was both functional and visually appealing.
    - **Solution**: Java Swing was used to create a clean and intuitive GUI, with easy-to-navigate controls and a responsive layout.

Conclusion

The **Music Playlist Manager** is now a fully functional Java application that allows users to create, manage, and play their music playlists. With features such as multithreading for smooth playback, the ability to save and load playlists, and a user-friendly graphical interface, this project demonstrates our ability to apply Java programming concepts such as object-oriented design, file I/O, and multithreading to a real-world application.

The project was completed successfully, meeting all objectives and demonstrating effective problem-solving skills in handling audio playback, UI design, and playlist management.

We are confident that the **Music Playlist Manager** provides a seamless and enjoyable music experience for users, and it serves as an excellent showcase of our development skills.

Future Work and Improvements

Although the application is complete, there are potential areas for future development:

- **Remove Song:** We could add a feature to remove song manually by the user by directly modifying the playlist with appropriate methods and updating the GUI.
- **Support for More Audio Formats**: Currently, the app supports MP3 files; we could expand it to handle other formats such as WAV or FLAC.
- **Improved UI Design**: A more modern design could be implemented, possibly with additional features like album art display or playlist search.

- **Advanced Features**: Integration with online music libraries, streaming services, or adding a recommendation system could provide an even richer user experience.

Reference

- DataFlair- https://data-flair.training/blogs/java-music-player/
- YouTube channel TapTap (https://www.youtube.com/@TapTap_196)
  Video link- https://youtu.be/xf0aH2K3oJ4?si=0Mzj4jup4VkygAfN
  We are deeply grateful to them!
- ChatGPT

---------- 0 ----------