Data Structures And Algorithms

Subject Code: CSL 209

Project Report



Faculty name:

Dr.Anuradha Dhull

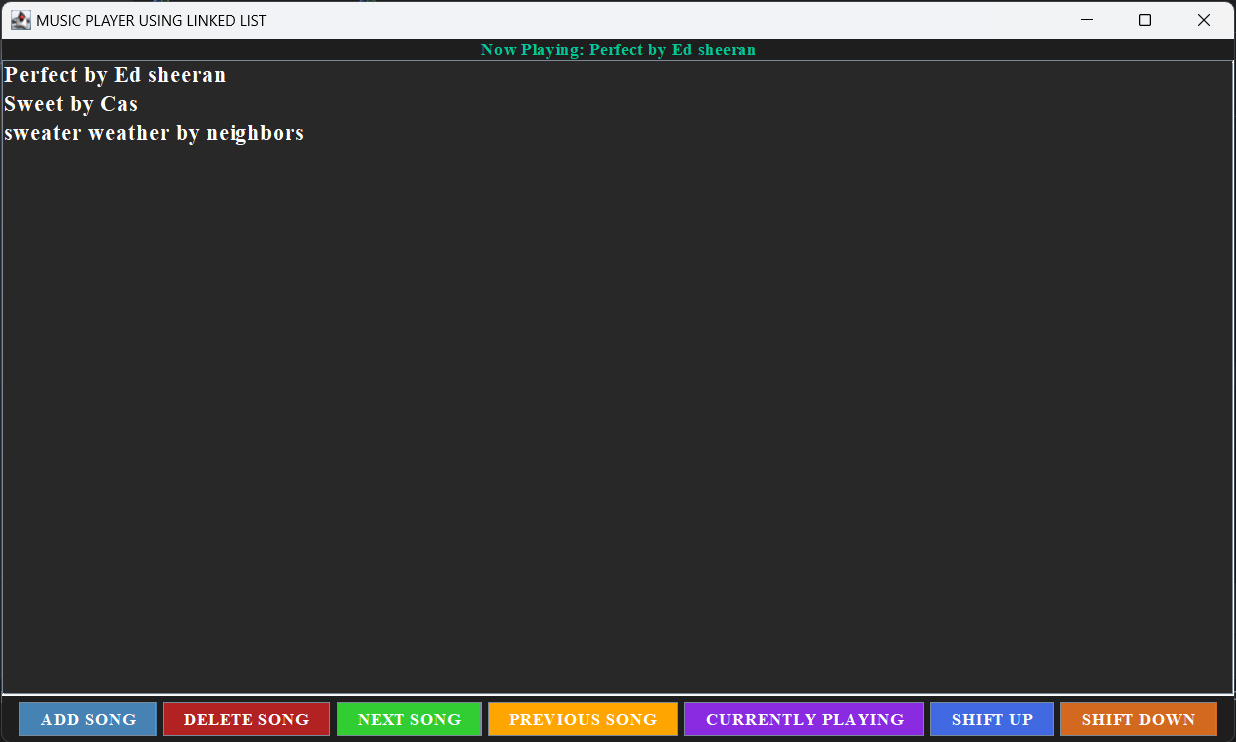
Team:

* Purab Gupta (23csu252)
* Rujul Bakshi (23csu269)

**Department of Computer Science and Engineering The NorthCap University, Gurugram- 122001, India**

Project Title:

MUSIC PLAYER



**ABSTRACT**

The **Music Player Application** is a Java-based desktop application designed to simulate the functionality of a music playlist manager. Leveraging a doubly linked list data structure, the application allows users to efficiently manage and interact with a playlist of songs. The graphical user interface (GUI) is implemented using the Swing framework, offering an intuitive and visually appealing user experience.

Key features include adding new songs with metadata (title, artist, and duration), removing songs, navigating through the playlist (next and previous), displaying the currently playing song, and rearranging songs within the playlist. The application ensures seamless playlist management while providing real-time updates to the "Now Playing" display. Advanced functionalities, such as the ability to move songs up or down in the playlist, enhance user control and flexibility.

This project showcases the integration of data structures with GUI development, demonstrating practical applications of Java programming in real-world scenarios. It aims to provide a user-friendly platform for managing music playlists while highlighting the importance of linked lists in dynamic data management. The application is a testament to effective software design principles and Java programming concepts.

**INDEX**

**1. Code Overview**

**2. User Options**

**3. Summary**

**4. Tools used**

**Objectives**

The primary objectives of the MUSIC PLAYER project include:

1. **Implementing Data Structures in Real-World Applications**  
   Utilize a doubly linked list to efficiently manage and manipulate a playlist, demonstrating the practical use of data structures in software development.
2. **Facilitating Playlist Management**   
   Provide users with essential playlist management functionalities, such as adding, removing, navigating, and reordering songs.
3. **Demonstrating Real-Time Interactivity**Implement dynamic updates to the "Now Playing" display, ensuring real-time feedback and interactivity for the user.
4. **Developing a User-Friendly GUI**Design an intuitive and interactive graphical user interface (GUI) using the Java Swing framework to simulate a music player application.

**The code consists of following concepts:**

1. **Object-Oriented Programming (OOP):**

**Classes and Objects:** The Song class represents individual songs as objects with attributes like title, artist, and duration.

**Encapsulation:** Song details and playlist management are encapsulated within the

1. **Data Structures:**

**Doubly Linked List:** Used to manage the playlist, allowing efficient forward and backward traversal, addition, removal, and reordering of songs.

1. **Algorithms:**

**Traversal:** Iterating through the linked list to find, add, or remove a song.

**Reordering:** Logic to swap nodes in the linked list when moving songs up or down.

1. **Error Handling:**

Validation for empty playlists and non-existent songs to ensure robust application behavior.

1. **User Input Handling:**

Prompting the user for song details (title, artist, duration) via input dialogs and ensuring the data is incorporated into the playlist.

1. **Graphical User Interface (GUI):**

Java Swing: The GUI framework provides components like JFrame, JList, JButton, and JLabel for building an interactive application.

**CODE REVIEW**

**MusicPlayer() (Constructor):**

* Initializes the GUI components: playlist display, control buttons, and "Now Playing" label.
* Sets up the layout (BorderLayout) and adds components to the frame.
* Configures visual properties like background colors, fonts, and button listeners for interaction.

**Song Class:**

* Represents a node in the doubly linked list.
* Each song has attributes (title, artist, duration) and pointers (next, prev) for list navigation

**createColoredButton(String text, Color bgColor, Color fgColor)**

* Utility method to create stylized buttons with consistent appearance and behavior.

**addSong()**

* Prompts the user for song details and creates a new Song node.
* Adds the node to the end of the doubly linked list and updates the playlistModel to reflect the change visually.
* Updates the "Now Playing" label.

**removeSong()**

* Removes a song based on the title entered by the user.
* Traverses the linked list to find the song, adjusts pointers to maintain list integrity, and updates the GUI.
* Handles edge cases like empty playlist or non-existent song.

**playNext()**

* Moves the current pointer to the next song in the playlist.
* Displays an appropriate message if at the end of the list or if the playlist is empty.

**playPrevious()**

* Moves the current pointer to the previous song in the playlist.
* Displays an appropriate message if at the start of the list or if the playlist is empty.

**moveSongUp()**

* Swaps the positions of the selected song and the one above it in the doubly linked list.
* Adjusts list pointers (next, prev) and updates the playlistModel to reflect changes in the display.
* Handles edge cases such as the first song in the list.

**moveSongDown()**

* Swaps the positions of the selected song and the one below it in the doubly linked list.
* Adjusts list pointers and updates the playlistModel.
* Handles edge cases such as the last song in the list.

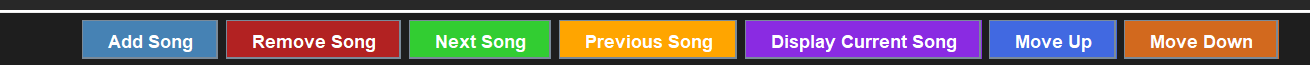
**getSongAt(int index)**

* Traverses the doubly linked list to retrieve the song node at a specific index.
* Facilitates list manipulation operations like moving songs up or down.

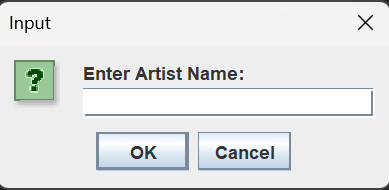
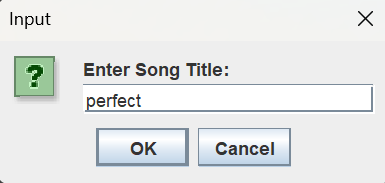
**updateNowPlaying()**

* Updates the "Now Playing" label to display the details of the current song.
* Displays "None" if the playlist is empty or no song is selected.

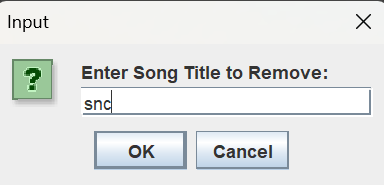
**INPUT OPTIONS**



**Add Song:**

****

**Remove song:**

****

**SUMMARY**

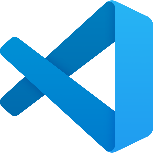
The Music Player project is a Java-based desktop application that simulates a playlist management system using a linked list data structure. It provides an intuitive GUI built with Swing for adding, removing, and navigating through songs in the playlist. Each song is represented as a node in a doubly linked list, enabling forward and backward traversal. Users can play the next or previous song, display the currently playing song, and rearrange songs in the playlist. The project integrates various concepts like object-oriented programming, data structures, event handling, and graphical user interface design, offering a comprehensive solution for playlist management in a user-friendly format.

**TOOLS USED**

THE TOOLS THAT WE HAVE USED IN THIS PROJECT ARE AS FOLLOWS:

1. VSCODE

TO RUN AND COMPILE THE CODE.



2. MICROSOFT POWERPOINT

TO CREATE THE PRESENTATION



3. ZOOM

TO DISCUSS THE PROJECT AMONGST THE GROUP.



4. INTELLIJ

PRIMARY CODING APP TO CODE AND TI]O CREATE JDBC

