**Project Report**



**Music Player And Playlist Manager**

**Group members:**

M. Saim Chughtai M. Umar Saleem

01-134241-033 01-134241-035

**Computer Science Department  
Bahria University Islamabad**

**1. Introduction**

This project is a console-based Music Player and PlaylistManager implemented in C++ using Data Structures and Algorithms (DSA). It allows users to:

* Add, delete, and search songs
* Play songs and track recently played tracks
* Sort and manage playlists
* Save and load playlists from a file

The program uses Doubly Linked Lists for the main playlist and a Stack for recently played songs.

**2. Objectives**

* Implement a **user-friendly** music playlist manager.
* Use **DSA concepts** (Linked Lists, Stack, Searching, Sorting).
* Ensure **data persistence** using file handling.
* Handle **input validation** for robustness.

**3. Features**

| **Feature** | **Description** |
| --- | --- |
| **Add Song** | Add songs to the playlist (spaces replaced with \_) |
| **Delete Song** | Delete by name or position |
| **Display Playlist** | Show all songs in the playlist |
| **Search Song** | Linear search for a song |
| **Play Song** | Mark a song as "Now Playing" and push to recent tracks |
| **Recently Played** | Stack-based history of played songs |
| **Sort Playlist** | Alphabetical sorting (Bubble Sort) |
| **Load/Save Playlist** | File I/O operations for persistence |
| **Error Handling** | Validates inputs and prevents crashes |

**4. Data Structures & Algorithms Used**

**4.1 Data Structures**

| **Structure** | **Purpose** |
| --- | --- |
| **Doubly Linked List** | Main playlist storage (node with prev and next) |
| **Stack** | Recently played songs (top pointer, LIFO order) |
| **Arrays (Strings)** | Store song names (char song[100]) |

**4.2 Algorithms**

| **Algorithm** | **Used In** |
| --- | --- |
| **Linear Search** | search1(), play(), del\_search() |
| **Bubble Sort** | sort() (alphabetical order) |
| **File Handling** | tofile(), delete\_file(), addplaylist() |

**5. Code Structure**

| **File/Module** | **Functions** |
| --- | --- |
| **Main Program** | main(), menu-driven interface |
| **Playlist Management** | add\_node(), del\_node(), printlist(), count\_nodes() |
| **Search & Play** | search1(), play(), recent(), topelement() |
| **Sorting** | sort() (Bubble Sort) |
| **File Operations** | tofile(), delete\_file(), addplaylist() |
| **Error Handling** | validateSongName(), input checks |

**6. Flow of Execution**

1. **Program Starts** → User enters playlist name.
2. **Main Menu** → Options for adding, deleting, searching, playing songs.
3. **File Handling** → Songs are saved in playlist.txt.
4. **Recently Played** → Stack keeps track of last played songs.
5. **Exit** → Program saves changes and closes.

**7. Screenshots (Sample Output)**

**Main Menu**

\*\*WELCOME TO MUSIC PLAYLIST\*\*

1. Add Song

2. Delete Song

3. Display Playlist

4. Total Songs

5. Search Song

6. Play Song

7. Recently Played

8. Last Played

9. Sort Playlist

10. Load Playlist from File

11. Exit

Enter your choice:

**Adding a Song**

Enter Song name (use '\_' instead of spaces): Bohemian\_Rhapsody

Song added successfully!

**Playing a Song**

Now Playing: Bohemian\_Rhapsody

Added to Recently Played.

**Recently Played**

Recently played tracks:

1. Bohemian\_Rhapsody

2. Sweet\_Child\_O\_Mine

**8. Advantages**

 Efficient Storage (Linked Lists for dynamic playlist)  
 Fast Search & Play (Linear Search)  
 Persistent Data (File Handling)  
 User-Friendly (Input validation, clear menus)

**9. Limitations & Future Improvements**

| **Limitation** | **Possible Improvement** |
| --- | --- |
| **Console-based only** | GUI (Qt, SFML) |
| **Basic sorting (Bubble Sort)** | Merge Sort / Quick Sort |
| **No multi-playlist support** | Use Hash Tables for multiple playlists |
| **No pause/volume control** | Integrate audio libraries (SFML, SDL) |

**10. Conclusion**

This project successfully demonstrates:  
 **DSA concepts** (Linked Lists, Stack, Searching, Sorting)  
 **File handling** for persistent storage  
 **User input validation** for robustness

It serves as a foundation for more advanced music player applications.