**DATA STRUCTURE AND ALGORITHM LAB**

**PROJECT PROPOSAL:**



**GROUP MEMBERS:**

|  |  |
| --- | --- |
| **NAME:** | **ENROLLMENT NO:** |
| **IQRA HAFEEZ** | **01-131232-036** |
| **AYESHA ARSHAD** | **01-131232-018** |

**Department of Software Engineering,**

**Bahria University Islamabad**.

**PROJECT TITLE:**

**PLAYLIST MANAGER WITH SMART NAVIGATION**



**TEAM NAME:**

**INCHARGE TEAM**

**PROJECT DESCRIPTION:**

The Music Playlist Manager is a GUI-based (Graphical User Interface) application designed to allow users to manage their music playlists. The core functionalities will include adding, deleting, and editing tracks, searching and sorting them, and even integrating with Spotify’s API to get live music data. The project will also provide a smooth navigation experience, where users can easily move between tracks in the playlist.

The word "Navigation" indicates that users will be able to control and move between tracks smoothly.

**Objectives:**

* **Build an intuitive playlist manager:** The main goal is to create a simple, user-friendly application where users can perform basic actions like adding, deleting, and editing tracks in their playlists.
* **Seamless navigation and search**: The app will allow users to search for specific tracks by title, artist, or album. Users will also be able to navigate through the playlist easily.
* **Integrate Spotify API:** For an enhanced experience, the project will include a Spotify API integration that allows users to access live music data from Spotify, such as track details or availability.
* **MySQL for persistent data storage**: A MySQL database will be used to store the playlist data permanently, even if the application is closed and reopened.

**Technology Stack:**

**Programming Language**

**C++ (with .NET Windows Forms):** C++ will be the main language used for the project, while .NET Windows Forms will help in building the GUI. This combination is powerful for desktop applications and provides flexibility in design.

**Tools and Frameworks:**

* **Visual Studio**: This IDE (Integrated Development Environment) is ideal for C++ development with Windows Forms, offering features like debugging and project management.
* **.NET Windows Forms**: This framework allows you to create rich desktop applications with graphical interfaces, making it easy to design and implement the GUI.
* **Spotify API:** Integrating this API allows the app to fetch real-time music data from Spotify, enriching the user experience by adding information such as artist names, track availability, and more.

**Database:**

**MySQL:** A reliable relational database that will be used to store user-created playlists, including track titles, artists, durations, and album names. This ensures that playlists are saved and can be accessed later, even after the application is closed.

**Features:**

* **Basic Features:**
* Add, delete, and edit tracks: Users will have the ability to add new music tracks, delete existing ones, and update track details like the title or artist.
* **Search tracks**: Users can search for specific tracks using keywords like the track's title, artist, or album.
* **Display playlists**: The app will display all tracks in the playlist in an easy-to-read format, using the List View component to show details like title, artist, and duration.

**Advanced Features:**

* **Sort playlists:** Users will be able to sort their playlists by different attributes such as track title, artist, or duration, making it easier to organize the music.
* **Navigate between tracks**: Features like "Next" and "Previous" buttons will allow users to easily move between tracks in the playlist.
* **Spotify API integration**: This feature will allow users to fetch live data from Spotify, such as artist information, track availability, and more, enhancing the user experience.
* **Persistent storage**: The playlist will be stored in a MySQL database, meaning users can save their playlists and load them later.

**Data Structures:**

**Used Data Structures:**

* + **Queues:** A queue will be used to manage the playback order of the tracks. As the user listens to a song, the next track in line will automatically start playing, providing a seamless experience.
  + These ensure that tracks play in the correct order, making it easy to queue up songs and listen without interruptions.
  + **Linked Lists**: A linked list will be used to dynamically store and manage the playlist. This allows easy addition or deletion of tracks without rearranging the entire list, making it efficient for playlist management.
  + They allow the playlist to grow and shrink dynamically, making it easy to add or remove tracks from any position.
  + **Arrays:** Arrays will be used for smaller, static datasets, such as temporary storage or quick access to certain pieces of data like track IDs.
  + Efficient for storing and accessing small collections of data, like track IDs or temporary lists.
  + **Binary Search Trees (BST):** For advanced searching and sorting functionalities, a Binary Search Tree will help efficiently find and sort tracks based on attributes like title or artist.
  + BSTs provide fast search and sorting capabilities, ensuring that users can quickly find tracks or sort them based on their preferences.

**System Design:**

* **User Interface**
* The user interface will be built using .NET Windows Forms, which allows easy creation of interactive forms. Key components of the UI will include:
* **Textboxes**: To allow users to input track details such as title, artist, and album.
* **Buttons:** For operations like Add, Delete, Search, and Sort.
* **List View**: A view to display the playlist with columns for track title, artist, and duration.

**Database Design:**

The MySQL database will have a Tracks table with the following schema:

* Track\_id: A unique identifier for each track (Primary Key).
* Title: The title of the track.
* Artist: The artist of the track.
* Duration: The length of the track in seconds.
* Album: The album the track belongs to.

**Development Plan**

**Milestones**

* Plan and design the project structure. Create the user interface and set up the MySQL database to store playlist data.
* Implement the CRUD operations for managing tracks (Add, Delete, Edit). Integrate the Spotify API to fetch live data.
* Implement data structures like queues and linked lists to handle playlist management.
* Implement sorting and searching features, using Binary Search Trees or other algorithms.
* Test the application, fix bugs, and prepare the final documentation.

**Testing:**

Test cases will be created for:

* **CRUD operations**: Ensure that tracks can be added, deleted, and edited.
* **Track navigation**: Verify that users can move between tracks (Next, Previous).
* **Spotify API:** Test the integration with Spotify to ensure the live music data is fetched correctly.
* **Database connectivity:** Make sure the app can successfully connect to and interact with the MySQL database.

**TECHNOLOGIES:**

* **Git**: Version control system to track changes in the project and collaborate with others.
* **Jira:** Task management tools to organize and assign tasks, ensuring timely completion of milestones.
* **VS:** VS as an IDE.
* **API:** API for song collection.
* **SQL:** SQL for data storage.
* **GUI:** GUI based implementation.

**CONCLUSION:**

We are creating a GUI based web application using the concepts of DSA including binary trees, queues, stacks and linked lists. Where the user will be able to navigate through an online platform for the search collection and saving of different songs and playlist.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***