Capital University of Science & Technology

Project Topic

Music Player

Group Members

1. Muhammad Haseeb Ahmad
2. Mohammad Abdul Rafay
3. Saif Akhtar
4. Muhammad Hamza Tariq

Contents

1. Project proposal

Our tem will develop the music player. In this project we will use the double ended circular linked List Double ended priority Queue. The Double ended priority Queue is used to create the playlist, Store all of the songs, display the songs in reverse, Forward, Delete any song from the playlist.

2. Scope of Project

Through the life time we can work all of time and we did not get enough time for them self so we are going to develop the music player using C++.

3. Features/Requirements

1. Can create your own playlist.
2. Download new song s and keep all of the song.
3. You can arrange the songs according to the user.
4. Easy to Use Interface.

4. Data Structure:

In our Application we are using Double ended priority queue. The reason of using type of data structure is just to create a playlist and give the user full control over the application.

5. Code of the Application:

```
   #include<iostream>
    #include<string>
using namespace std;

    struct node {
            int song_id;
            string song_name;
            string singer_name;
            string movie;
            node * next;
            node * prev;
    };

    class dll {
            node *front = nullptr;
            node *rare = nullptr;

    public:

            void insert(int value, string song, string singer, string movie)
            {
```

```
        system("color 9C");
        node *temp = new node();

        if (front == nullptr)
        {

            temp->song_id = value;
            temp->singer_name = singer;
            temp->song_name = song;
            temp->movie = movie;
            temp->prev = nullptr;
            temp->next = front;
            front = temp;
            rare = temp;
        }
        else
        {
            system("color 8C");
            int i = 0;
            if (checkD(value,i))
            {

                if (i==1) {
                    cout << "The Id already exit in the Music Player" << endl;

                    return;
                }

                else
                {
                    temp->song_id = value;
                    temp->singer_name = singer;
                    temp->song_name = song;
                    temp->movie = movie;
                    rare->next = temp;
                    temp->prev = rare;
                    temp->next = nullptr;
                    rare = temp;
                }
            }


        }
    }

    int checkD(int value,int i)
    {

        int a = value;
        node * temp= front;
        while (temp != nullptr)
        {
                cout << "Inside the Loop" << endl;
                if (a == temp->song_id)
                {
                        i = 1;
                        return i;
                }

                temp = temp->next;
        }


}
void delete_node() {
        system("color 10C");
        int song_id;

        cout << "\nEnter the Song ID to delete : ";
        cin >> song_id;

        delete_song(song_id);
}

void delete_song(int song_id)
{
        system("color 6C");
        node * current = front;
        while ((song_id != current->song_id) || (current != nullptr)) {
                if (song_id == current->song_id)
                {
                        current->prev->next = current->next;
                        current->next->prev = current->prev;
                        delete current;
                        cout << "Song Deleted.\n";
                        return;
                }
                current = current->next;
        }
}
void display()
{
        system("color A5");
        node *current = front;

        if (front == nullptr)
        {
                cout << "No Song Exit." << endl;
        }
        else
        {
            while (current != nullptr)
            {

                //system("cls");
                cout << "song ID : " << current->song_id << "\n";
                cout << "Singer Name : " << current->singer_name <<
"\n";
            cout << "Song : " << current->song_name << "\n";
                cout << "Movie Name : " << current->movie << "\n";
                cout
<<"__________________________________________________________\n\n\n";


                current = current->next;
            }
        }

        cout << "\nAll the information displayed.\n";
    }

    void dispaly_reverse()
    {
        system("color 8C");
        node *current = rare;

        while (current != nullptr) {
            //      system("cls");
            cout << "song ID : " << current->song_id << "\n";
            cout << "Singer Name : " << current->singer_name << "\n";
            cout << "Song : " << current->song_name << "\n";
            cout << "Movie Name : " << current->movie << "\n";
            current = current->prev;
        }
        cout << "\nAll the information displayed.\n";
    }

    void menu() {
        system("color A1");
        int choice;
        do {

            //system("cls");
            cout            <<              "\n****************Audio
Player****************\n";
            cout << "1.Display all songs.\n";
            cout << "2.Display the songs in reverse order.\n";
            cout << "3.Insert the song.\n";
            cout << "4.Sort the songs.\n";
            cout << "5.Delete the song.\n";
            cout << "\nEnter your choice : ";
            cin >> choice;


            condition_checker(choice);

        } while (choice != 0);


    }

    void condition_checker(int choice)
    {
        system("color 8C");
        if (choice == 0) {
            cout << "\nGood Bye ";
        }
        else if (choice == 1) {
            cout << "\n****************All
Playlist****************\n";
            display();
        }
        else if (choice == 2) {
            cout << "\n****************Reverse
Playlist****************\n";
            dispaly_reverse();
        }
        else if (choice == 3) {
            cout << "\n****************Insertion in a
Playlist****************\n";
            sending_data();
        }
        else if (choice == 4) {
            cout << "\n****************Sorting****************\n";
            cout << "The songs has been sorted.\n";
            sort();
        }
        else if (choice == 5) {
            cout <<
"\n****************Deleation****************\n";
            delete_node();
        }
        else {
            cout << "\nSome error found in insertion of data check manually
and try again.";
        }
    }

    void sending_data()
    {
        system("color 7C");
        int song_id;
        string song_name;
        string singer_name;
        string movie;

        cout << "Enter the song id : ";
        cin >> song_id;
        cin.ignore();
        cout << "\nEnter the song name : ";
        getline(cin, song_name);
        cout << "\nEnter the Singer name : ";
        getline(cin, singer_name);
        cout << "\nEnter the name of movie : ";
        getline(cin, movie);
        insert(song_id, song_name, singer_name, movie);
    }

    void sort()
    {
        system("color 9C");
        node *current = front;
        node *curr = front->next;
        while (current != nullptr) {
            if (current->song_id < curr->song_id) {
                swap(current->song_id, curr->song_id);
                swap(current->singer_name, curr->song_name);
                swap(current->singer_name, curr->singer_name);
                swap(current->movie, curr->movie);
            }
            current = current->next;
            if (current->next == curr->next) {
                cout << "Data sorted.";
                return;
            }
            curr = curr->next;
        }
    }
};

int main() {
    dll d;
    d.menu();
    system("pause");
    return 0;
}
```

6. Total Functions:

There are a lot of Functions that are used in this Application sand some of these functions are given bellow:

- **Insert function**

This Function can be used to add any song into the Playlist.

- **Check DID function**

This function to check if there are any duplicate id in the system or if there is any duplicate id in the system it will remove the duplicate id present into the system or the user is trying to enter the same id which is already exit in the system, then this function will not allow the user to enter the Data into the playlist.

- **Delete Song Function**

This function can be used to delete the song from the Playlist

- **Display the playlist in forward.**

This function is used to display all of the forward method.

- **Display Reverse Function**

This function will display all of the data in reverse method.

- **Menu function**

This Function is the front end of the Application

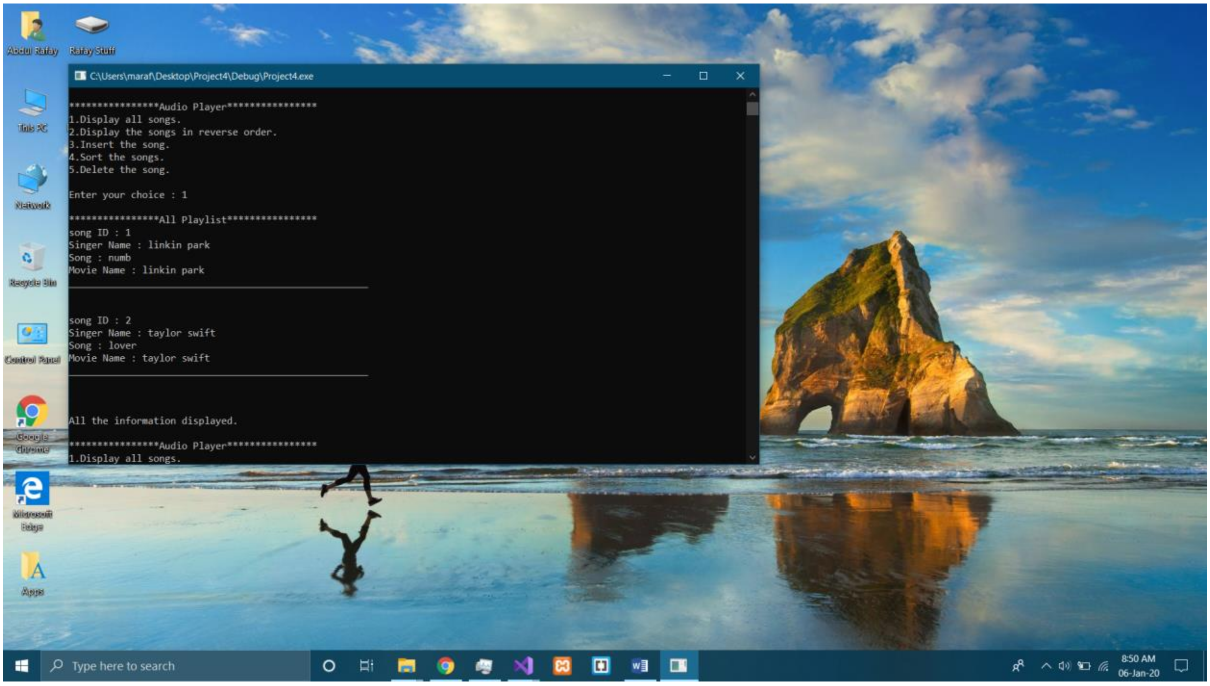
- **Sending data Function**

This function is used to get the data from the user and then sent it to the Insert function

- **Condition Checker function**

This Function will give the user what to do in the Application or in other word there will be different option and the user have to select any one of the options.

7. Output:



```
****************Audio Player****************
1.Display all songs.
2.Display the songs in reverse order.
3.Insert the song.
4.Sort the songs.
5.Delete the song.

Enter your choice : 1

****************All Playlist****************
song ID : 1
Singer Name : linkin park
Song : numb
Movie Name : linkin park
_____________________________________________


song ID : 2
Singer Name : taylor swift
Song : lover
Movie Name : taylor swift
_____________________________________________



All the information displayed.

****************Audio Player****************
1.Display all songs.
```