CA 3: Experiential Learning

Group Members:


| Sr. No. | PRN | Name of Student | Mail id |
|---|---|---|---|
| 1. | 22070122079 | HIMANI ARORA | himani.arora.btech2022@sitpune.edu.in |
| 2. | 22070122063 | MEET GOLANI | meet.golani.btech2022@sitpune.edu.in |
| 3. | 22070122080 | HIMANSHU CHOPADE | himanshu.chopade.btech2022@sitpune.edu.in |


**Problem Statement: Develop a "SPAMTIFY" Music Player Management System allowing users to create playlists, add songs, and play music. Users can search for songs by partial name and manage playlists seamlessly, including options to pause, skip, and quit during playback.**

**Explanation:**

The Music Player Management project is a simplified music player application developed using the C++ programming language. It aims to provide a basic music player with essential features for playback control and song searching.

- Basic Playback Control:
    - This project includes functionality for playing and pausing songs. Users can easily start or pause playback, providing a fundamental user experience in music playback.
- Song Search:
    - The application offers a search feature based on song titles or artists. This allows users to quickly locate specific songs in their collection, making it more user-friendly.
- C++ Classes and Inheritance:
    - The core of the project's implementation revolves around the use of classes and inheritance. Classes are used to model songs, and inheritance is employed to structure and organize the code efficiently. For example, a base "Song" class can be inherited to create specialized classes for different types of songs or audio files

**Class Diagram:**


## Code snippets:

```cpp
class Song {
protected:
    string title;
    string artist;
    string file_name;

public:
    Song() {
        title = "";
        artist = "";
        file_name = "";
    }

    Song(string title1, string artist1, string file_name1) {
        title = title1;
        artist = artist1;
        file_name = file_name1;
    }

    void play_song() {
        cout << "Playing: " << title << " by " << artist << endl;
        PlaySound(TEXT(file_name.c_str()), NULL, SND_FILENAME | SND_ASYNC);
    }

    void pause() {
        PlaySound(NULL, NULL, SND_ASYNC);
        cout << "Song paused." << endl;
    }

    string get_title() const {
        return title;
    }

    string get_artist() const {
        return artist;
    }

    string get_file_name() const {
        return file_name;
    }
};
```

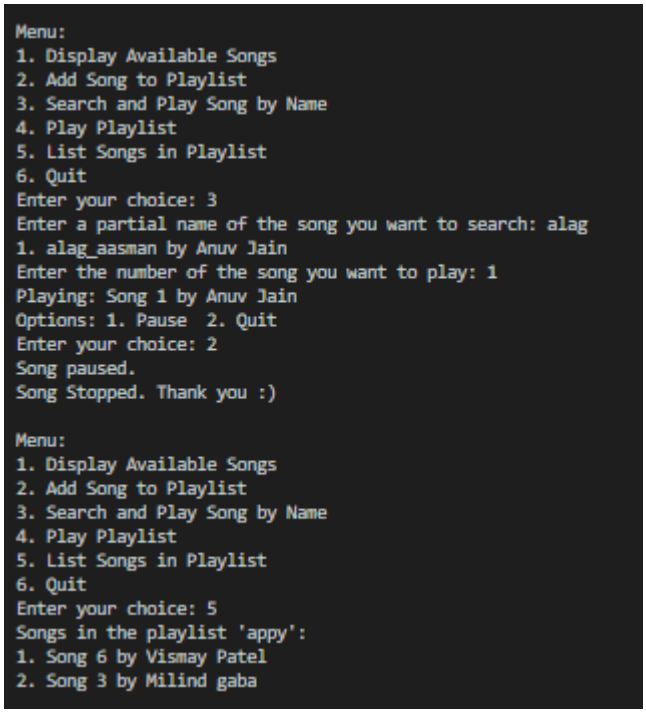
```cpp
class Playlist {
private:
    vector<Song> availableSongs;
    vector<Song> playlistSongs;
    string playlist_title;

public:
    Playlist() {
        playlist_title = "";
    }

    ~Playlist() {
        // Ensure songs stop when the program exits
        PlaySound(NULL, NULL, SND_ASYNC);
    }

    void createPlaylist() {
        cout << "Enter the title of your playlist: ";
        cin.ignore();
        getline(cin, playlist_title);
    }

    void addSongToAvailableSongs(string title, string file_name, string artist) {
        // Check if the song is already in the availableSongs list
        for (const Song& song : availableSongs) {
            if (song.get_title() == title) {
                cout << "Song '" << title << "' by " << artist << " is already in the available songs list." << endl;
                return;
            }
        }
        availableSongs.push_back(Song(title, artist, file_name));
    }

    void addSongToPlaylist(int index) {
        if (index >= 1 && index <= availableSongs.size()) {
            const Song& songToAdd = availableSongs[index - 1];

            // Check if the song is already in the playlist
            for (const Song& song : playlistSongs) {
                if (songToAdd.get_title() == song.get_title()) {
                    cout << "Song '" << songToAdd.get_title() << "' by " << songToAdd.get_artist() << " is already in your playlist." << endl;
                    return;
                }
            }

            playlistSongs.push_back(songToAdd);
            cout << "Added " << songToAdd.get_title() << " by " << songToAdd.get_artist() << " to your playlist." << endl;
            availableSongs.erase(availableSongs.begin() + index - 1); // Remove the song from available songs
        } else {
            cout << "Invalid choice. Please try again." << endl;
        }
    }
```

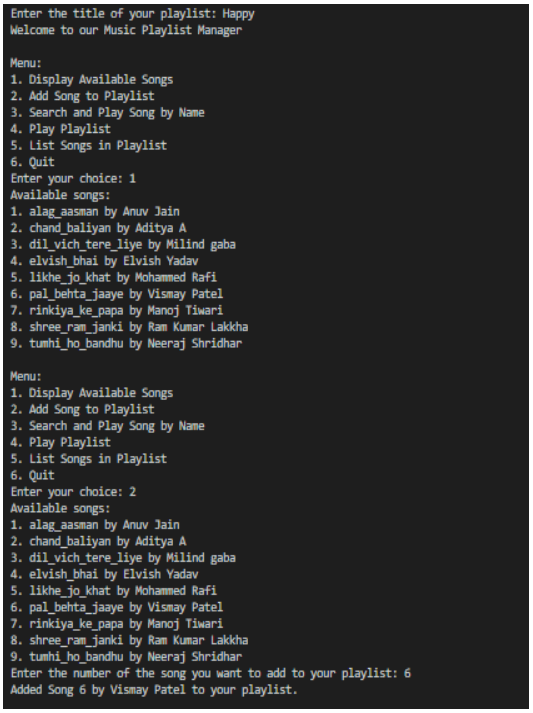
```
void displayAvailableSongs() {
    cout << "Available songs:\n";
    for (int i = 0; i < availableSongs.size(); i++) {
        string songName = availableSongs[i].get_file_name();
        songName = songName.substr(0, songName.find_last_of('.'));
        cout << i + 1 << ". " << songName << " by " << availableSongs[i].get_artist() << endl;
    }
}

void searchAndPlayByPartialName(string partial_name) {
    bool found = false;

    for (int i = 0; i < availableSongs.size(); i++) {
        string fileName = availableSongs[i].get_file_name();
        size_t dotPos = fileName.find_last_of('.');
        string fileNameWithoutExt = fileName.substr(0, dotPos);

        if (availableSongs[i].get_title().find(partial_name) != string::npos || fileNameWithoutExt.find(partial_name) != string::npos) {
            cout << i + 1 << ". " << fileNameWithoutExt << " by " << availableSongs[i].get_artist() << endl;
            found = true;
        }
    }

    if (!found) {
        cout << "No matching songs found." << endl;
        return;
    }

    int choice;
    cout << "Enter the number of the song you want to play: ";
    cin >> choice;

    if (choice >= 1 && choice <= availableSongs.size()) {
        availableSongs[choice - 1].play_song();

        while (true) {
            cout << "Options: 1. Pause  2. Quit\n";
            cout << "Enter your choice: ";
            cin >> choice;

            if (choice == 1) {
                availableSongs[choice - 1].pause();
            } else if (choice == 2) {
                availableSongs[choice - 1].pause();
                stop();
                return;
            } else {
                cout << "Invalid choice. Please try again." << endl;
            }
        }
    } else {
        cout << "Invalid choice. Please try again." << endl;
    }
}
```

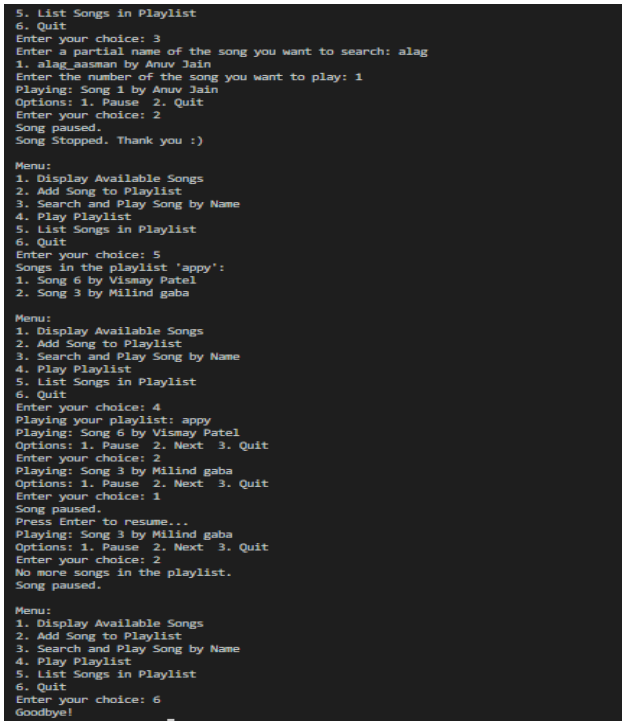
```
    void play() {
        if (playlistSongs.empty()) {
            cout << "No songs in the playlist." << endl;
            return;
        }

        cout << "Playing your playlist: " << playlist_title << endl;
        for (int i = 0; i < playlistSongs.size(); i++) {
            playlistSongs[i].play_song();
            while (true) {
                cout << "Options: 1. Pause  2. Next  3. Quit\n";
                cout << "Enter your choice: ";
                int choice;
                cin >> choice;
                if (choice == 1) {
                    playlistSongs[i].pause();
                    cout << "Press Enter to resume...";
                    cin.ignore();
                    cin.get();
                    playlistSongs[i].play_song();
                } else if (choice == 2) {
                    if (i == playlistSongs.size() - 1) {
                        cout << "No more songs in the playlist." << endl;
                        playlistSongs[i].pause(); // Stop the last song
                        return;
                    }
                    break; // to the next song
                } else if (choice == 3) {
                    playlistSongs[i].pause();
                    stop();
                    return;
                } else {
                    cout << "Invalid choice. Please try again." << endl;
                }
            }
        }
    }

    void stop() {
        // Do not clear the playlistSongs vector
        cout << "Song Stopped. Thank you :)" << endl;
    }

    void listSongs() {
        if (playlistSongs.empty()) {
            cout << "No songs in the playlist." << endl;
        } else {
            cout << "Songs in the playlist '" << playlist_title << "':" << endl;
            for (int i = 0; i < playlistSongs.size(); i++) {
                cout << i + 1 << ". " << playlistSongs[i].get_title() << " by " << playlistSongs[i].get_artist() << endl;
            }
        }
    }
};
```

```
int main() {
    Playlist myPlaylist;
    myPlaylist.createPlaylist();

    myPlaylist.addSongToAvailableSongs("Song 1", "alag_aasman.wav", "Anuv Jain");
    myPlaylist.addSongToAvailableSongs("Song 2", "chand_baliyan.wav", "Aditya A");
    myPlaylist.addSongToAvailableSongs("Song 3", "dil_vich_tere_liye.wav", "Milind gaba");
    myPlaylist.addSongToAvailableSongs("Song 4", "elvish_bhai.wav", "Elvish Yadav");
    myPlaylist.addSongToAvailableSongs("Song 5", "likhe_jo_khat.wav", "Mohammed Rafi");
    myPlaylist.addSongToAvailableSongs("Song 6", "pal_behta_jaaye.wav", "Vismay Patel");
    myPlaylist.addSongToAvailableSongs("Song 7", "rinkiya_ke_papa.wav", "Manoj Tiwari");
    myPlaylist.addSongToAvailableSongs("Song 8", "shree_ram_janki.wav", "Ram Kumar Lakkha");
    myPlaylist.addSongToAvailableSongs("Song 9", "tumhi_ho_bandhu.wav", "Neeraj Shridhar");

    cout << "Welcome to our Music Playlist Manager" << endl;

    while (true) {
        cout << "\nMenu:\n";
        cout << "1. Display Available Songs\n";
        cout << "2. Add Song to Playlist\n";
        cout << "3. Search and Play Song by Name\n";
        cout << "4. Play Playlist\n";
        cout << "5. List Songs in Playlist\n";
        cout << "6. Quit\n";
        cout << "Enter your choice: ";

        int choice;
        cin >> choice;

        switch (choice) {
            case 1:
                myPlaylist.displayAvailableSongs();
                break;
            case 2: {
                int index;
                myPlaylist.displayAvailableSongs();
                cout << "Enter the number of the song you want to add to your playlist: ";
                cin >> index;
                myPlaylist.addSongToPlaylist(index);
                break;
            }
            case 3: {
                string partial_name;
                cout << "Enter a partial name of the song you want to search: ";
                cin.ignore();
                getline(cin, partial_name);
                myPlaylist.searchAndPlayByPartialName(partial_name);
                break;
            }
            case 4:
                myPlaylist.play();
                break;
            case 5:
                myPlaylist.listSongs();
                break;
            case 6:
                cout << "Goodbye!" << endl;
                return 0;
            default:
                cout << "Invalid choice. Please try again." << endl;
                break;
        }
    }

    return 0;
```

**Input/Output:**




**GitHub repository link:** https://github.com/himanshuchopade97/Spamtify.git