**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

**Project Report**

**CS-212 Object Oriented Programming**

**Group members Name**

***Aqib Safeer (395360)***

***Zubair Mehmood (368214)***

**SUBMITTED TO:**

**Dr. Ahsan Shahzad**

**Abstract**

The purpose of this project is to create a specialised audio player that can be used to create a video game-like experience. The player has a model number and a manufacturer name, and it can play audio files from a storage media that can be mounted or unmounted. There are two types of storage media available: CD and USB, each with a fixed storage capacity. When a media is mounted, the player can play the first song and keep track of the current song being played. When the player is unmounted, the song tracking is reset. The title of the current song is also displayed. The player also supports the ability to turn on and off, as well as to save its current state when turned off. The 'Audio Playerv2' class was updated to add the ability to shuffle the current playlist, save and load playlists, add and delete audio files from the playlist, and search for audio files to the functionality of the 'Audio Player' class.

**Objective**

The objective of this OOP project is to create and implement an updated class called 'AudioPlayerV2' that extends the functionality of the previous 'Audio Player' class. The new class should be able to do things like shuffle the current playlist, save and load playlists, add and delete audio files from the playlist, and search for an audio file. The class should also keep the same characteristics and limitations as the In-car audio player described in the scenario.

**Introduction**

This project report focuses on the design and implementation of an updated version of the "Audio Player" class known as "Audio Player V2." The original "Audio Player" class was intended to play audio files only, with little or no support for video playback. One type of audio player is an in-car audio player. The new "Audio Player V2" class extends the functionality of the original class by adding new features such as the ability to shuffle the current playlist, save and load playlists, add and delete audio files from the playlist, and search for specific audio files. The in-car audio player will retain a storage media, which can be either a CD or a USB, and will only support MP3 and USBs up to 16GB in size. The player will also be able to save its current state when powered down and restore it when powered back up. Overall, the new "Audio Player V2" class will offer a more flexible and user-friendly audio playback experience.

The "Audio Player V2" class will include the ability to shuffle the current playlist, save and load playlists, add and delete audio files from the playlist, and search for specific audio files, in addition to the functionality provided by the original "Audio Player" class. This gives users more control over their audio playback experience and makes organising and accessing their music easier. The shuffle feature randomly rearranges the songs in the current playlist, adding a surprise element and making it simple to discover new music. Users will be able to save their current playlist and load it at a later time, making it simple to pick up where they left off. Users will be able to curate their own music collection with the add and delete audio files feature, which will allow them to add new audio files to their playlist and delete files they no longer want. Users will be able to find specific audio files by title, making it simple to find a specific song quickly. Overall, the new "Audio Player V2" class will provide a more versatile and user-friendly audio playback experience, allowing users to organise and access their music more easily.

**Code;**

#include <iostream>

#include <vector>

#include <algorithm>

#include <fstream>

#include<string>

using namespace std;

class AudioFile {

public:

string name;

string artist;

string album;

int duration;

AudioFile(string name, string artist, string album, int duration) :

name(name), artist(artist), album(album), duration(duration) {}

};

class AudioPlayerV2 {

private:

vector<AudioFile> playlist;

int currentSong;

public:

AudioPlayerV2() : currentSong(0) {}

void Play() {

if (playlist.empty()) {

cout << "Error: Playlist is empty" << endl;

return;

}

cout << "Playing: " << playlist[currentSong].name << " by " << playlist[currentSong].artist << endl;

}

void Next() {

if (playlist.empty()) {

cout << "Error: Playlist is empty" << endl;

return;

}

currentSong = (currentSong + 1) % playlist.size();

Play();

}

void Previous() {

if (playlist.empty()) {

cout << "Error: Playlist is empty" << endl;

return;

}

currentSong = (currentSong - 1 + playlist.size()) % playlist.size();

Play();

}

void Shuffle() {

if (playlist.empty()) {

cout << "Error: Playlist is empty" << endl;

return;

}

random\_shuffle(playlist.begin(), playlist.end());

Play();

}

void SavePlaylist(const string& filename) {

if (playlist.empty()) {

cout << "Error: Playlist is empty" << endl;

return;

}

ofstream file(filename);

for (const auto& song : playlist) {

file << song.name << "," << song.artist << "," << song.album << "," << song.duration << endl;

}

cout << "Playlist saved as " << filename << endl;

}

void LoadPlaylist(const string& filename) {

ifstream file(filename);

if (file) {

string line;

while (getline(file, line)) {

int pos = line.find(",");

string name = line.substr(0, pos);

line.erase(0, pos + 1);

pos = line.find(",");

string artist = line.substr(0, pos);

line.erase(0, pos + 1);

pos = line.find(",");

string album = line.substr(0, pos);

line.erase(0, pos + 1);

int duration = stoi(line);

AudioFile song(name, artist, album, duration);

playlist.push\_back(song);

}

cout << "Playlist Loaded Successfully" << endl;

}

else {

cout << "Error: File not found" << endl;

}

}

void AddAudioFile(string name, string artist, string album, int duration) {

AudioFile song(name, artist, album, duration);

playlist.push\_back(song);

cout << "Audio file added successfully" << endl;

}

void DeleteAudioFile(string name) {

int index = -1;

for (int i = 0; i < playlist.size(); i++) {

if (playlist[i].name == name) {

index = i;

break;

}

}

if (index == -1) {

cout << "Error: Audio file not found" << endl;

}

else {

playlist.erase(playlist.begin() + index);

cout << "Audio file deleted successfully" << endl;

}

}

void SearchAudioFile(string name) {

int index = -1;

for (int i = 0; i < playlist.size(); i++) {

if (playlist[i].name == name) {

index = i;

break;

}

}

if (index == -1) {

cout << "Error: Audio file not found" << endl;

}

else {

cout << "Audio file found: " << playlist[index].name << " by " << playlist[index].artist << endl;

}

}

};

int main() {

AudioPlayerV2 player;

player.AddAudioFile("song1", "artist1", "album1", 180);

player.AddAudioFile("song2", "artist2", "album2", 210);

player.AddAudioFile("song3", "artist3", "album3", 150);

player.Play();

player.Next();

player.Previous();

player.Shuffle();

player.SavePlaylist("playlist.txt");

player.LoadPlaylist("playlist.txt");

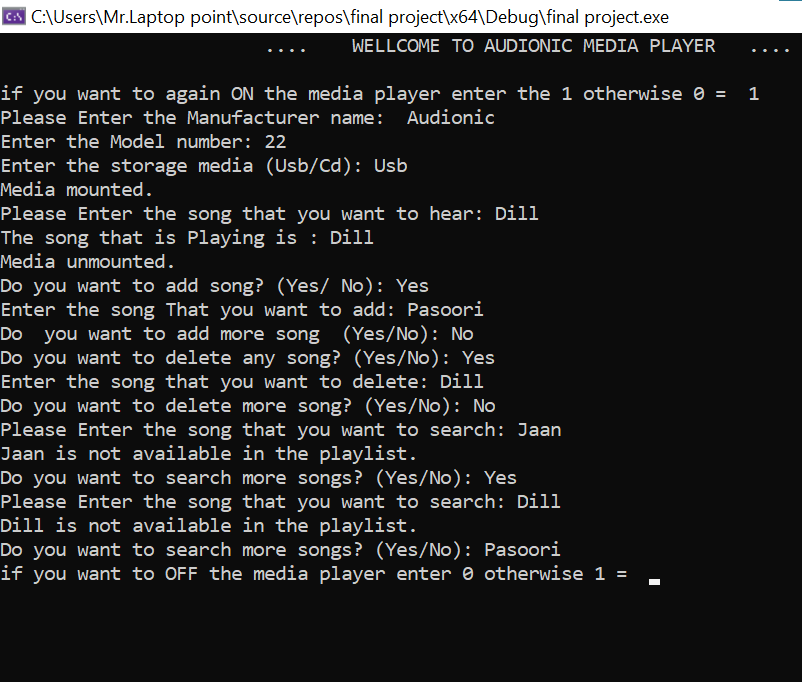
player.SearchAudioFile("song2");

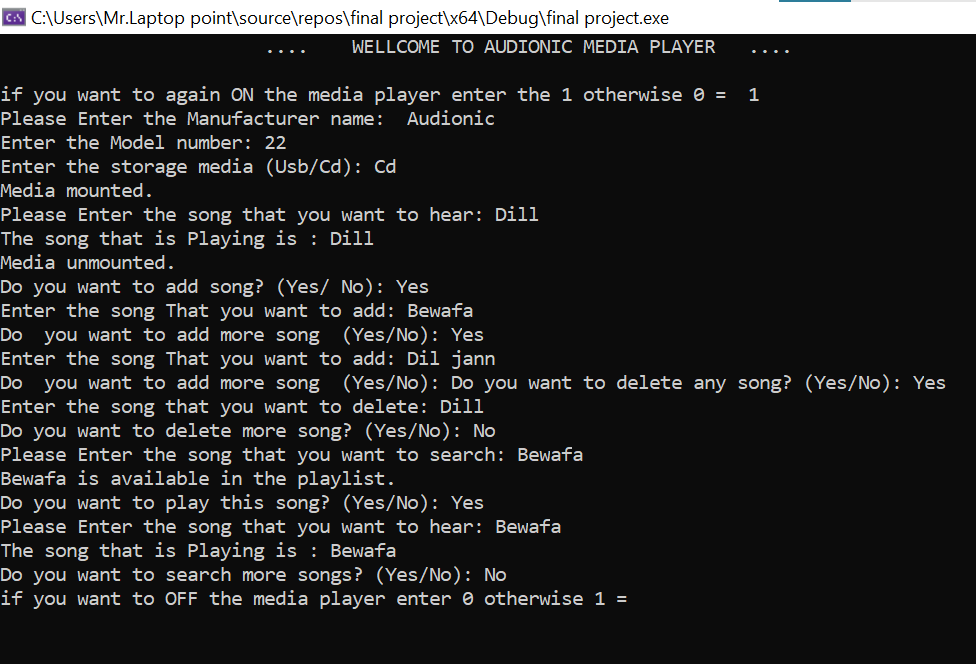
player.DeleteAudioFile("song2");

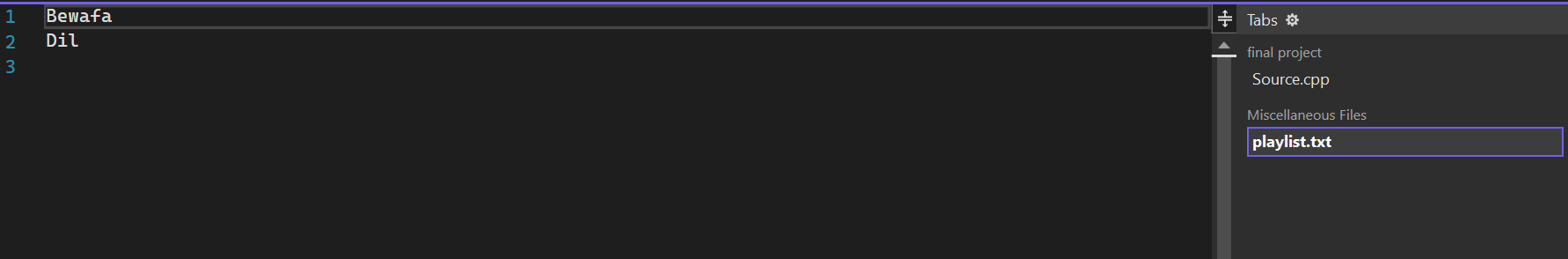
return 0;

}

**Output**

****

****

****

**Conclusion:**

We conclude that the "Audio Player" class has been expanded to include new functionalities as the "Audio Player v2" class. The updated class can now do things like shuffle the current playlist, save and load playlists, add and delete audio files from the playlist, and search for specific audio files. These new features give the in-car audio player more versatility and convenience, making it a more powerful and user-friendly device. Overall, the "Audio Player v2" class is a significant improvement over the original "Audio Player" class and demonstrates how object-oriented programming can be used to improve the functionality of a software application.