User Manual

Media Player

This will be your guide through Team 20's media player, it will help you use it the correct way to avoid any unintended behavior and will also explain how it works

Team 20

Team Members:

Ahmed Mohamed Elsheikh

21P0109

Mazen Ahmed Galal

21P0103

Kareem Ahmed Samir

21P0096

Youssef Mahmoud Hassan

21P0130

Omar Ahmed Salah Ahmed

2100790

ASU

Established 1950
Computer Engineering
Faculty of Engineering Ain Shams

CSE131 Computer Programming

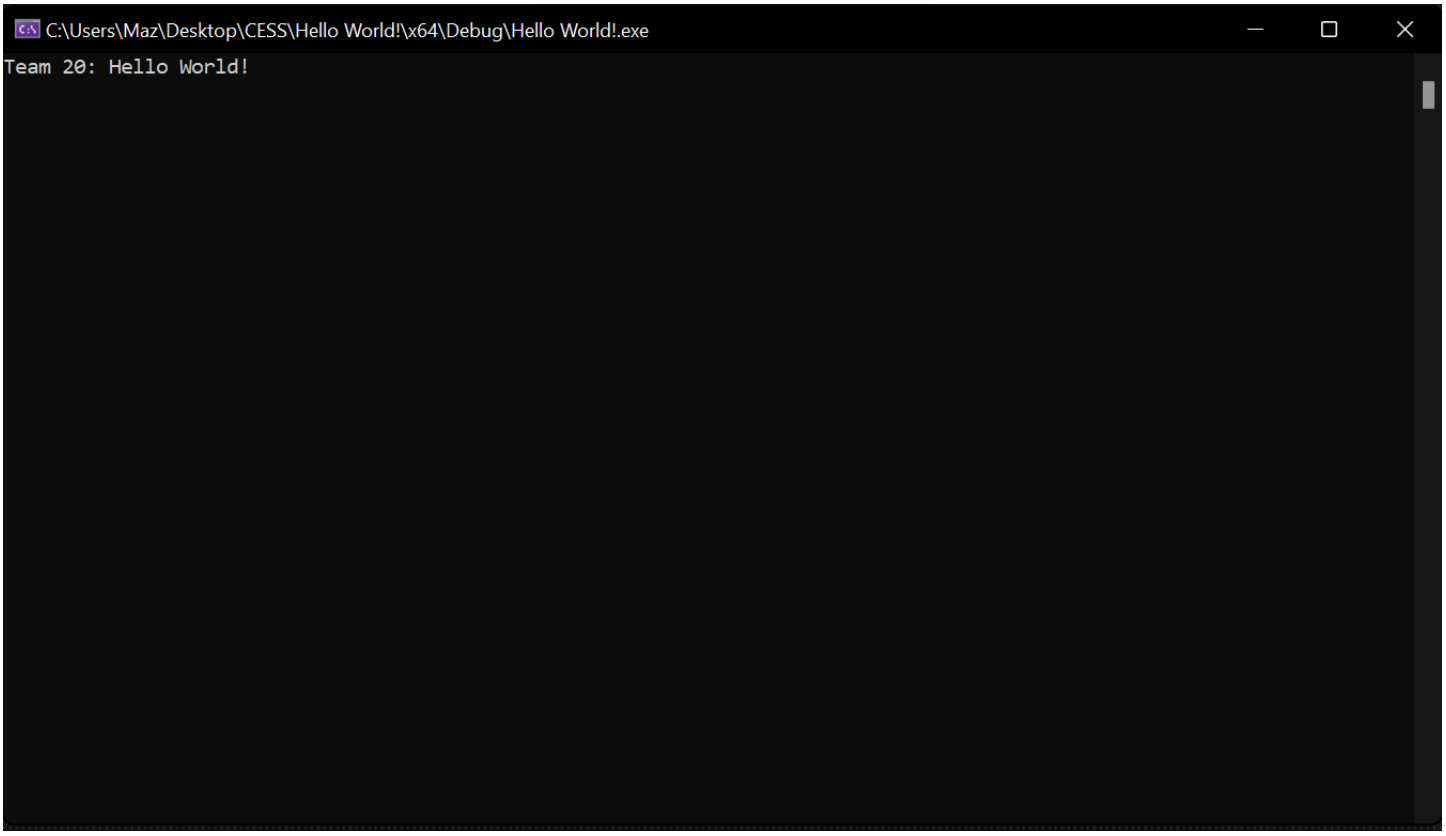
Major task

Index

Chapter 1: how to use the program

Chapter 2: how the program works

CHAPTER 1: HOW TO USE THE PROGRAM



0-0 What is this program:

This is a media player with the ability to add any song from your computer as long as you have the path to the folder that holds your music files, it accepts almost all types of audio files with only a few limitations and can play them all with full playlist controls.

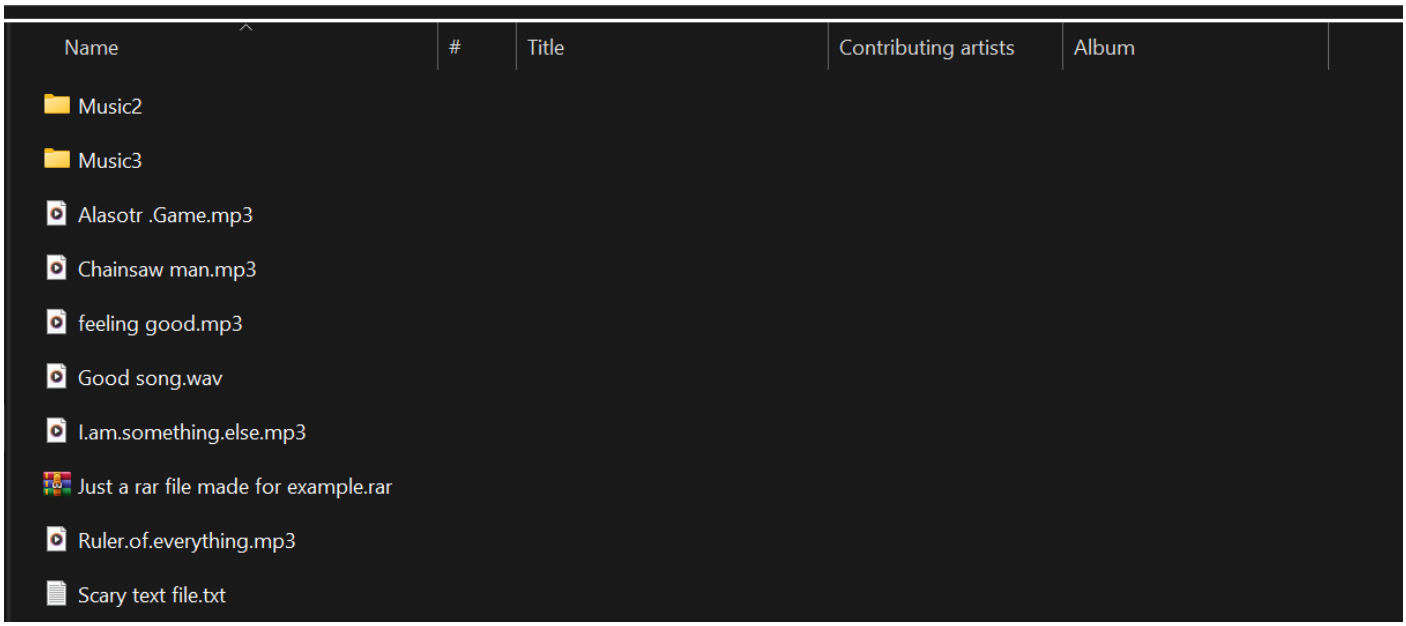
This program makes a playlist inside of it so no need to worry about where the files are on your computer

0-1 Acceptable folder and file names:

As of right now the program only works with English file names and folders, The program has been adjusted to accept spaces and dots in the file and folder names as valid inputs.

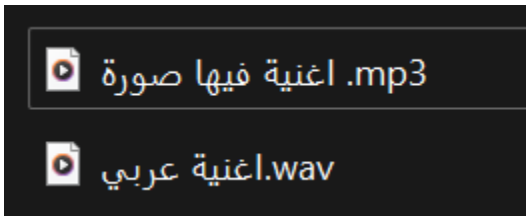
!!!However Arabic and other languages that uses different characters from English are not accepted as a valid file name or folder!!!

Acceptable:



All of these are examples of acceptable files and folders in your music folder
Our program can parse and find the paths of the songs you have in there without changing their location order or editing anything in the folder.

Not acceptable:



0-2 Limitations:

Sadly there are a few limitations to how our program works, such as that !!!if the audio file has a thumbnail it will not work!!! this is because of the fact we use mciSendString for it's versatility and computability with our vision, but it comes at that one price of not being able to use songs that have cover art.

1-0 Login:

This is a simple login function where you will be asked to enter a username and password to start the program.

```
Please enter your username:

Please enter your password:
```

Keep in mind you have 5 failed attempts before the program closes and you would have to start it again.

```
You have made too many attempts!
```

Check 1-1 for accounts.

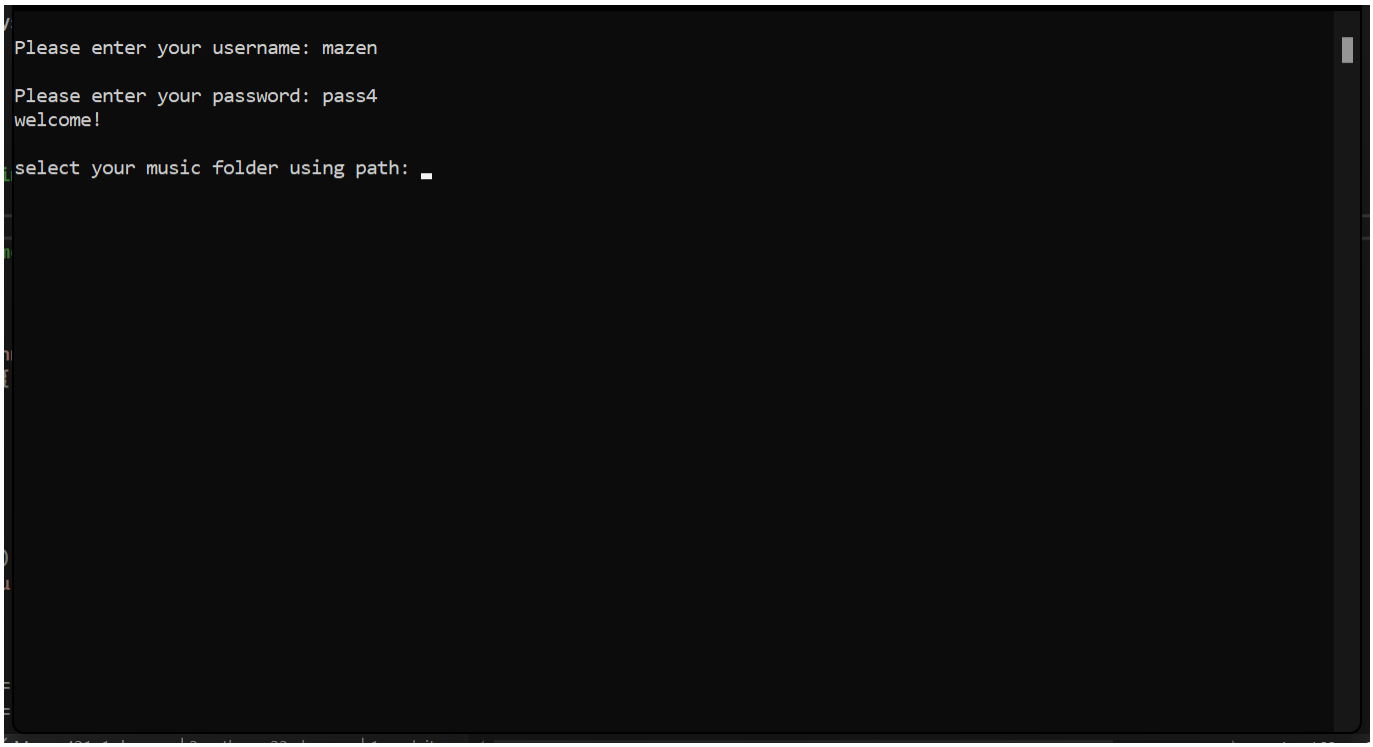
1-1 Saved Accounts:

Here is a table with the accounts you can use to use the program.

Accounts	
Username	Password
ahmed	pass1
youssef	pass2
omar	pass3
mazen	pass4
kareem	pass5

!!!MAKE SURE TO USE SMALL LETTERS NOT CAPITAL!!!

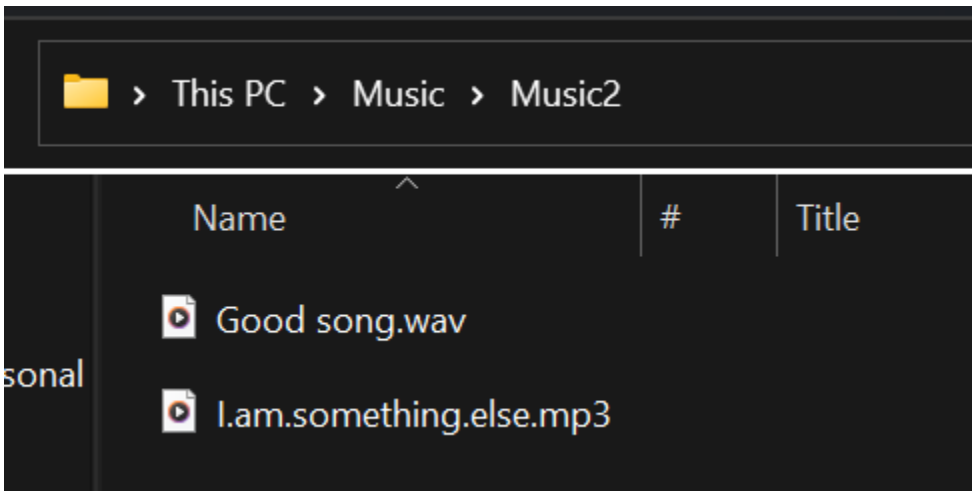
2-0 Folder Path:



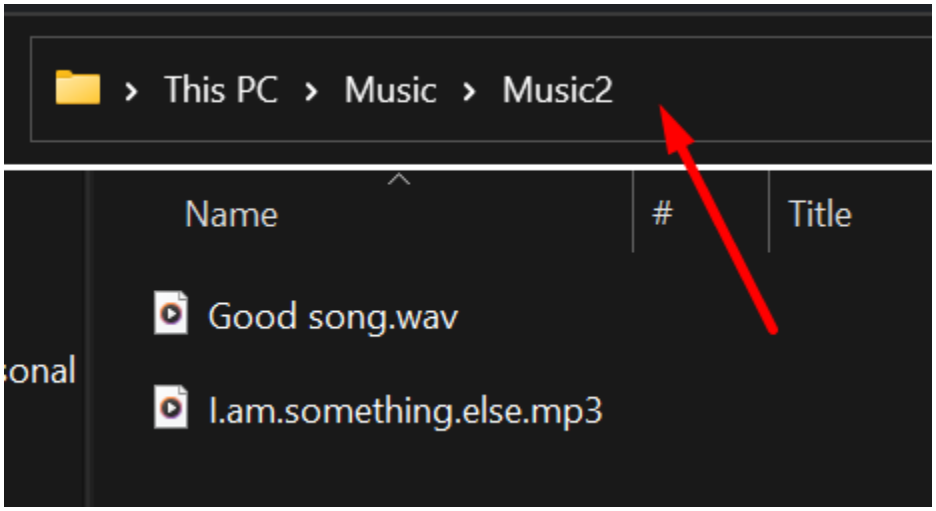
Our program runs on search the insides of a folder and grabbing all the songs inside to add it to the playlist, that means you are not limited to the songs given as example for you! You can add any song from your computer just by entering the folder path go to section 2-1 and 2-1-1 for more info.

2-1 How to get the folder path:

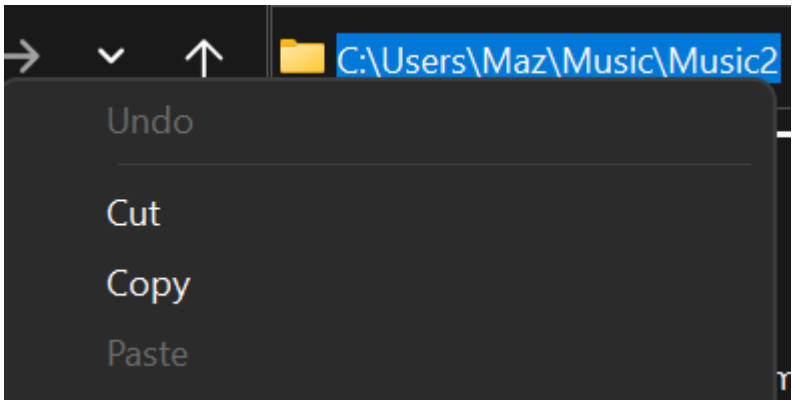
What some might think hard is actually quite easily, first browse to where your songs are



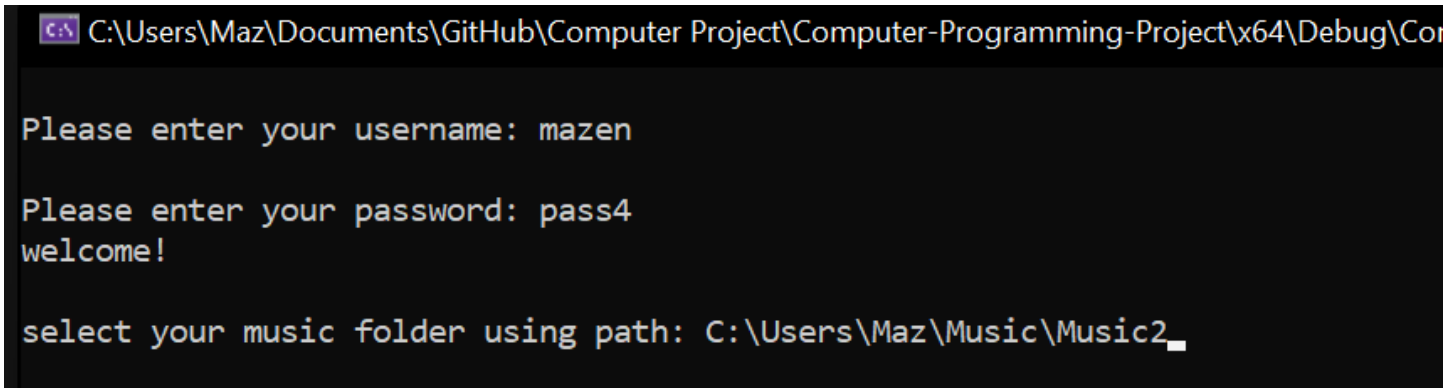
Then click on the bar right here



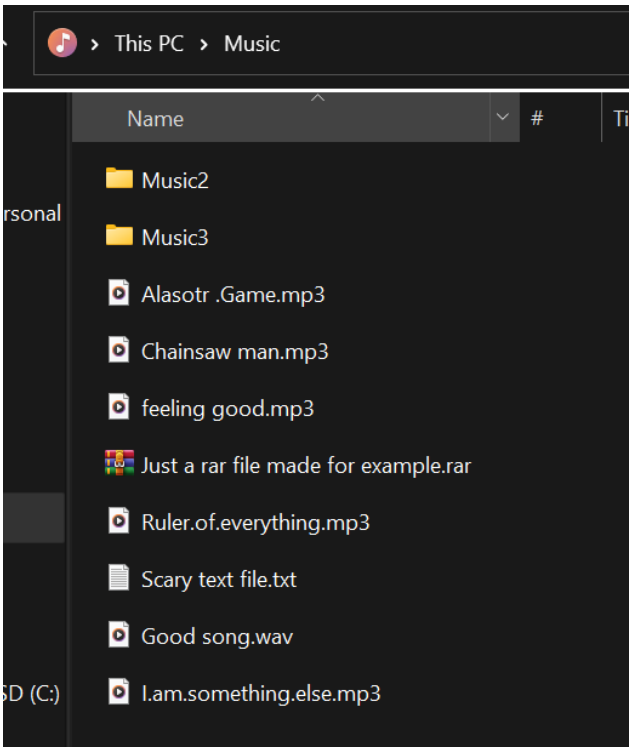
Then copy the highlighted text with CTRL + C or right click -> copy



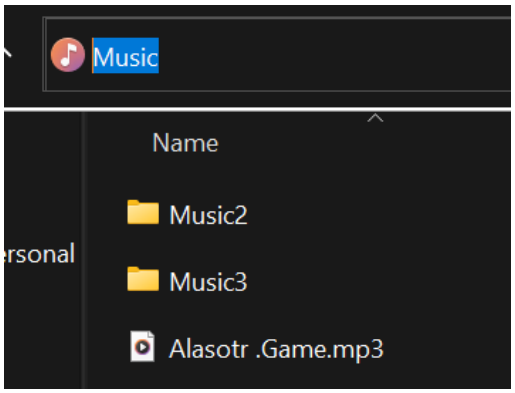
Go back to the program and just right click to put the copied path into it



What if your folder looks like this???

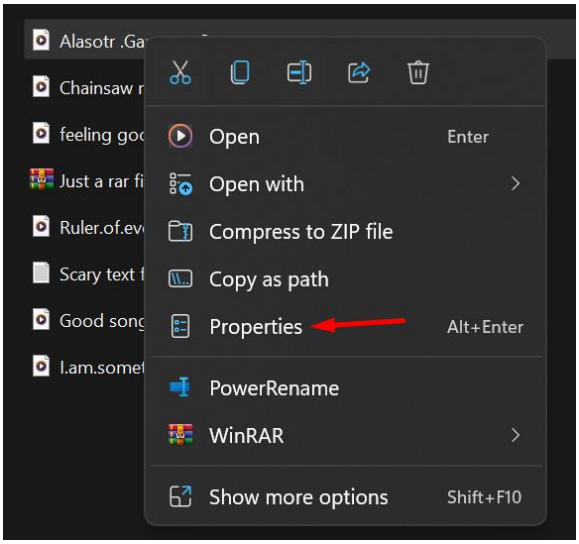


Tapping the bar up top will not give you the link

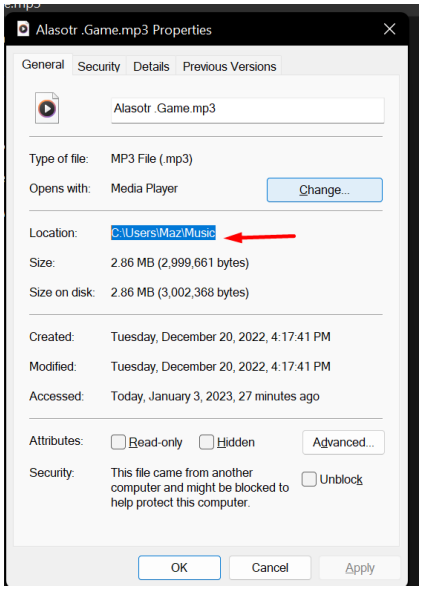


So do this method instead:

Right click one of the songs and hit properties

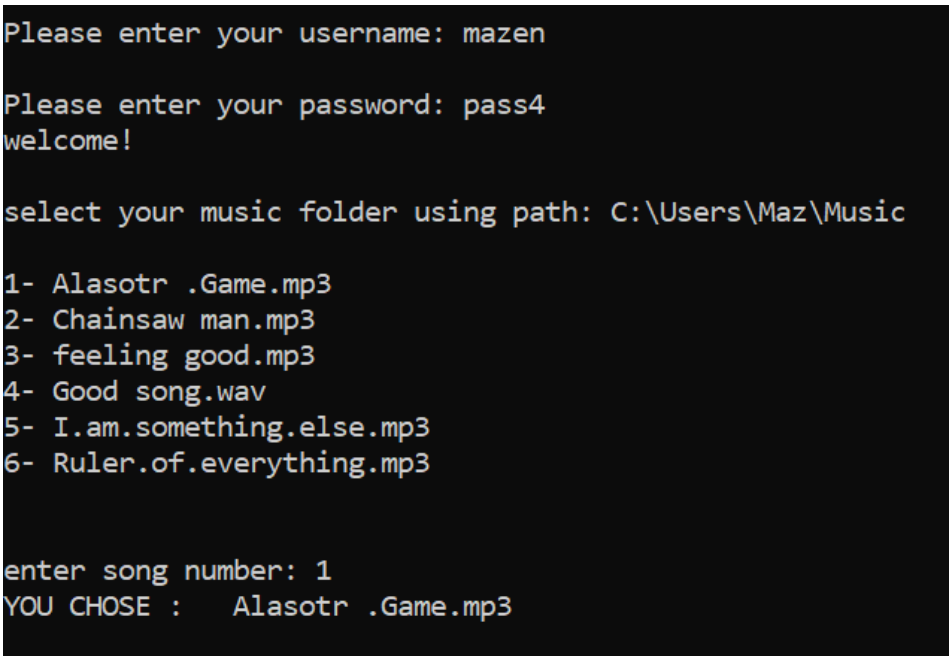


Then copy the “location” to the program and it will work the same



2-1-1 Choose one of the songs to start with:

After you enter the path then hit the key “ENTER” you have created a playlist with the songs in the folder, you will be given a choice, as to where to start the playlist from, enter any number to choose that as the beginning of the playlist.



2-2 Acceptable folder and file names

In the area to enter the path you must ensure your path is a correct and valid windows path, if you follow the steps shown in 2-1 you should have no issue but here is a few common issues you might run into

Issue 1:

!!!NOT ENTERING A VALID PATH!!!

This the path is a variable some users may enter path that is not valid so it will crash the program, so ensure you enter a valid path

Issue 2:

Entering a folder path that doesn't have a song

You may enter a folder path that has no songs, if that happens just enter 1 and then enter and it will default to the program running where you can add songs as you wish in the playlist control

Issue 3:

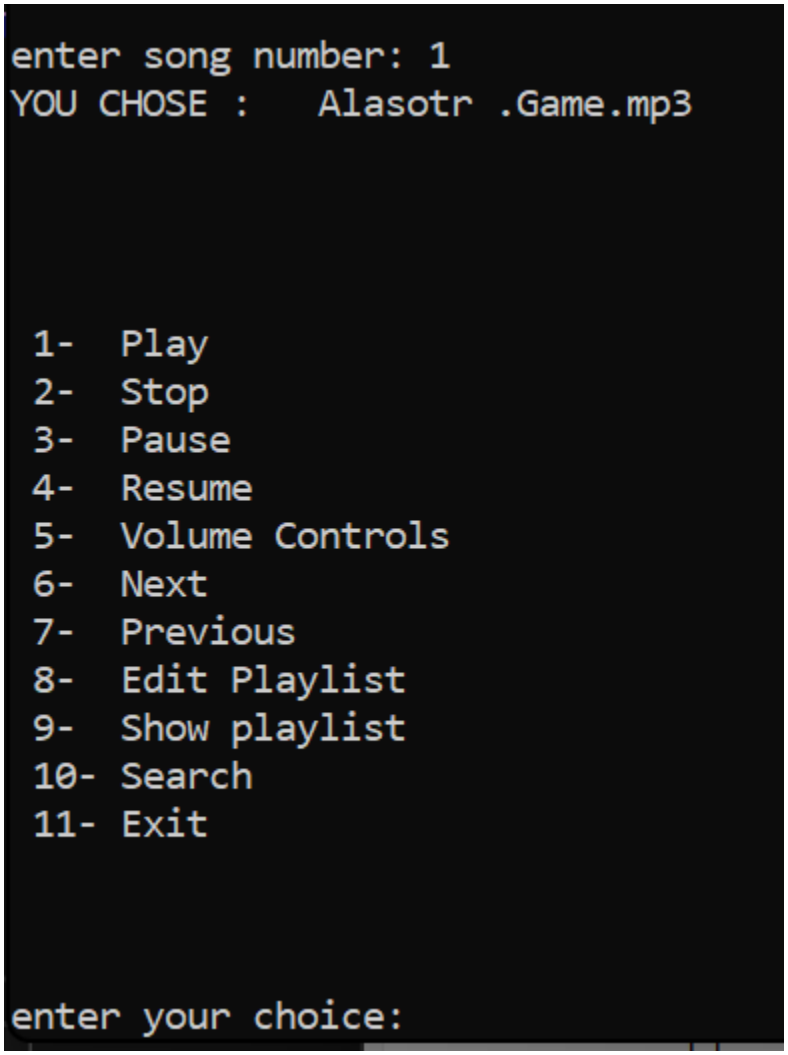
!!!Hitting enter before right clicking to paste the path into the console or left clicking somewhere on the console!!!

This issue is because of the nature of window's console applications, if you do any of these two you will have to restart the program or enter the path manually which would be too much work

Fix: Type something then delete it then right click

3-0 Media Player:

If you have done everything successfully so far you will be met with this



This is the media player each number has a function attached to it where you can use each to browse and enjoy your music playlist, Enter your choice

3-1 Play/Stop:

These are your play and stop buttons, be careful the stop button will reset the song from the start

```
Playing...: Alasotr .Game.mp3
```

The songs will repeat after they are done unless you hit next

3-2 Pause/Resume

Normal Pause and Resume one will stop the music and one will resume it from where it stopped

```
Pausing...:Alasotr .Game.mp3
```

```
Resuming...:Alasotr .Game.mp3
```

3-3 Volume controls:

This will change the program volume not the system volume so no need to worry about fixing your systems volume when using this, the default value is 5 while min is 0 and max is 10 anything out of the 10 range does nothing so it was locked for ease of use

```
Current audio: 5


 1- Control volume
 2- Exit Volume Controls

pick an option: 1
```

```
enter your desire volume from 0 to 10, default value 5
0


Current audio: 0
```

3-4 Next/Previous:

Note: Songs don’t auto next

This will move to the next song in the playlist or the song before the one currently playing, if you go next from the last song it goes to the start while if you go to the previous song on the first song it will play the last in the playlist

3-5 Edit playlist

This program runs on making a custom playlist for you to enjoy all the freedom to control it as you wish, therefore we added an option to edit the playlist as much as you want with 3 features starting with….

```
1- Choose a folder to make as a playlist

2- Pick a track to add to the platlist from a folder

3- remove a track from the playlist

4- Exit playlist controller
```

3-5-1 Choose a New Folder:

```
1- Choose a folder to make as a playlist
```

Bored of the current and have a second folder that has songs you want to play? Choose this option and enter the path to the new folder and bam your playlist will be cleared and filled with the new songs from that folder

```
enter the path to the new folder:C:\Users\Maz\Music



1- Alasotr .Game.mp3
2- Chainsaw man.mp3
3- feeling good.mp3
4- Good song.wav
5- I.am.something.else.mp3
6- Ruler.of.everything.mp3
```

3-5-2 Add a song to the playlist

```
2- Pick a track to add to the platlist from a folder
```

Like mentioned before this program has a playlist so if you pick a folder here you add songs from it to the playlist

Ensure to enter a valid path then choose which songs to add to the playlist

```
Pick the folder with tracks you wish to add: C:\Users\Maz\Music

Songs in the folder:
1- Alasotr .Game.mp3
2- Chainsaw man.mp3
3- feeling good.mp3
4- Good song.wav
5- I.am.something.else.mp3
6- Ruler.of.everything.mp3

1- Add all the songs to the playlist

2- Pick a song to add

3- Exit folder
```

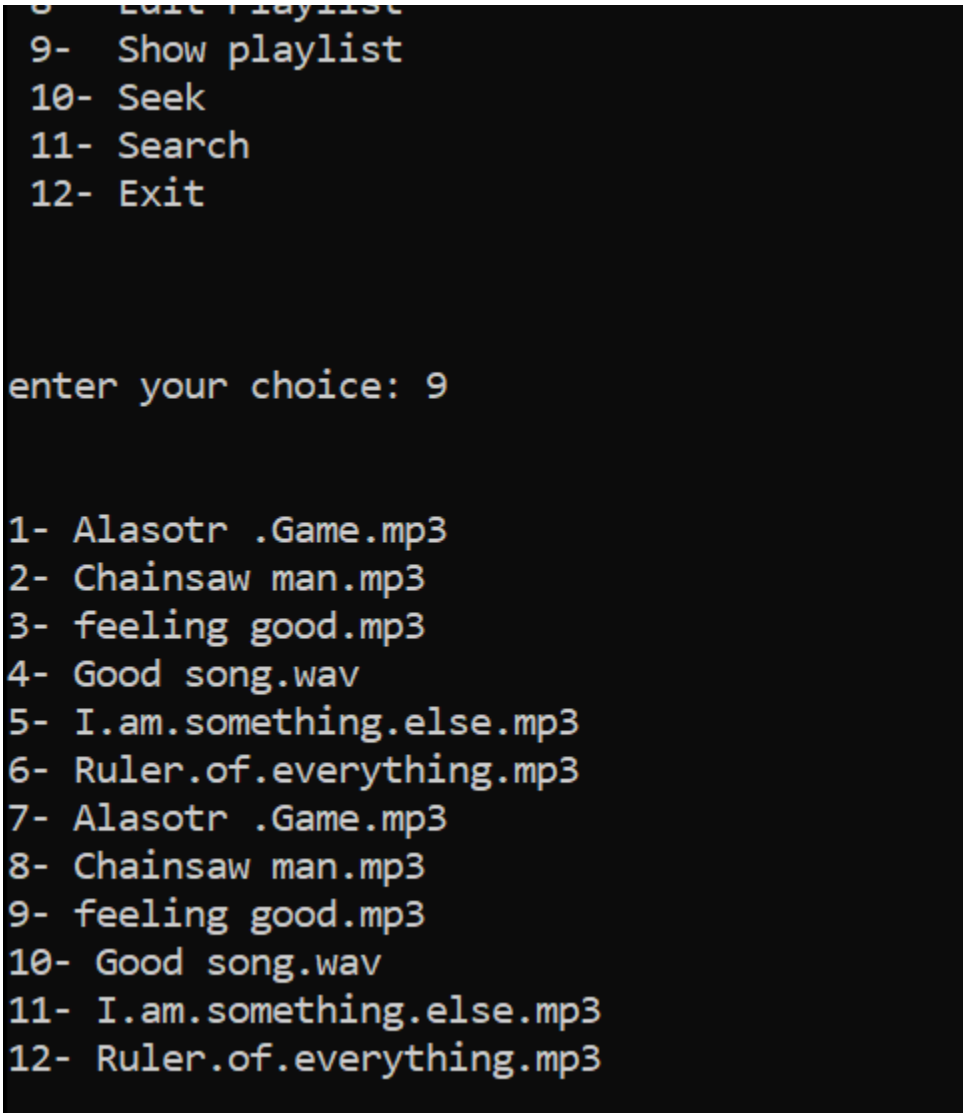
After that you are given two options, to add one song to the playlist or add all the songs in the folder to the playlist (Songs can be added more than once)

3-5-2 Remove a song from the playlist

Using this you can remove any track from the playlist easily

3-6 Show Playlist

Shows you everything in the playlist



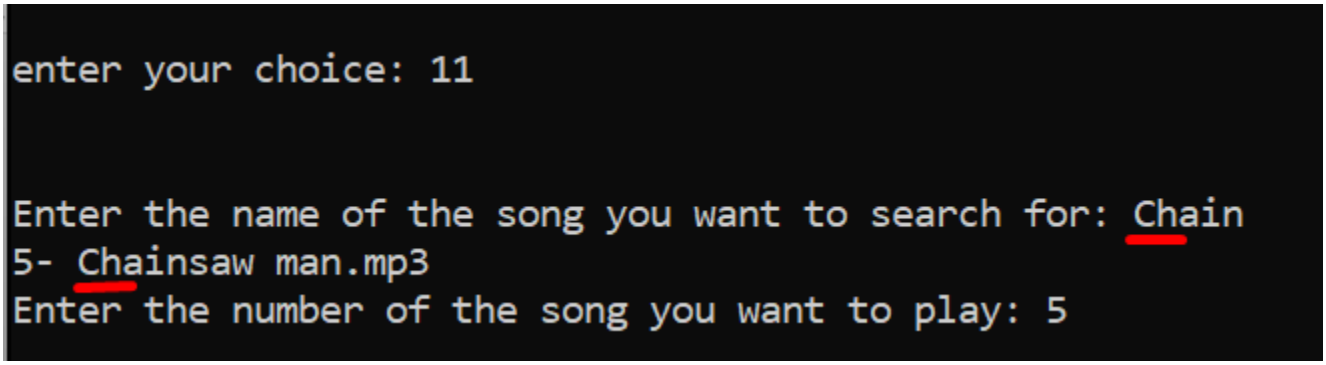
3-7 Seek:

Allows you to enter a song number from the playlist so you can jump to that song or any other song anytime you want

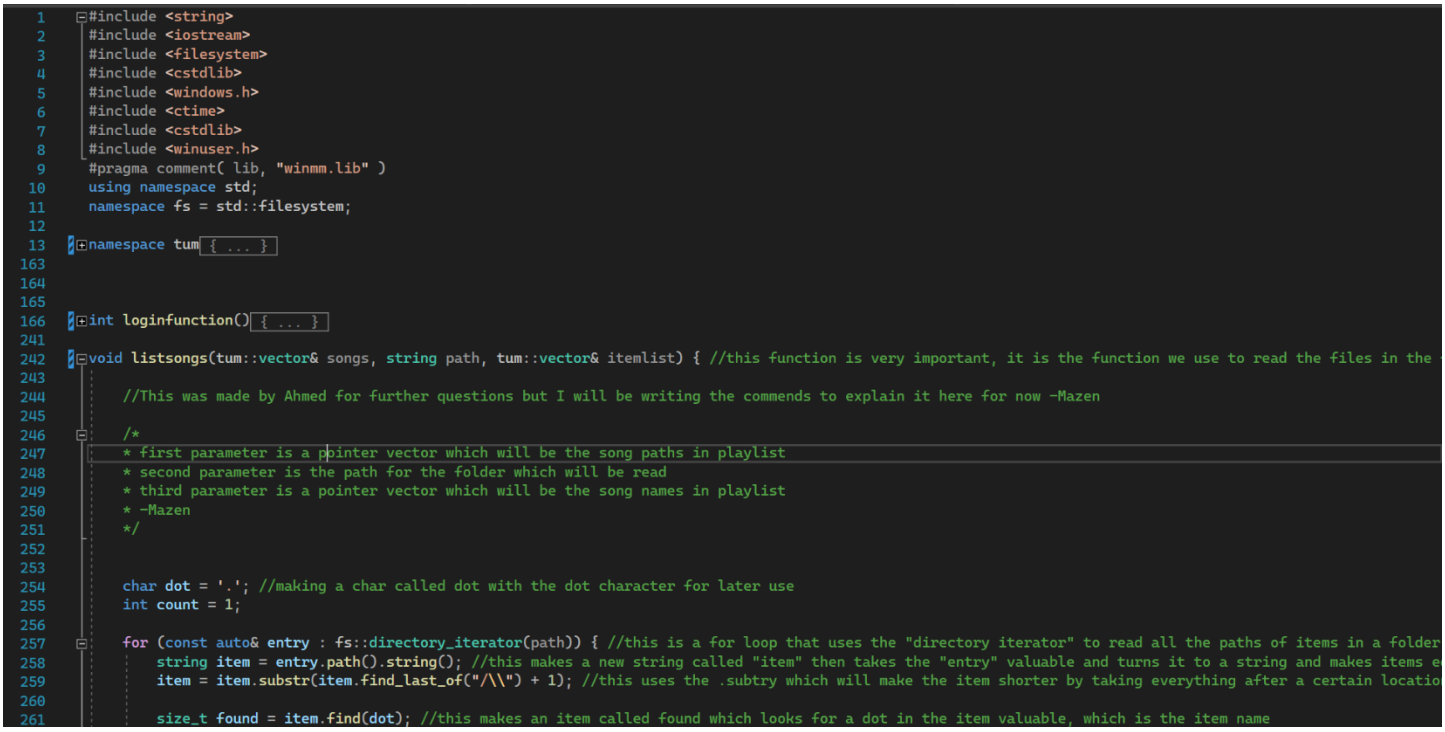
3-8 Search:

This feature allows you to search for any song using a part of the name for it

Note: this function is case sensitive so ensure if your song has a capital letter to enter it or else you won't find it



CHAPTER 2: HOW THE PROGRAM WORK



```
#include <string>
#include <iostream>
#include <filesystem>
#include <cstdlib>
#include <windows.h>
#include <ctime>
#include <cstdlib>
#include <winuser.h>
#pragma comment( lib, "winmm.lib" )
using namespace std;
namespace fs = std::filesystem;

namespace tum { ... }



int loginfunction() { ... }

void listsongs(tum::vector& songs, string path, tum::vector& itemlist) { //this function is very important, it is the function we use to read the files in the

    //This was made by Ahmed for further questions but I will be writing the commends to explain it here for now -Mazen

    /*
    * first parameter is a pointer vector which will be the song paths in playlist
    * second parameter is the path for the folder which will be read
    * third parameter is a pointer vector which will be the song names in playlist
    * -Mazen
    */


    char dot = '.'; //making a char called dot with the dot character for later use
    int count = 1;

    for (const auto& entry : fs::directory_iterator(path)) { //this is a for loop that uses the "directory iterator" to read all the paths of items in a folder
        string item = entry.path().string(); //this makes a new string called "item" then takes the "entry" valuable and turns it to a string and makes items e
        item = item.substr(item.find_last_of("/\\") + 1); //this uses the .subtry which will make the item shorter by taking everything after a certain locatio

        size_t found = item.find(dot); //this makes an item called found which looks for a dot in the item valuable, which is the item name
```

0-0 Makeshift Vectors:

So dynamic arrays, they are like normal arrays but can a expanded upon just the limit that was set to it in the code, which makes them the perfect functions to control our playlist and program, the reason

When we started the programs, we were using vectors until we were informed that it is not allowed in the project, so we improvised, we made our own tum::vector struct with it's own set of functions

0-1 Struct Vector:

The tum::Vector Struct is a struct that contains some variables and functions to simulate the standard library vectors using our own custom code and working on dynamic libraries.

Variables: Size: a variable that keeps track of the maximum size of the array, this variable dynamically changes based on the needs of the array.

Capacity: a variable that keeps track of the total capacity of the array (how full the array is), these variable updates based on the number of elements in the array.

0-2 Why Named Vector:

When we wanted to do the switch from std::vectors to our own vector we noticed that we have already used vectors in so many places that going over each one and changing it from vector to another name would be too exhausting, so we went for option two! Name our struct vector so it would register all the vectors in the program as our own type of vectors when we remove the vector library from the #include

0-3 Namespace tum:

We made struct vectors we got them working but we got an issue. How in the world do we pass a struct into a function so we can use our own vectors inside! So we came up with this solution, when we try to use vectors in the functions it looks for std::vector so we made our own namespace called tum and we passed it through and into the vectors using tum::vector to be able to get it working inside making us able to use all the code we written before with out new tum::vector

0-4 vector .functions:

Vector(): Constructor function that initializes the dynamic array
.Push(): add an item to the end of the array, size expands accordingly.
.Pop(): Remove an item from the end of the array, size shrinks accordingly.
.Erase(): Removes an item from a specific index by pushing every item in front one index step back and deleteing the empty spot.
.Getsize(): Returns capacity.
.Get(): Returns an item from a specific index.
.Clear(): Clears the whole array.
.Print(): Prints the whole array.

1-0 Login function:

First we make a string containing the names of usernames who have access to the programs • We make another string containing the passwords of every user
We use a while loop that runs when login = true and the bool login turns false if the user enter a wrong username or pasword more than 5 times
Then we use getline to take in from the user a username and password
Then we use for loops and if conditions to check whether the entered username and password match our database or not
We also check if the password entered is the right password for this specific username
If the user enters a false username or password he is told that the username or password entered are not correct and then he is asked to enter a new username and password until he tries more than 5 times

2-0 List songs:

Listsongs function is the function we use to read the files in folders and sort them out using the extension sorting we have inside. This function takes mainly 3 parameters
The First: is the reference to the tum::vector we want to put items inside
The Second: is the path to the folder that contains the songs we wish to add the paths of to the playlist
The Third: is a reference to another tum::vector where we will have the songs name in the play list put into their own separate tum::vector after it's gets pulled out of the path.
First we read all the items inside the given folder path using directory iterator and get all the paths
Second, we will do two things first read after the last dot in the path to see the extension of the file if it is .wav or .mp3 since they are the most common audio files for songs we are going to push the path to the playlist and the name to the itemlist

2-1 Directory Iterator:

The directory iterator iterates over the entry elements of a directory. It reads the paths of every file inside a folder. Disregards folders and items with non-audio extensions (.exe, .cpp, .mp4 etc . . .) by taking an extension substring that starts from after the last "." in the path.

The remaining valid paths get stored in an array for later use. And subsequently shortened to just the name of the song by using a substring that starts after the last "/". The names get pushed to a different array for storage and display.

3-0 Edit Playlist:

Using our tum::vector we are able to do a nice trick, instead of keeping the songs in the program director, we can copy the path and put it into a playlist and through .substrg() extract the name of the file and put it into a tum:vector itemlist, therefore allowing us to make our own playlist free from any folder or pre existing playlists, and we used mciSendString to do this

3-1 New Folder New Playlist:

Since we are no longer tied down with the files in one folder, we can scrub all the items in the playlist and add items from a new folder into it! Completely changing the playlisted we started with, with something new and unrelated

3-2 Add a Track to The Playlist:

Since we are using a playlist by using the directory iterator and tum:vectors we can scan files in a folder and pick one song inside to take it’s path and add it to out tum::vector playlist…why stop at one when you can add the entire folder as well! Complete playlist control

3-3 Remove a Track From the Playlist:

In our tum::playlist we made a .erase function which takes makes a new dynamic array puts everything into it except the the one item we want to erase then put everything back into our array effectively removing any item we want using the index which gives us all the freedom we want to remove items from the playlist

4-0 mciSendString:

This is the meat of our program, mciSendString, it is a very strong std library that comes with C++ that allows you to play a lot of types of media ranging from midi to audio to video, but it has a certain amount of quirks to it, first it doesn't take normal functions it takes a string of type LPCSTR as a the command to execute the functions it can perform

4-1 mcicommand function:

So we created this function called mcicommand, which makes us a variable called command that is a the fully prepared command using <string> to take the template and make it into the command we want, then we return command, we use it to make two commands for mciSendString, one is audio the other is the song we want to open

When we get command returned to us we use .c_str() to turn it from a string to LPCSTR

4-1-1 Command Path:

Here we talk more about how mciSendString open command works, the syntax goes like this "open path type type_flag alias alias_name"

Open is the command sent to mciSendString, it tells the program to open a file, then path is the path to that file on windows in this case our music file, type is another command telling the program to open the file as what exactly, in our case we choose mpegvideo because it covers a lot of audio files very easily and without much issues, then alias is just a name you give the open file so you can execute all sorts of mciSendString command on it like, Play, Stop, Pause, Resume, Close.

4-1-2 Command Volume:

In this we need to prepare another command this time the syntax is "setaudio alias_name volume to value"
set audio is the command we send to send string, alias_name is the name we gave to the file we have open so it knows which to execute this command on, volume to is another mciSendString command, and then value which is the value of the audio we want, mciSendString uses a value of 0 to 10,000 so we edited it so the input of the user would be multiplied by 100 before being sent to the command function and edited to be all set into one string

5-0 The Media Player:

Play: after the command function is executed we run the mciSendString function with the syntax “play alias_name flag”, we put the alias “songs” we defined when opening the file in the alias_name spot and as a flag we put repeat so the song repeats after it is done playing

Stop: to actually stop the file and when played again it starts from the beginning we use close instead of stop, as the mciSendString stop functions like a pause, therefore our stop case works by closing the file and opening it again so it would play from the start

Pause/Resume: Send we are using mciSendString is is easy to stop or pause the song playing by using “resume” or “pause” functions of mciSendString

Volume controls: as explained in 4-1-2 this will take an int as an input and multiply it by 100 then turn it from an int to a string and send it to the volume command function to return a fully working command to set the audio to what we want

Next: works by closing the song first then incrementing songnum with one and getting a new path from the playlist tum::vector after getting the new path we send it to the command function to return us a command to use in mciSendString

if songnum is bigger than the size of the tum::vector we set it to 1 instead so it plays the first song

Previous: is the same but we decrement songnum by one

Edit Playlist: Already explained in details in 3-0, 3-1, 3-2, 3-3 this function allows us to easily control the playlist tum::vector to edit it as much as we want removing songs adding songs or clearing the playlist all together and adding a new one

Show playlist: goes through the entire dynamic array (tum::vector) printing out all the items inside

Seek: Works by first printing out everything in the tum::vector and asking you for which song would you wish to play, when you enter a value the program takes it and equates it to songnum then calls upon the command function to return us a usable command with the new song path to use in mciSendString

Search: works by going through all the items in the playlist tum::vector looking for anything similar to what the user entered as a string and when it finds a similar item it pushes it to the searchlist tum:vector then displays it, and allows you to pick one of the songs that appeared to play the playlist from it

Exit: clears the items inside the playlists and closes the program