ACtivity 3 30019812

Programming 3

Contents

[Introduction 1](#_Toc57994864)

[Test Table 2](#_Toc57994865)

[Debugging 3](#_Toc57994866)

[Test proof 1: 3](#_Toc57994867)

[Test proof 2: 3](#_Toc57994868)

[UML Diagrams: 4](#_Toc57994869)

[Client UML 4](#_Toc57994870)

[Server UML 5](#_Toc57994871)

[Code Regions 6](#_Toc57994872)

[Client 6](#_Toc57994873)

[Server 7](#_Toc57994874)

[Functions 8](#_Toc57994875)

# Introduction

I was hired to develop a server client login where the client is able to login and access a media player. The client should then be able to make use of the functions in the media player allowing them to play songs, load songs from a csv file, sort these songs as well as save them to a new csv file.

I was asked to follow certain standards and requirements given by the client. For sorting I made use of a merge sort method, for hashing I made use of the SHA-256 method, for searching I made use of a simple Linear search method. In order for clients to login, their details had to be saved in a text file on the server’s end. I made use of Linked lists as a data structure and finally for a 3rd party library I made use of CSVReader to be able to read csv files and display them in the list box of the media player. All methods were separated by comments that explain what that method does

e.g.

//button to play the first song in the listbox

//button to find a song in the listbox and play it

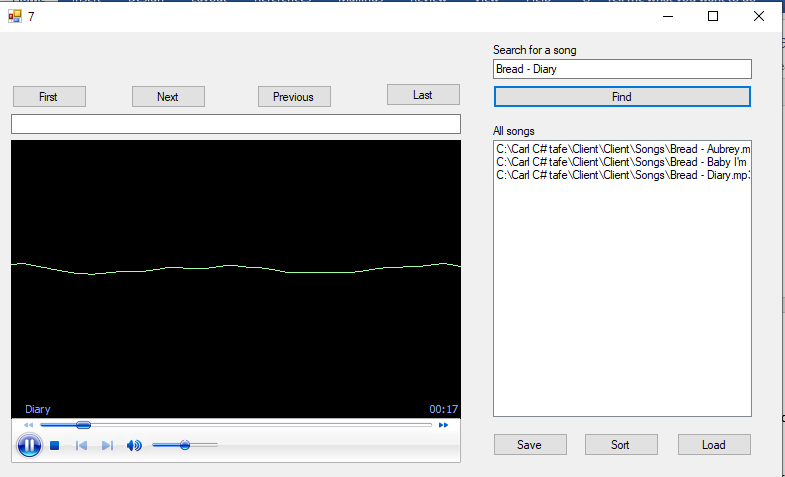
# Test Table

|  |  |  |  |
| --- | --- | --- | --- |
| What is being tested | Test | Did it pass | Where to find the code |
| Username connection | Ref 1 | Pass |  |
| Password connection | Ref 2 | Pass |  |
| Play first song | Ref 5 | Pass |  |
| Search song in the list | Ref 9 | Pass |  |
| Play Next song | Ref 6 | Pass |  |
| Play Previous song | Ref 7 | Pass |  |
| Play Last song | Ref 8 | Pass |  |
| Save to csv | Ref 10 | Pass | Being called out at |
| Load from csv file | Ref 3 | Pass |  |
| Sort the list in the listbox | Ref 4 | Pass | Being called out at |
| Server receives hashing for password | Ref 2 | Pass |  |

# Debugging

## Test proof 1:

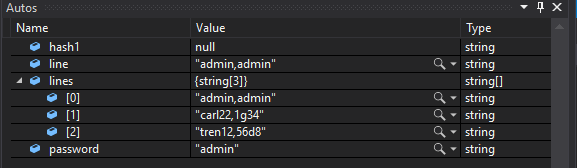
Testing to see if the find button does selects the song from the list and plays it?



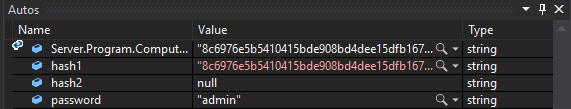
As you can see, I’m searching for the third song in the list box.

## Test proof 2:

Testing to see if the hash code is the exact password it is connected to and not linked to a different one?



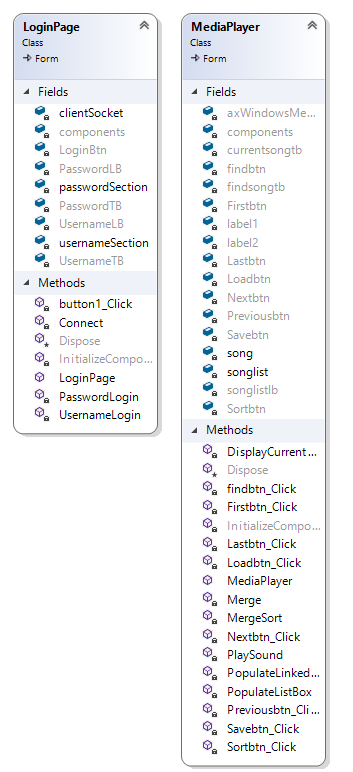
It selects the password “admin” and steeping further into it, it compares to see if the hash for admin in the login from the client end matches the hash for admin in the Users text file.



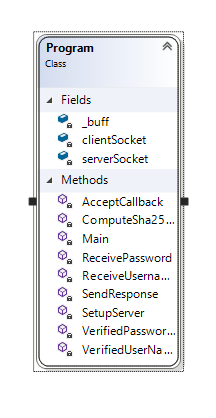
Therefore, I am able to say that the test is true.

# UML Diagrams:

## Client UML



## Server UML



# Code Regions

## Client

|  |  |
| --- | --- |
| Regions | Line Region found on |
| private void Savebtn\_Click(object sender, EventArgs e)  {} | 139 |
| private void Findbtn\_Click(object sender, EventArgs e)  {} | 118 |
| private void Lastbtn\_Click(object sender, EventArgs e)  {} | 102 |
| private void Previousbtn\_Click(object sender, EventArgs e)  {} | 85 |
| private void Nextbtn\_Click(object sender, EventArgs e)  {} | 68 |
| private void Firstbtn\_Click(object sender, EventArgs e)  {} | 52 |
| private void Sortbtn\_Click(object sender, EventArgs e)  {} | 145 |
| private void PopulateLinkedList()  {} | 156 |
| private static List<string> MergeSort(List<string> copy)  {} | 169 |
| private static List<string> Merge(List<string> left, List<string> right)  {} | 195 |
| private void Loadbtn\_Click(object sender, EventArgs e)  {} | 231 |
| private void PopulateListBox()  {} | 237 |
| private void PlaySound(string playsong)  {} | 252 |
| private void DisplayCurrentSongTitle()  {} | 267 |
| public void ReadCsv()  {} | 297 |
| public void SaveCsv()  {} | 316 |
| public class NewSongs  {} | 289 |
| public class SongMap : ClassMap<Song>  {} | 280 |
| public class Song  {} | 274 |
| public MediaPlayer()  {} | 39 |
| public partial class MediaPlayer : Form  {} | 37 |

## Server

|  |  |
| --- | --- |
| Regions | Line Region found on |
| class Program  {} | 12 |
| static void Main(string[] args)  {} | 18 |
| private static void SetupServer()  {} | 26 |
| private static bool VerifiedUserNames(string username)  {} | 36 |
| private static bool VerifiedPasswords(string password)  {} | 53 |
| private static void AcceptCallback(IAsyncResult AR)  {} | 76 |
| private static void ReceiveUsername(IAsyncResult AR)  {} | 89 |
| private static void ReceivePassword(IAsyncResult AR)  {} | 117 |
| private static void SendResponse(IAsyncResult AR)  {} | 142 |
| private static string ComputeSha256Hash(string rawData)  {} | 148 |

# Functions

Continue Button:

Once the user inputs the user name, they are then allowed access to input their password and then login.

Login:

This allows the user to access the media player and play songs.

Hashing:

The server receives the user login details and hash code connected to the password.

Sorting:

If the list box is unsorted, by clicking the sort button the user is able to sort the songs in alphabetical order.

Search:

If the list box contains quite a bit of songs, the user doesn’t have to scroll through each of them, instead he is able to type it in the textbox and click “find” once the song is found, the media player will then play that song.

First:

Using this button, the user is able to play the first song in the list box

Save:

This allows the user to make use of the 3rd party library to save the songs in the list box to a new csv file called “NewSongs.csv”.

Load:

The user is able to populate the songs1.csv file and play the songs that are in the file.

Next:

User is able to play the next song to the one currently being played

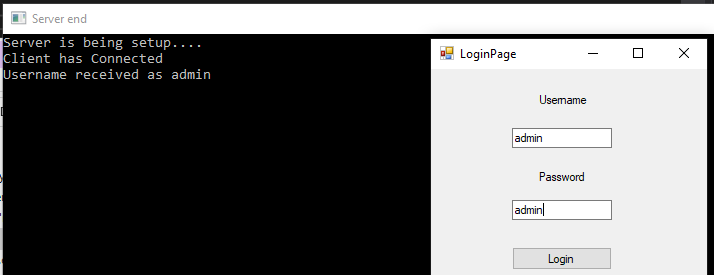
Previous:

User is able to play the previous song to the one currently being played

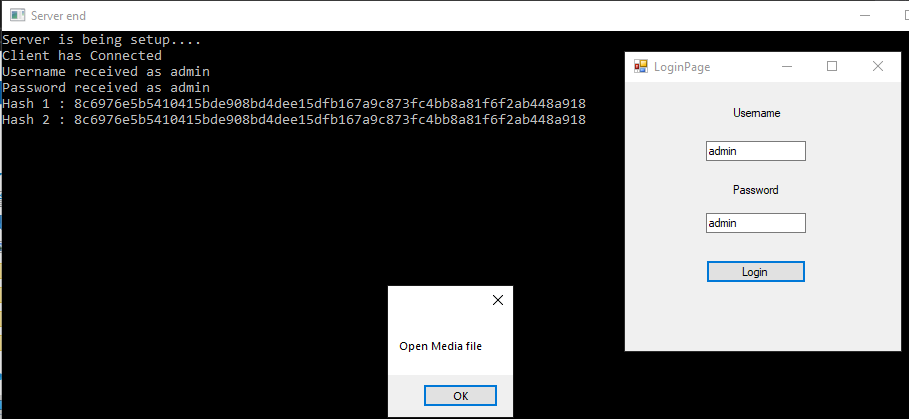
Last:

User is able to play the last song in the list box

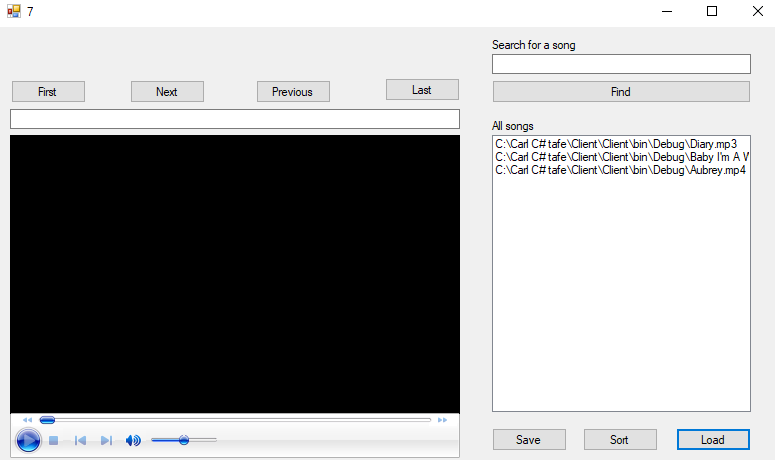
Reference images



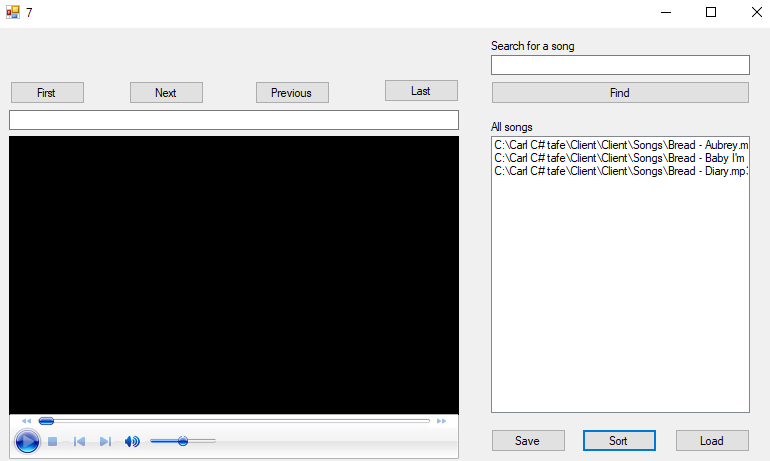
Ref 1



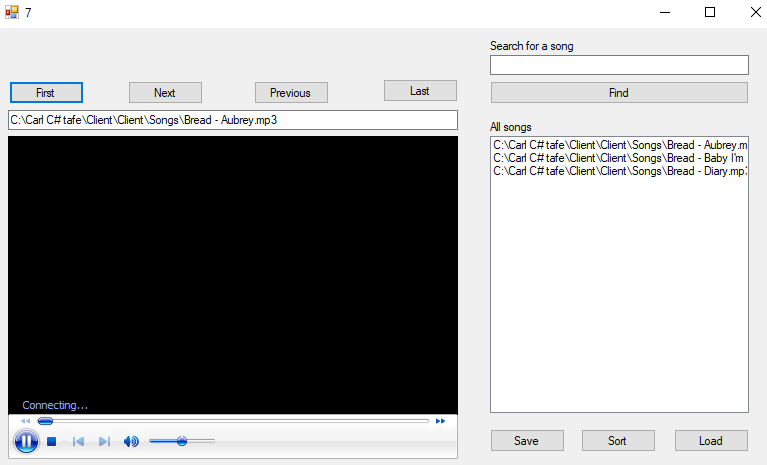
Ref 2



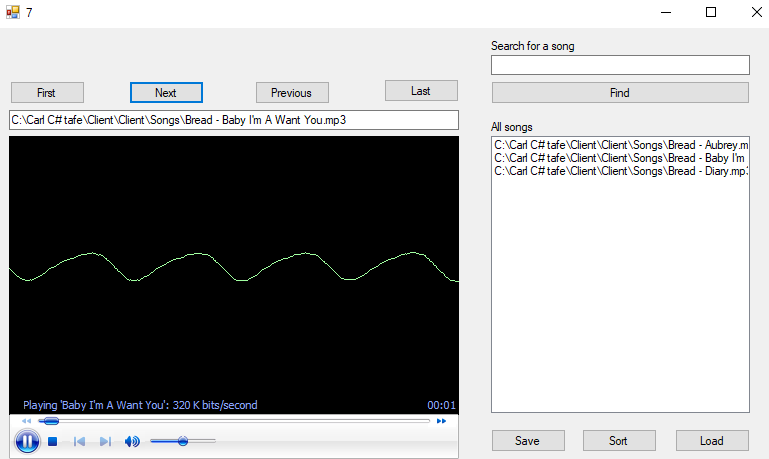
Ref 3



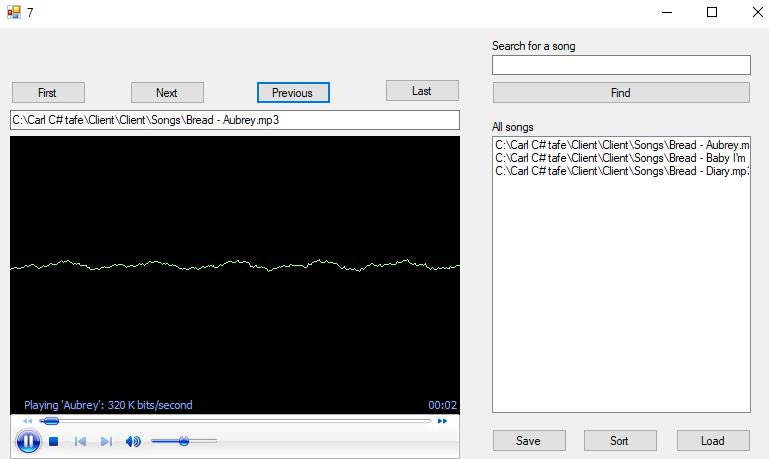
Ref 4

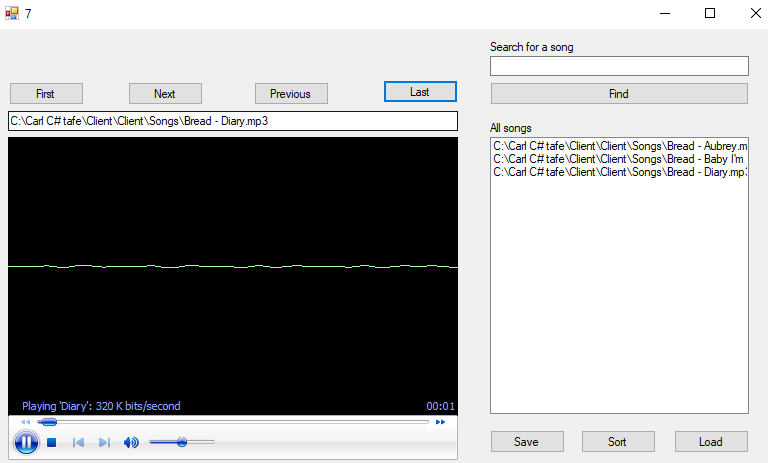


Ref 5

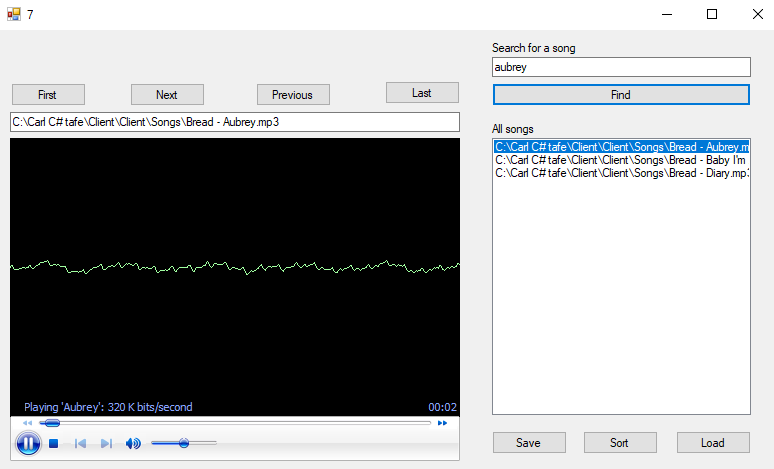


Ref 6

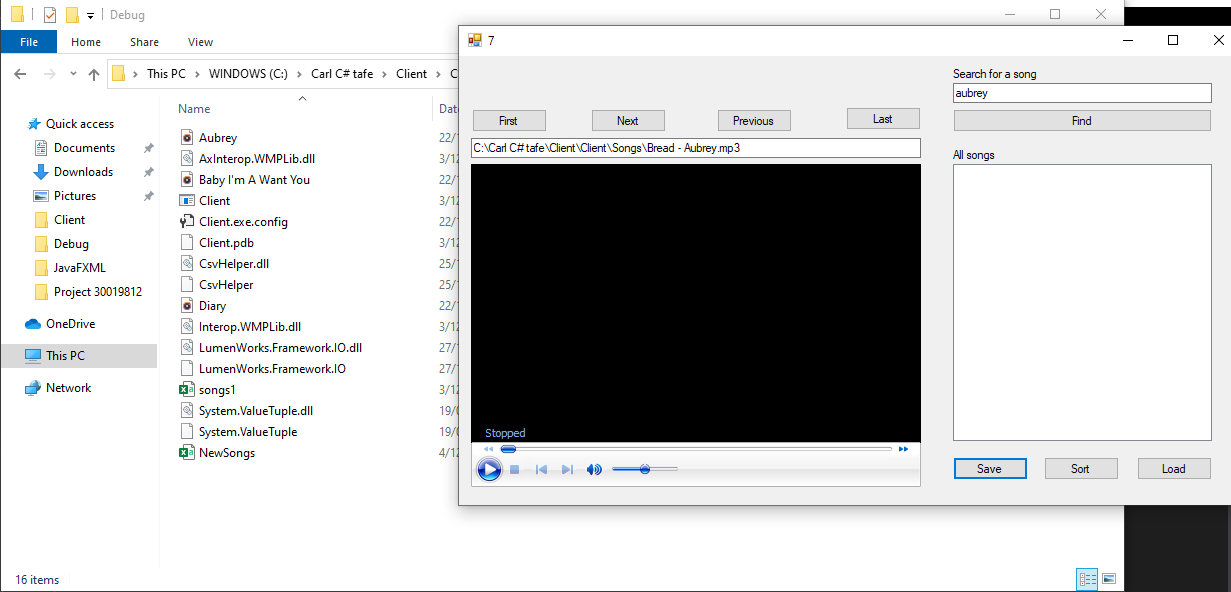
Ref 7



Ref 8



Ref 9



Ref 10