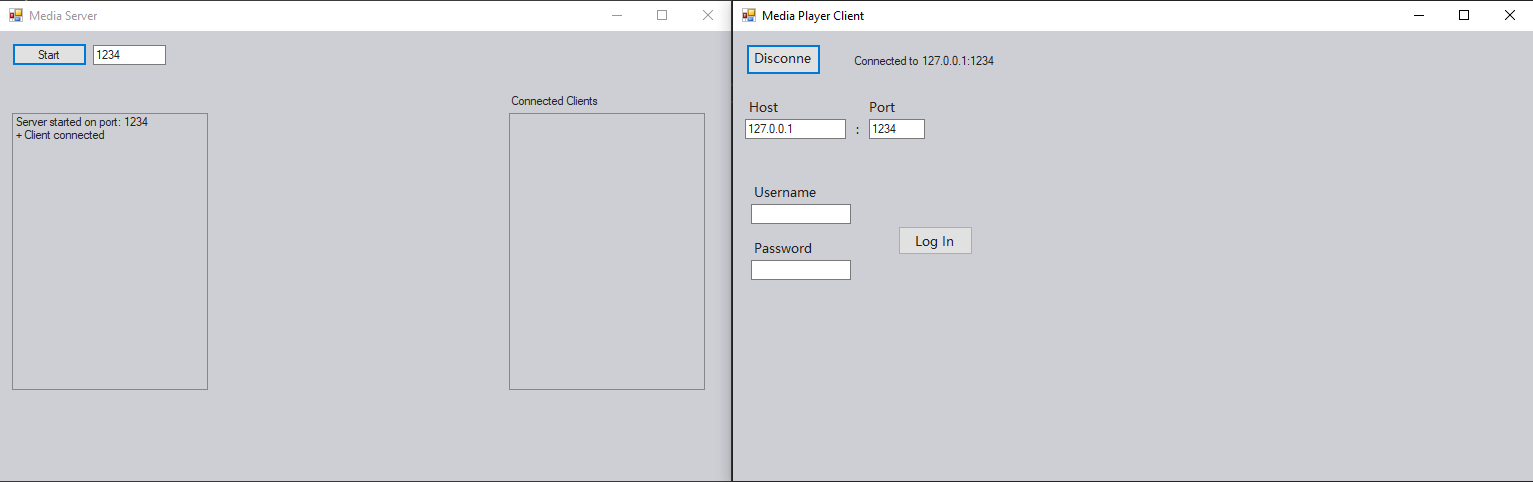
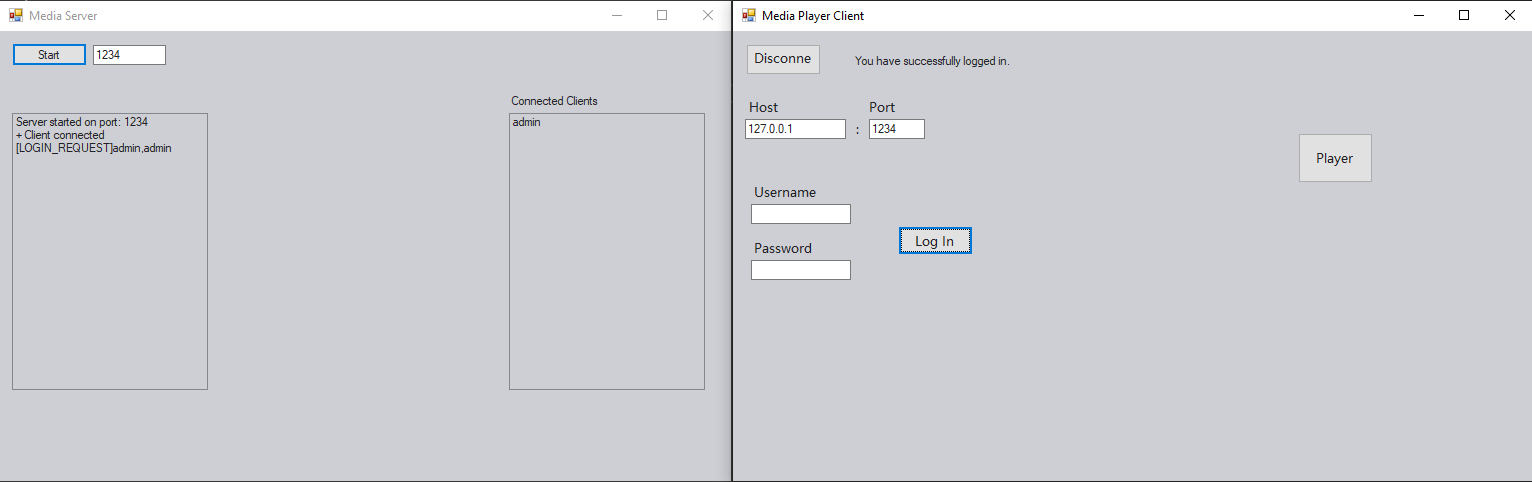
# AT3 Project - *by**Sean Boaden*

## Multiple-Client/Server Socket Application with Media Player

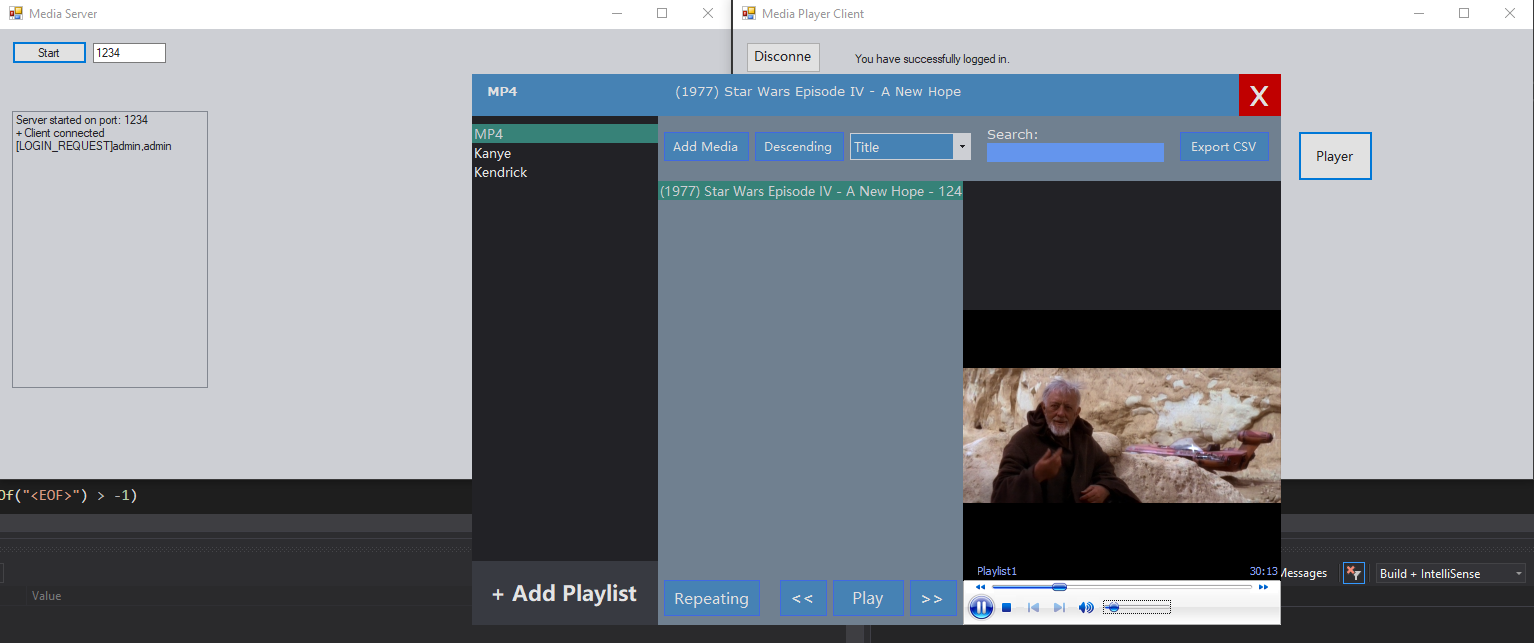
Start Server by clicking Start. Connect the Client by clicking Connect. The port number must be free.



Default admin account: user = **admin**, pass = **admin**.

Upon successful login, you can access the Player module.

Since Windows Media Player control is used, user can double click to enter full screen mode.



Contents

[AT3 Project - *by**Sean Boaden* 1](#_Toc43903428)

[Multiple-Client/Server Socket Application with Media Player 1](#_Toc43903429)

[Source Control 3](#_Toc43903430)

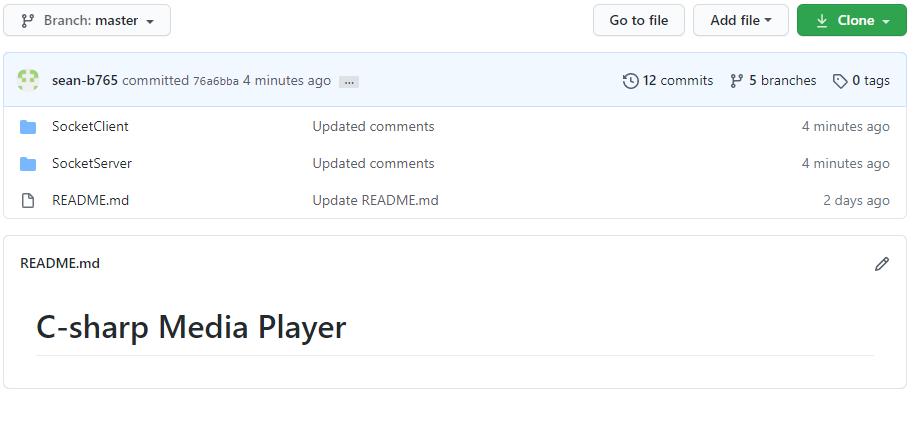
[Requirements 4](#_Toc43903431)

[Debugging Documentation 6](#_Toc43903432)

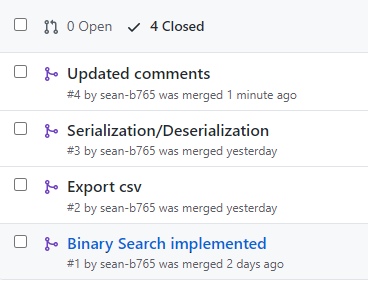
[Test Table 10](#_Toc43903433)

## Source Control

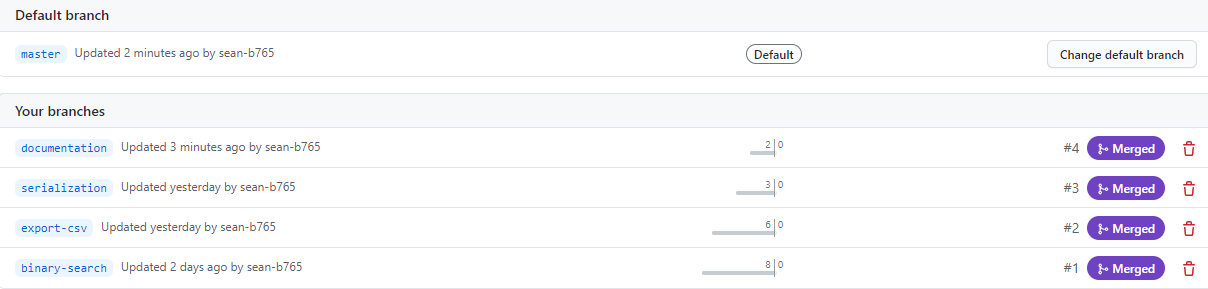
**Master**



**Pull Requests**

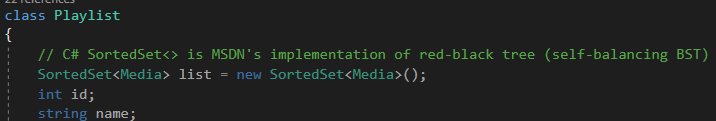


**Branches**



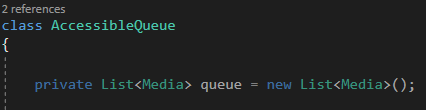
## Requirements

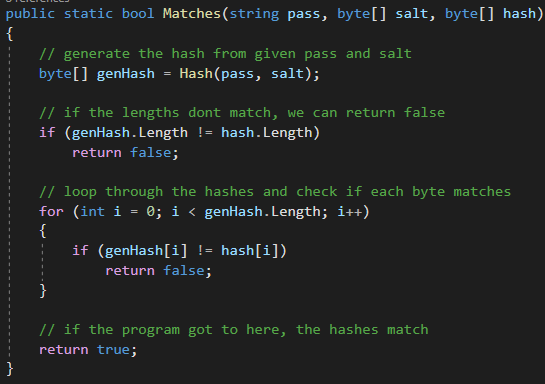
**Dynamic Data Structures:**

Stack, Queue, SortedSet<Media>



AccessibleQueue uses a List<> as a queue, but unlike a Queue, this class can access any element of the queue, rather than only the first element.





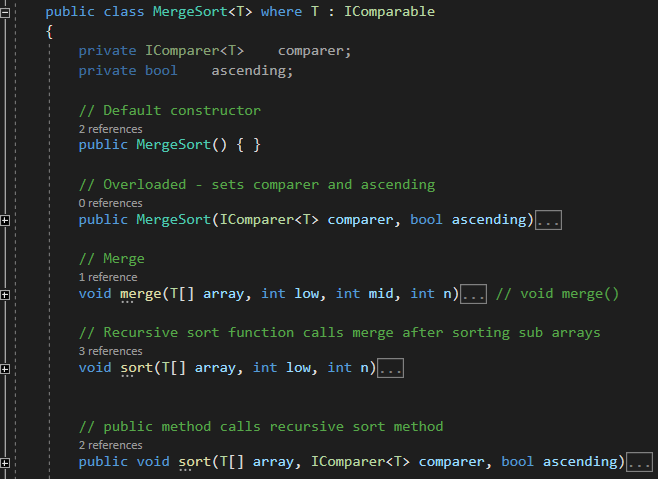
**Hashing Techniques:**

All hashing validation is performed on server side.

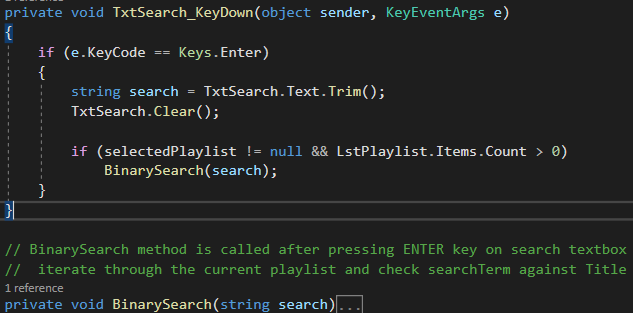
The class ClientFactory contains hash, salt generation methods. Also contains the Matches() method for validating hash.

**Sorting Algorithm:**

My generic Merge Sort class is used to sort a comparable array. Can take custom comparers, and asc/desc boolean.



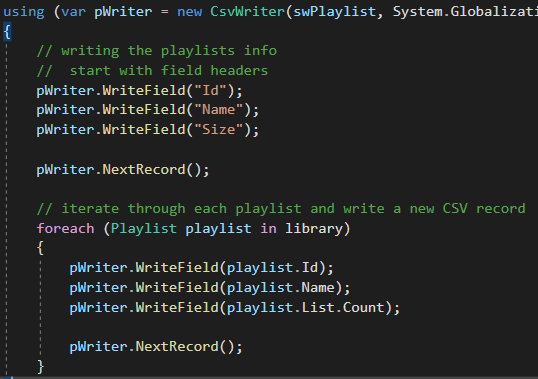
**Searching Algorithm:**



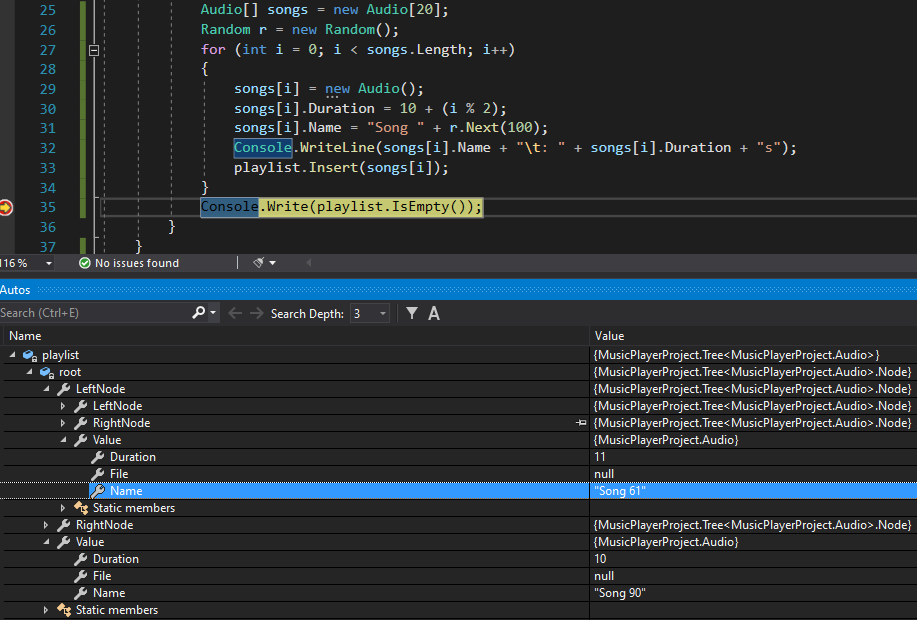
**Third Party Library:**

CSVHelper is the library used to export a CSV representation of the playlists and media objects. It will export Media and Playlist instance variables.

*BtnExport\_Click():*

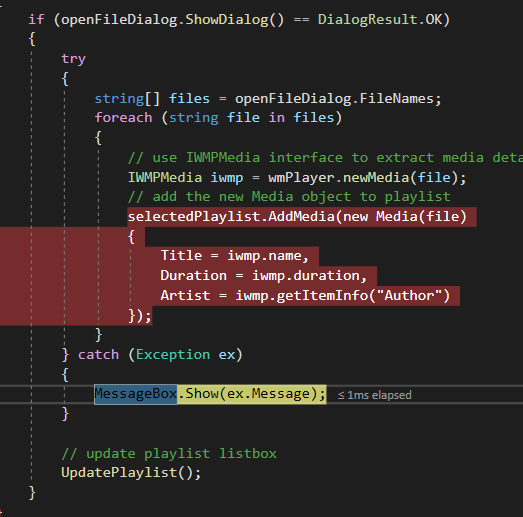


## Debugging Documentation



Testing my approach to song storage. I instantiated 20 song objects with random song names, and insertted these into my standard Binary Search Tree implementation. It will be unbalanced, but will be naturally sorted.

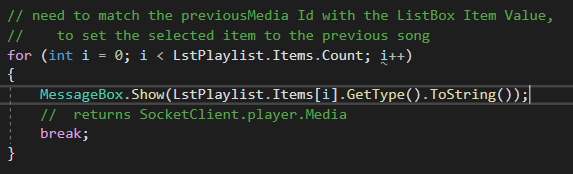
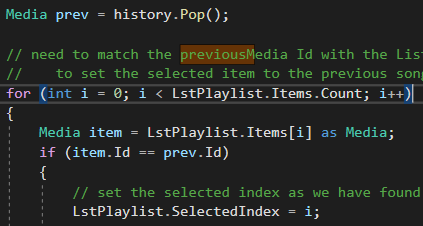
I ended up changing from my implementation of binary tree into C#’s **SortedSet**, as it is a self-balancing binary search tree.



A **SortedSet<T>** will require T to implement IComparable, as it is naturally sorted. This is why the above code caught an exception.

**Song History Stack**

The below code snippet is executed when pressing the Previous button. To figure out a way to get the Media object at the current ListBox item, I made sure that the types stored in the ListBox were of type Media.

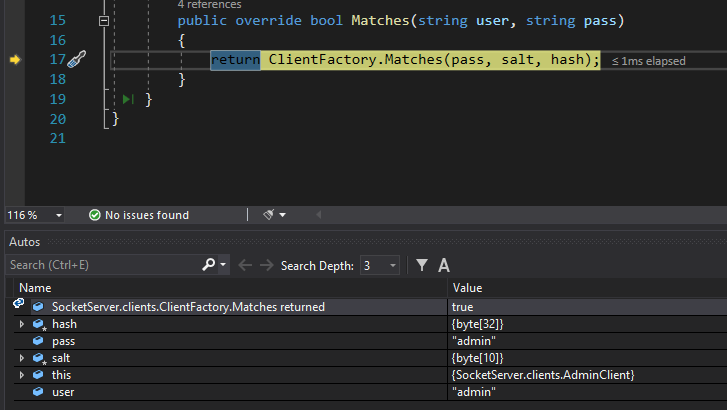


After confirming this, I could then get the previous Media using the song history Stack pop(), and checking if it is in the current playlist by iterating. If the Id’s match, then I can set the selected index, so the previous song gets highlighted.

**Hashing debugging**

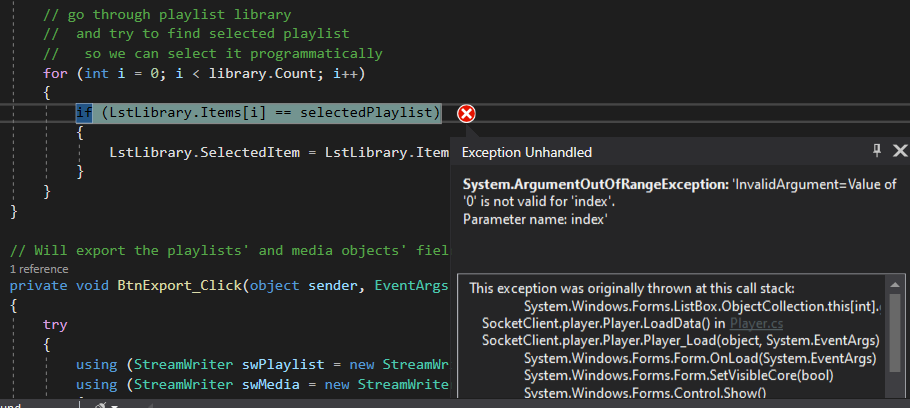
Validating hashes was done in the Account instance. First, loop through all accounts and see if Client user input matches the Account username. If so, call the account.Matches() method, which uses the ClientFactory static method.

We can see here that the ClientFactory Matches method returns true when user and pass is **admin**, as this is the default admin record. It will return false if no account matches Client user input.

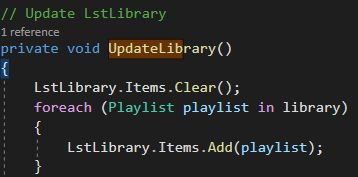


**Deserialization**

When the Player form loads, deserialization will occur to retrieve the saved program state.

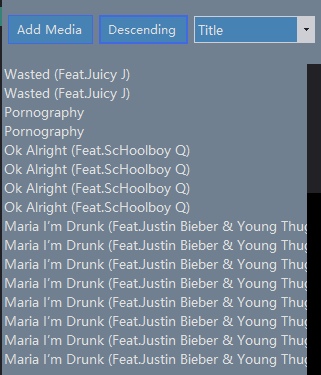
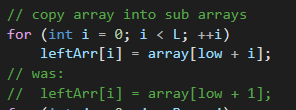


After deserialization, I ran into this error. Which was because LstLibrary had no items added to it, despite all playlists being serialized. To fix, I implemented the UpdateLibrary() method, which gets called after deserialization.



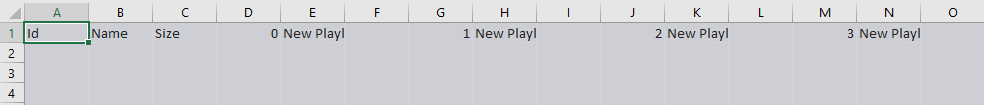
**Generic Merge Sort**

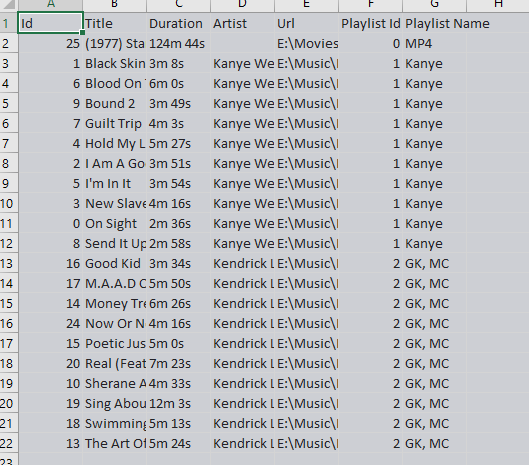
When attempting to sort songs, there was an obvious error.

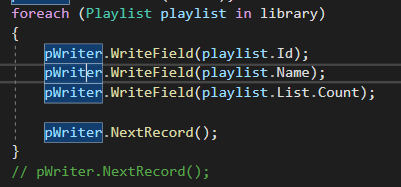
This seemed as though it would be obvious to spot in code, so after performing a quick code review of my MergeSort class, I found the source of the error.

**CSV Exporting**

When exporting my playlists to the CSV file using CSVHelper, I first saw this:



This was because my foreach loop (which iterates each playlist) did not call NextRecord(), so I fixed that, and it looked better.



## Test Table

*If the link cannot be opened, go to screencaps folder for figures 1 - 6 (GIF Files).*

|  |  |
| --- | --- |
| [Figure 1](https://tafewa-my.sharepoint.com/personal/30010353_tafe_wa_edu_au/Documents/Diploma/Programming%20III/AT3%20Project/screencaps/figure%201.gif)  *Testing the song queue, history stack* | [Figure 2](https://tafewa-my.sharepoint.com/personal/30010353_tafe_wa_edu_au/Documents/Diploma/Programming%20III/AT3%20Project/screencaps/figure2.gif)  *Queue songs between playlists* |
| [Figure 3](https://tafewa-my.sharepoint.com/personal/30010353_tafe_wa_edu_au/Documents/Diploma/Programming%20III/AT3%20Project/screencaps/figure3.gif)  *Testing new playlist, edit playlist* | [Figure 4](https://tafewa-my.sharepoint.com/personal/30010353_tafe_wa_edu_au/Documents/Diploma/Programming%20III/AT3%20Project/screencaps/figure4.gif)  *Remove a song from playlist*  ***Removing the currently playing song will stop the player*** |
| [Figure 5](https://tafewa-my.sharepoint.com/personal/30010353_tafe_wa_edu_au/Documents/Diploma/Programming%20III/AT3%20Project/screencaps/figure5.gif)  *Testing sorting by Title and Duration* | [Figure 6](https://tafewa-my.sharepoint.com/personal/30010353_tafe_wa_edu_au/Documents/Diploma/Programming%20III/AT3%20Project/screencaps/figure6.gif)  *Remove Media, then view CSV spreadsheet for changes* |
| Figure 7    Incorrect login attempts | Figure 8    Correct login |
| Figure 9    Client program stops / disconnect from server | Figure 10    Client reconnects |
| Figure 11    Connection Request on wrong port |  |

Fin.