Student Name: Calvin Moylan

Student ID: 30018702

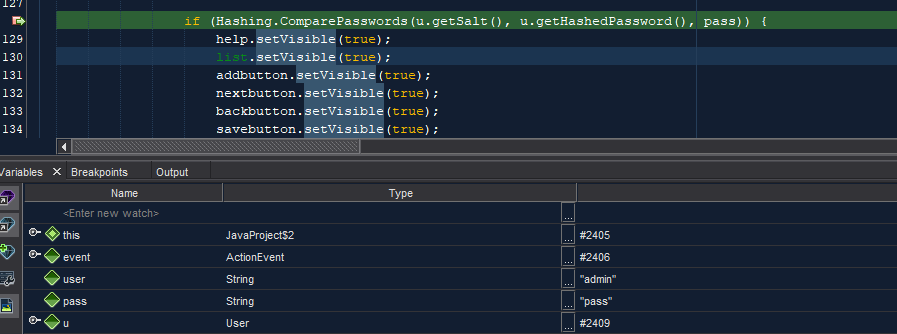
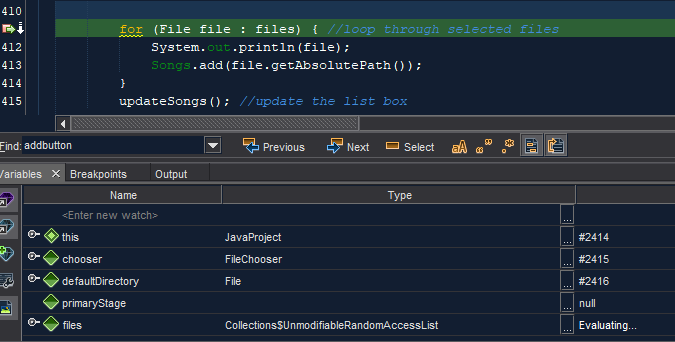
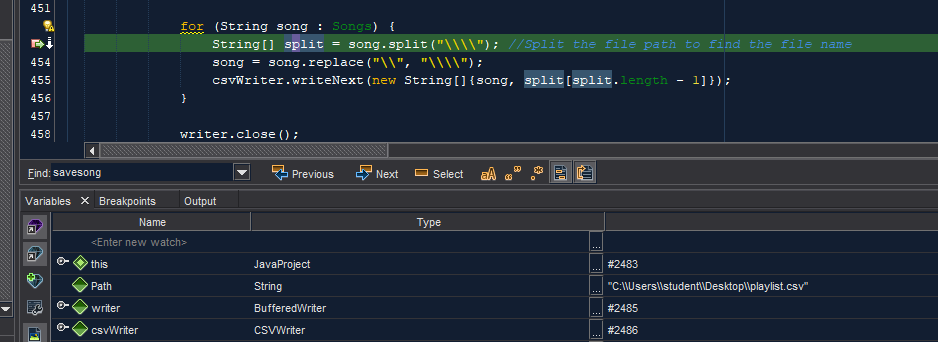
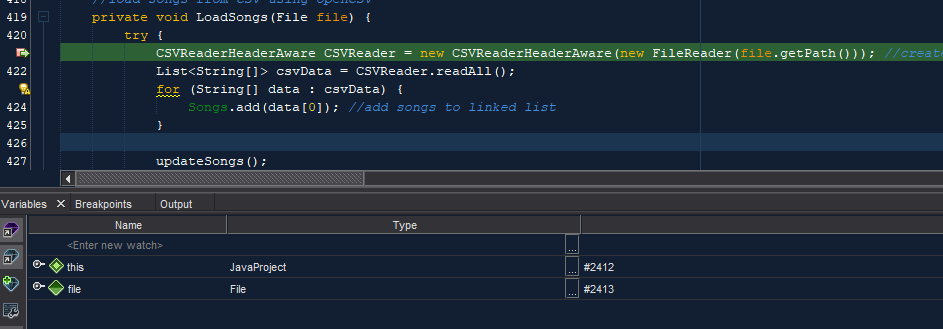
Date: 03/12/2020

Assessment Title: AT3 Project

## AT 3: Question 1

### UML

## Debugging

Logging in:  
  
Adding song:  
  
Saving playlist:  
  
Loading playlist:  


### Test and validate with sample inputs with screenshots.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Test Case** | **Data** | **Expected Result** | **Actual Result / Comment** |
| *Case 1* | *Correct username and password* | *Login details: “admin” “pass”* | *Signs into the application* | *Results as expected.*  *Ref Figure 1.1 & 1.2* |
| *Case 2* | *Incorrect username and password* | *“wwww” as username and password* | *User is not logged into the application* | *Results as expected.*  *Ref Figure 1.3* |
| *Case 3* | *User loads songs into application from CSV* | *Previously saved CSV on disk.* | *Songs are loaded into application* | *Results as expected.*  *Ref Figure 1.4 & 1.5* |
| *Case 4* | *User saves song into CSV* | *Songs already added to the LinkedList* | *Songs are sorted accordingly* | *Results as expected.*  *Ref Figure 1.6 & 1.7* |
| *Case 5* | *User sorts songs* | *Path for user to save file* | *Songs are exported into a csv file.* | *Results as expected.*  *Ref Figure 1.8 & 1.9* |
| *Case 6* | *User searches for song that exists.* | *User inputted data to search for* | *The application finds and highlights the found song* | *Results as expected.*  *Ref Figure 1.10* |
| *Case 7* | *User searches for song that does not exist.* | *User inputted data to search for* | *The application tells the user the song cannot be played.* | *Results as expected.*  *Ref Figure 1.11* |

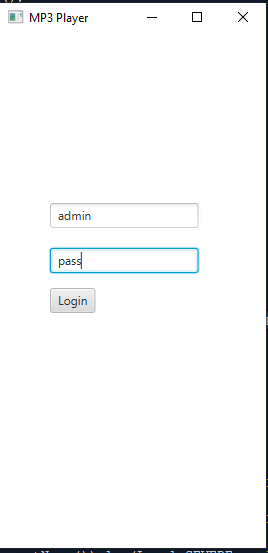


Figure 1.

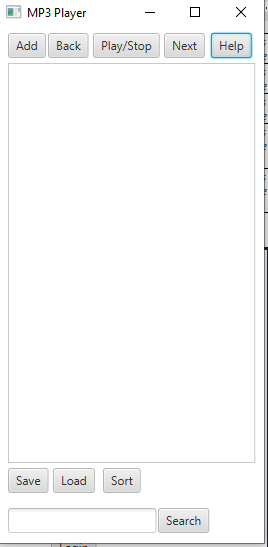


Figure 1.2

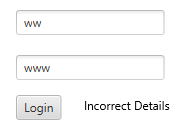


Figure 1.3

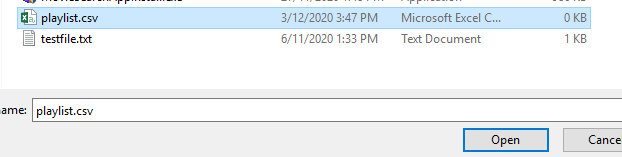


Figure 1.4

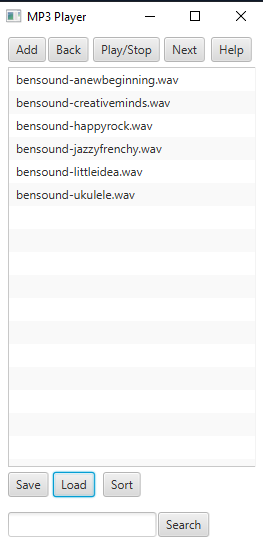


Figure 1.5

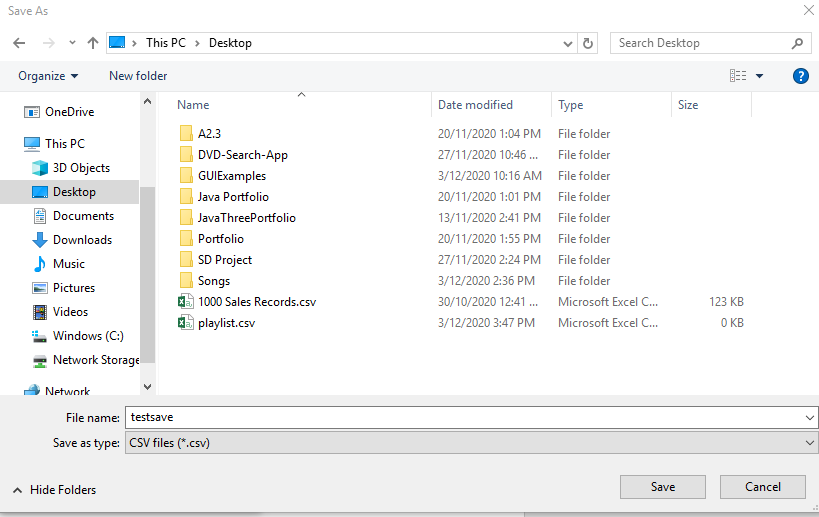


Figure 1.6

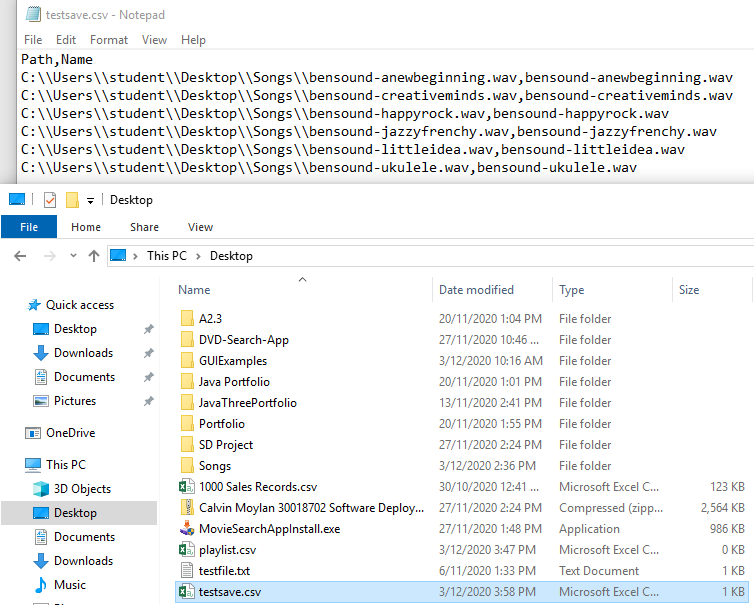


Figure 1.7

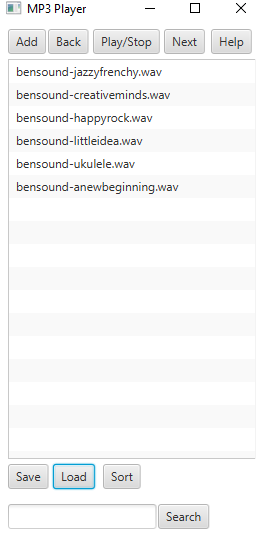


Figure 1.8

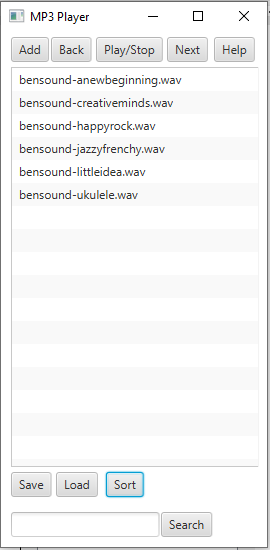


Figure 1.9

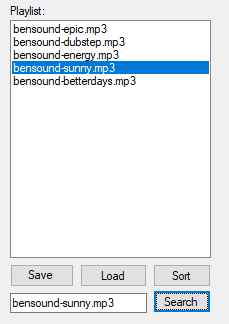


Figure 1.10

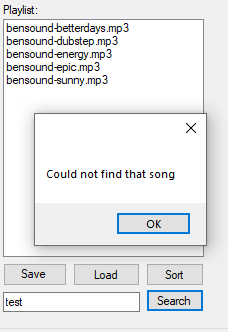


Figure 1.11

END OF TEST TABLE

## Meeting all requirements

### - Implement your solution

Implemented.

### - Must contain a dynamic data structure

For my dynamic data structure, I decided to use a Linked List of type string. This is because I believe this would best suit an application like this and as I am reusing code I had already written before, I decided it’s best to leave it as it is.

### - Must contain hashing techniques

I use hashing when storing the admin username and password. This is done inside the user repo along with the rest of the classes that help me achieve this.

### - Must contain sorting algorithm

I chose to use the merge sort algorithm, this is because it was one of the requirements and I decided that it would be a good fit for this project. It is used when the user wants to sort their playlist.

### - Must contain searching technique

The chosen searching technique I used for the project was the binary search. It is used when the user wants to search for a song in the playlist. I chose this as it was one of the requirements for the task.

### - Must contain 3rd party library

I chose to use the 3rd party library CSVHelper. This is because it has a good feature set and implements really well into a project. It is also really easy to use and has documentation of how to use it. I have also used it in the past so I am quite familiar with it.

### - Must have a GUI

The application is done in WinForms which is a Desktop GUI.

### - Must adhere to coding standards

The code of the application has followed all coding standards that can be found here: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>  
I have also commented the code so someone can read and understand what the code is doing.