Student Name: Calvin Moylan

Student ID: 30018702

Date: 03/12/2020

Assessment Title: AT3 Project

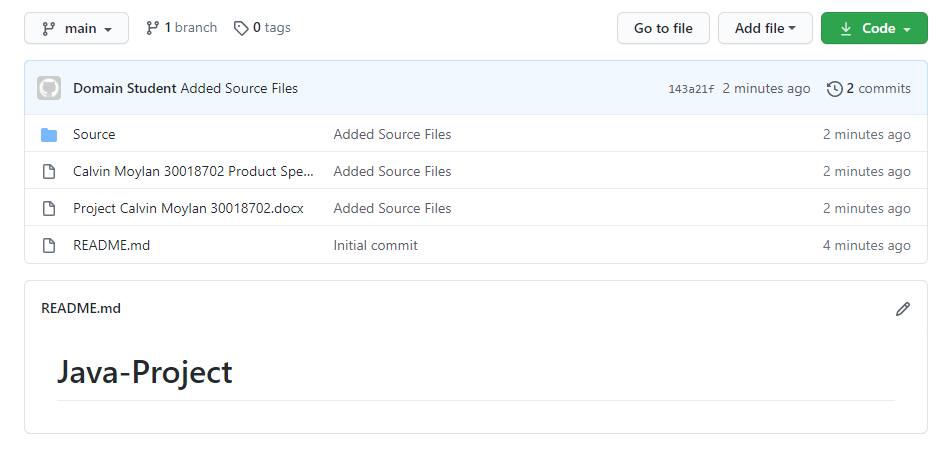
## AT 3: Question 1

### UML

### 

### GitHub

For this project, I will be using Git for version and source control along with GitHub to maintain and deploy my code. This is because I am most familiar with these and I will be able to use my previous knowledge to my advantage.

Here is a screenshot of the repo which can be found at: <https://github.com/CalvinMoylanTAFE/Java-Project>  


## Test Table

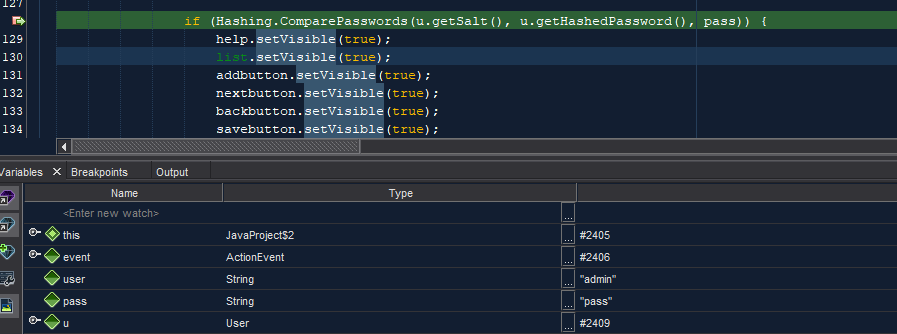
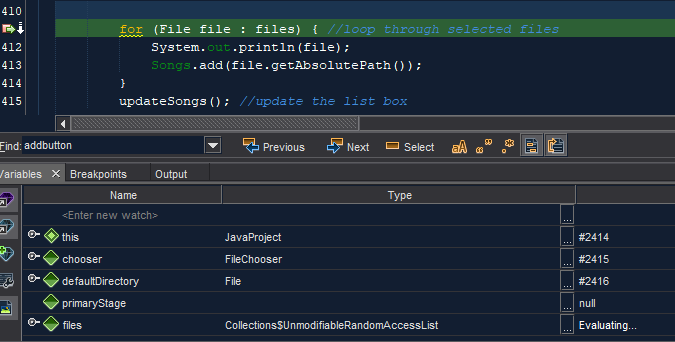
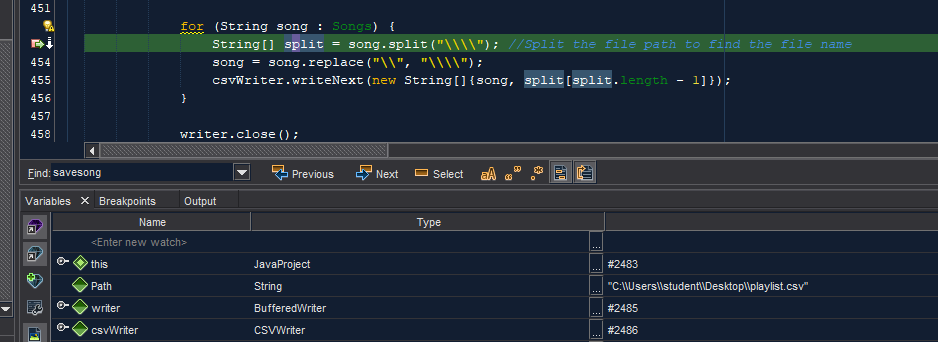
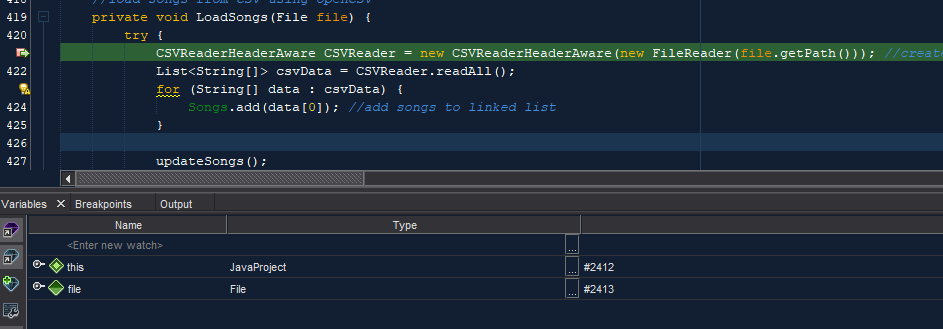
This is a mock-up of the test table I will be using to test the software that I will have implemented in Question 3:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Test Case** | **Data** | **Expected Result** |
| *Case 1* | *Correct username and password* | *Login details: “admin” “pass”* | *Signs into the application* |
| *Case 2* | *Incorrect username and password* | *“wwww” as username and password* | *User is not logged into the application* |
| *Case 3* | *User loads songs into application from CSV* | *Previously saved CSV on disk.* | *Songs are loaded into application* |
| *Case 4* | *User saves song into CSV* | *Songs already added to the LinkedList* | *Songs are sorted accordingly* |
| *Case 5* | *User sorts songs* | *Path for user to save file* | *Songs are exported into a csv file.* |
| *Case 6* | *User searches for song that exists.* | *User inputted data to search for* | *The application finds and highlights the found song* |
| *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.* |
| *Case 8* | *User plays song* | *User clicks the play button* | *The song is played* |
| *Case 9* | *User goes to next song* | *User clicks the next button* | *The application goes to the next song* |
| *Case 10* | *User goes back a song* | *User clicks previous song* | *The application goes to the previous song* |

## Product Design Specification Document

The document can be found along with this file named “Calvin Moylan 30018702 Product Specification Design Document”.

## Debugging

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


## Testing

### 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* |
| *Case 8* | *User plays song* | *User clicks the play button* | *The song is played* | *Results as expected.*  *Ref Figure 1.12 & 1.13* |
| *Case 9* | *User goes to next song* | *User clicks the next button* | *The application goes to the next song* | *Results as expected.*  *Ref Figure 1.14* |
| *Case 10* | *User goes back a song* | *User clicks previous song* | *The application goes to the previous song* | *Results as expected.*  *Ref Figure 1.15* |
| *Case 11* | *User stops playing song* | *User clicks stop* | *The application stops playing audio* | *Results as expected.*  *Ref Figure 1.16* |

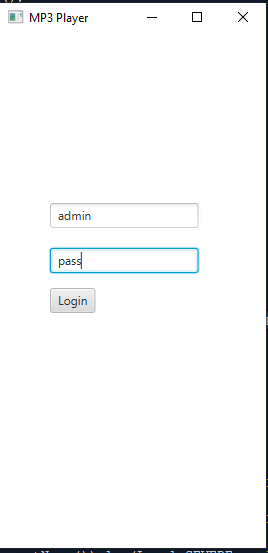


Figure 1.1

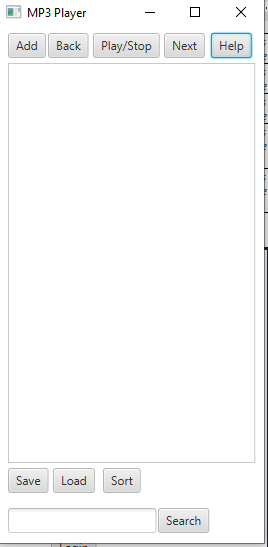


Figure 1.2

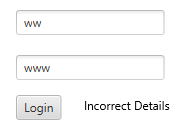


Figure 1.3

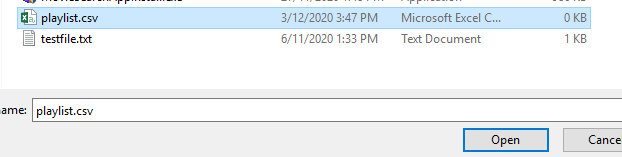


Figure 1.4

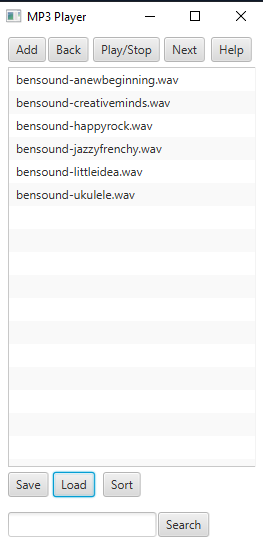


Figure 1.5

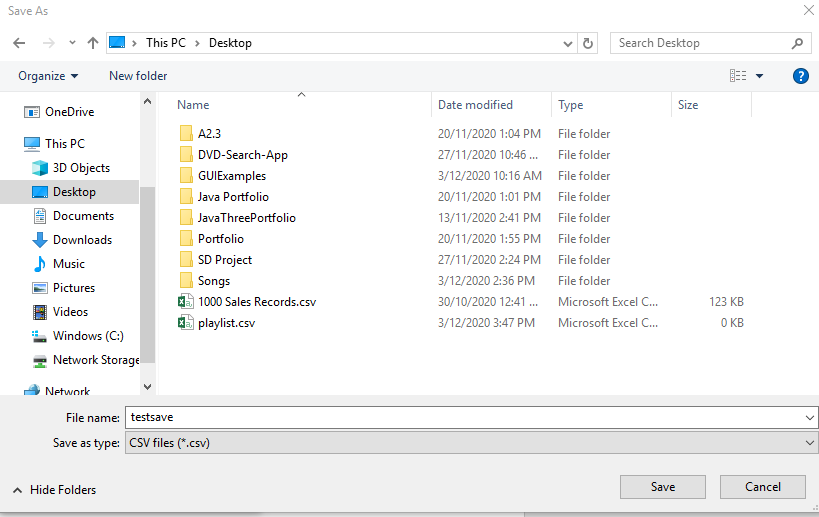


Figure 1.6

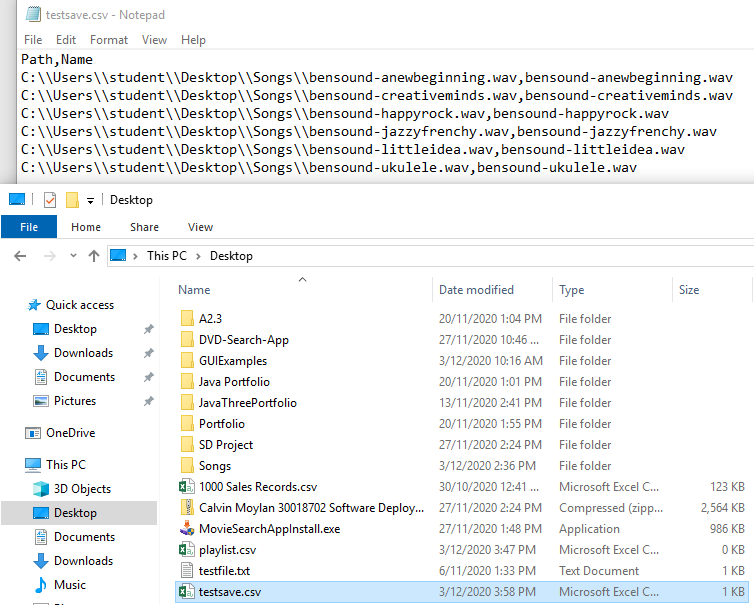


Figure 1.7

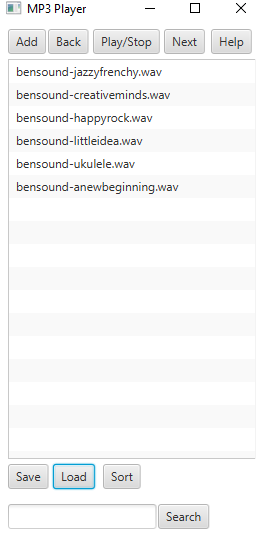


Figure 1.8

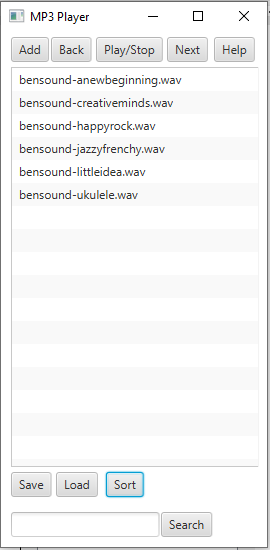


Figure 1.9

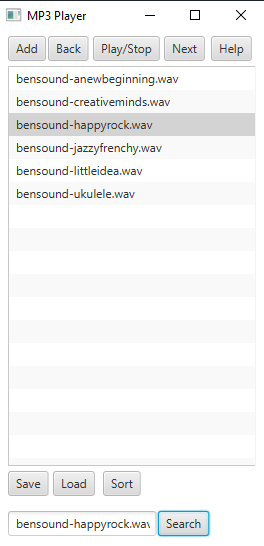


Figure 1.10

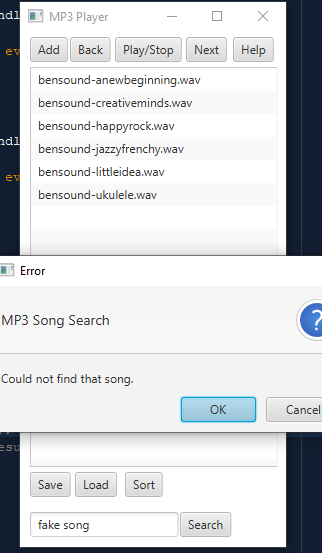


Figure 1.11

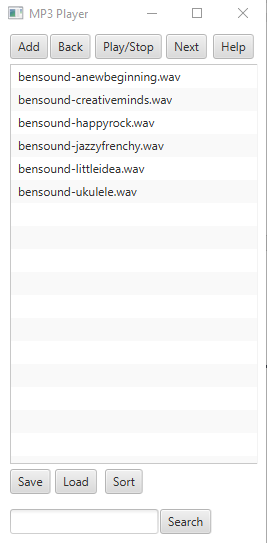


Figure 1.12

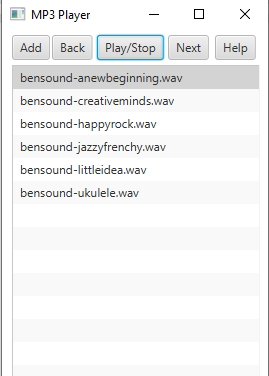


Figure 1.13

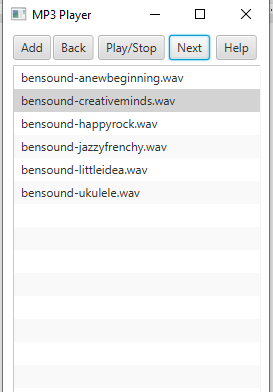


Figure 1.14

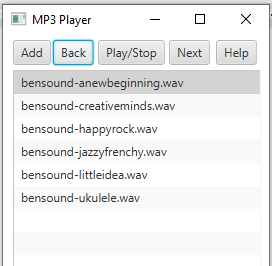


Figure 1.15

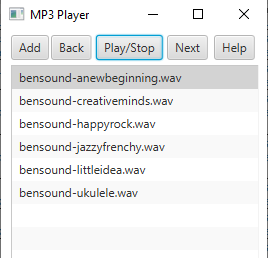
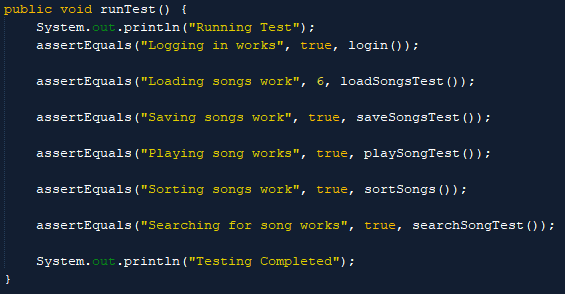
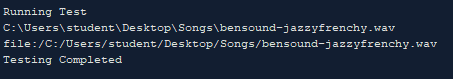


Figure 1.16

END OF TEST TABLE

From this test table, I can conclude that in the final version of the application there are no bugs and the application is working as intended to a satisfactory level.

### Junit Testing

I have put together 6 tests which revolve around the main functions of the program. These tests are:   
Logging in, Loading, Saving, Playing Song, Sorting and Searching.  
Successful run of Tests:  
  
  
Making Logging in Fail:  
  
Making Loading Songs Fail:  
  
Making Save Songs Fail:  
  
Making Play Song Fail:  
  
Making Sort Songs Fail:  
  
Making Search Song Fail:  


I was able to successfully run all tests and make them pass. I also forcefully made them fail to also double check that the checks are working themselves. This way I can guarantee that the application is working as intended.  
From these tests, I can conclude that the application is working as intended.

## 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. I went this route because I believe this would best suit this type of application. It is included in my main class

### - Must contain hashing techniques

I use hashing and salting when storing the admin username and password. This is done inside the JavaProject class which keeps a HashMap of String, User to store the admin details.

### - 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. The algorithm is used when sorting the playlist.

### - Must contain searching technique

I chose to use binary search for searching for a song in the application. It is used when the user wants to search for a song in the playlist. It was a requirement to use this searching technique.

### - Must contain 3rd party library

I chose to use the 3rd party library “OpenCSV”. I used this third party library as I had used it before I was already aware of how to use and implement it into my project.

### - Must have a GUI

The application is a JavaFX project 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://www.oracle.com/technetwork/java/codeconventions-150003.pdf>.  
The code has also been commented adequately.

### - Must contain help files

The application includes a help button that when pressed gives a detailed guide on how to use the application.