# Pre-Development

## Requirements

I will be developing a small program based on these requirements:

* The program must be of equal complexity to one of this description:
* An advanced music player that allows the ability to sort and search the songs stored in a binary tree (any sort and search algorithm you select will have to be approved if it is not merge sort and binary search), the GUI should display the sorted track list and highlight and play the searched track, it should save the track list to a csv using a 3rd party library. The music player must load and play files
* It must also meet these specific requirements:
* Must contain dynamic data structures
* Must contain hashing techniques
* Must contain sorting algorithm
* Must contain searching technique
* Must contain 3rd party library
* Must have a GUI
* Must adhere to coding standards

## Proposal

Based on these requirements, I am proposing to create a program as follows:

* *The program will be an Audio file browser in C#.*
* *The program will have a GUI made with Windows Forms and .NET Framework.*
* *The user will be able to browse audio files in a sorted* ***double linked list****.*
* *The double linked list will be sorted with* ***Merge Sort***
* *The user will be able to search for a file by name with* ***Binary Search****.*
* *The user will need to log in before they can use the program. Their password will be hashed for storage using SHA-256.*
* *The passwords will be retrieved from a CSV file using an external library.*
* *The songs list can be saved to a CSV file using an external library.*
* *The guidelines outlined here* [C# Coding Standards (c-sharpcorner.com)](https://www.c-sharpcorner.com/UploadFile/ankurmalik123/C-Sharp-coding-standards/) will be observed to the best of my ability when creating code.
* *All variables will be named properly and the code will be formatted. All methods will have proper Visual Studio documentation generator comments. (“///” comments)*
* *Instructions for the user will be included in a separate document.*

The source control for the project will be done using GitHub. The repository will be located at [NickoBrown/Programming-III-Project (github.com)](https://github.com/NickoBrown/Programming-III-Project)

## Testing Plan

The program will be tested according to these cases:

### Login Page

|  |  |  |
| --- | --- | --- |
| Test case | Expected Result | Actual Result |
| Login with correct credentials. | Proceed to next frame. |  |
| Login with incorrect credentials | Do not proceed. |  |

### Music Player Page

|  |  |  |
| --- | --- | --- |
| Test case | Expected Result | Actual Result |
| Add song | Song added |  |
| Play song | Song plays |  |
| Play song where no song is available. | Song does not play. |  |
| Stop song | Song stops |  |
| Search song | Correct song appears |  |
| Save songs to CSV | Songs save to CSV successfully. |  |
| Load songs from CSV | Songs load from CSV successfully. |  |
| Load deformed CSV file | Deformed songs are removed. |  |

# Post-Development

## Requirements mapping

#### Must contain dynamic data structures

The program makes use of a doubly linked list. It utilizes the built in LinkedList class provided by C#. This has the benefit of implementing valuable interfaces like IEnumerable, while still following the required doubly linked list structure of Node <-> Node <-> Node, and associated methods of access.

#### Must contain hashing techniques

The program makes use of the password hashing technique PBKDF2 “Password Based Key Derivation Function 2” to store passwords. On further research, the proposed algorithm SHA-256 is not considered suitable for password storage. For that reason, the change was made to using PBKDF2.

#### Must contain sorting algorithm

The program uses a Merge sort to sort the doubly linked list. The merge sort is generic and converts the input data into an array for sorting. The output is also in array form, and is fed back into the linked list structure after sorting.

#### Must contain searching technique

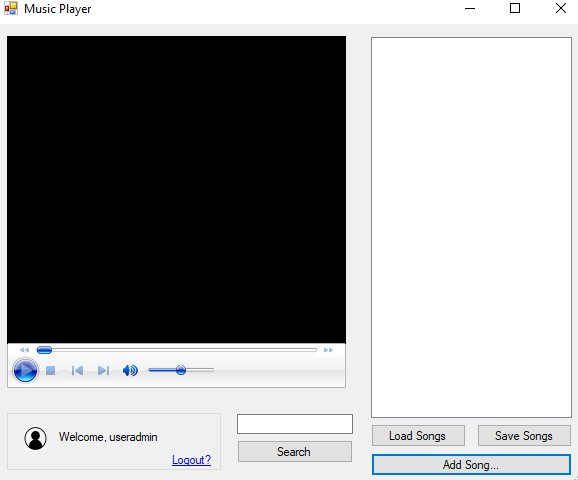
The program uses a generic binary search to provide search functionality.

#### Must contain 3rd party library

The program uses the 3rd party library CSVHelper.

#### Must have a GUI

The program uses Windows Forms to provide a GUI.



#### Must adhere to coding standards

The code adheres to the standards defined in the pre-development specifications to the best of my understanding.

## Testing

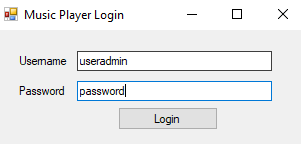
### Login Page

|  |  |  |
| --- | --- | --- |
| Test case | Expected Result | Actual Result / Screenshot |
| Login with correct credentials. | Proceed to next frame. | 1, 2 |
| Login with incorrect credentials | Do not proceed. | 3, 4 |

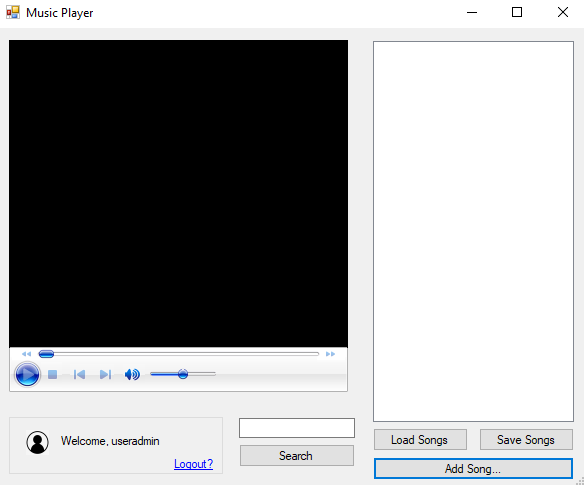
### Music Player Page

|  |  |  |
| --- | --- | --- |
| Test case | Expected Result | Actual Result |
| Add song | Song added | 5, 6 |
| Play song | Song plays | 7 |
| Play song where no song is available. | Song does not play. | No such button exists in the final version. |
| Stop song | Song stops | 8, 9 |
| Search song | Correct song appears | 10, 11 |
| Save songs to CSV | Songs save to CSV successfully. | 12 |
| Load songs from CSV | Songs load from CSV successfully. | 13, 14 |
| Load deformed CSV file |  | 15, 16, 17 |

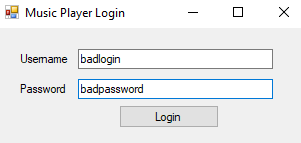
## Screenshots



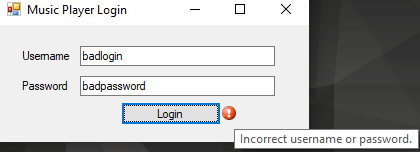
Screenshot 1



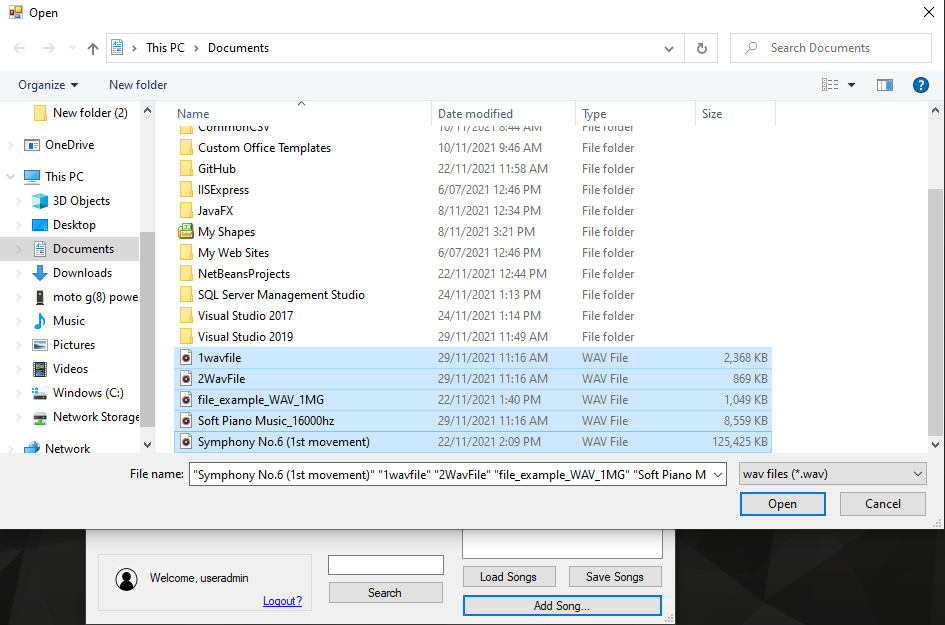
Screenshot 2



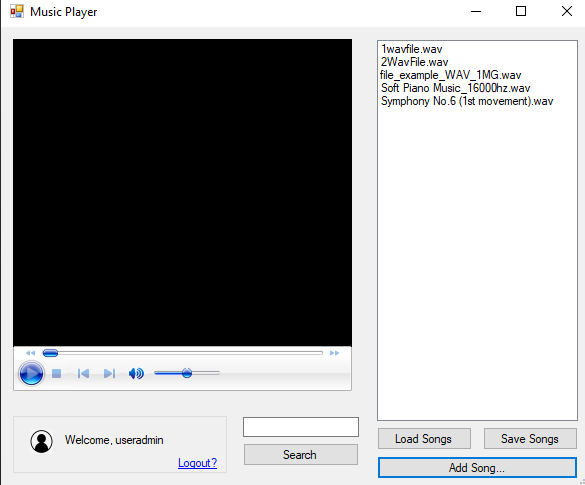
Screenshot 3



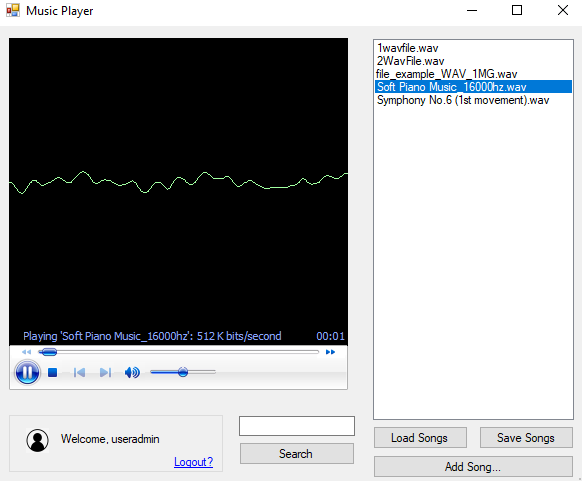
Screenshot 4



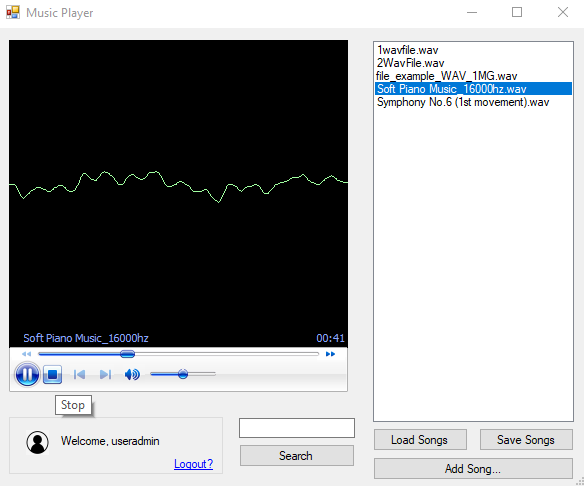
Screenshot 5



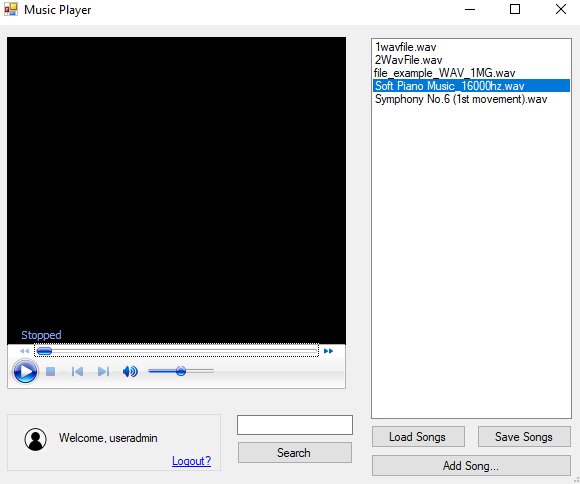
Screenshot 6



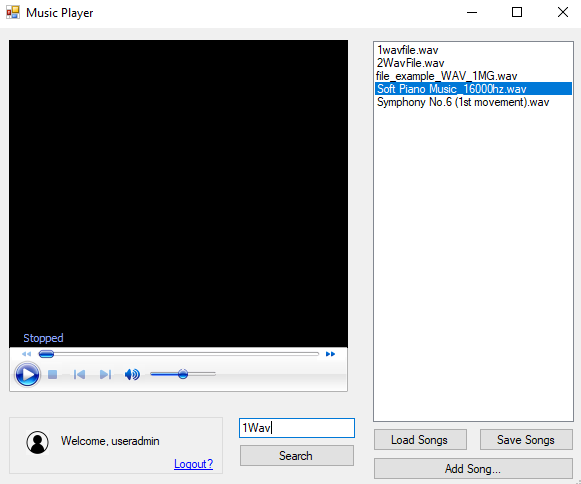
Screenshot 7



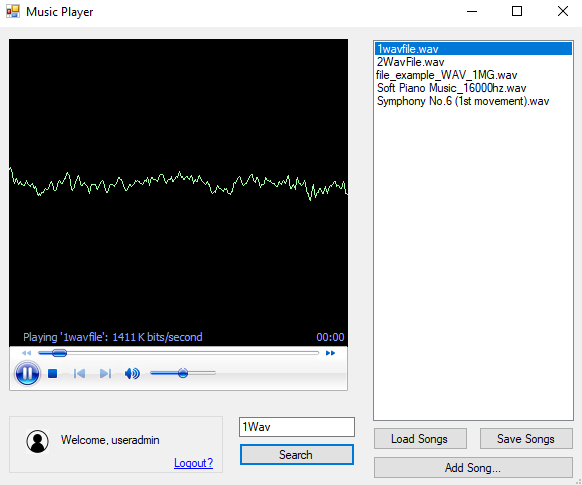
Screenshot 8



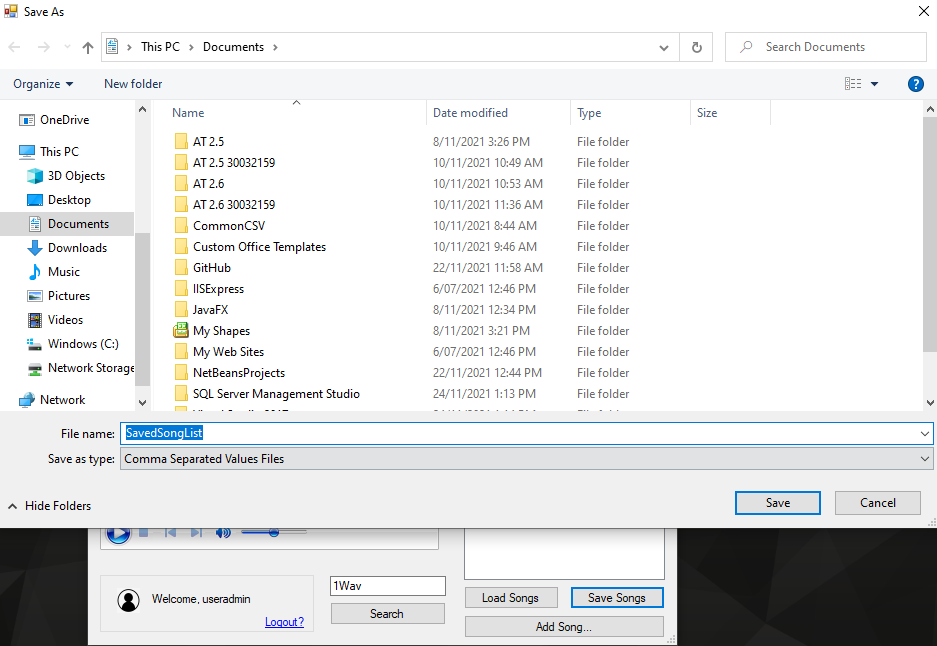
Screenshot 9



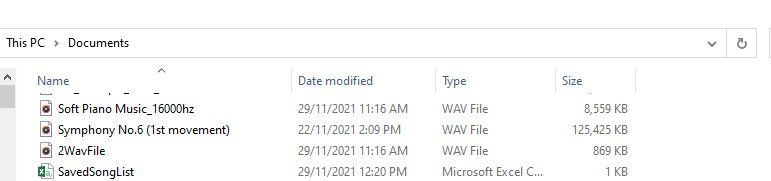
Screenshot 10



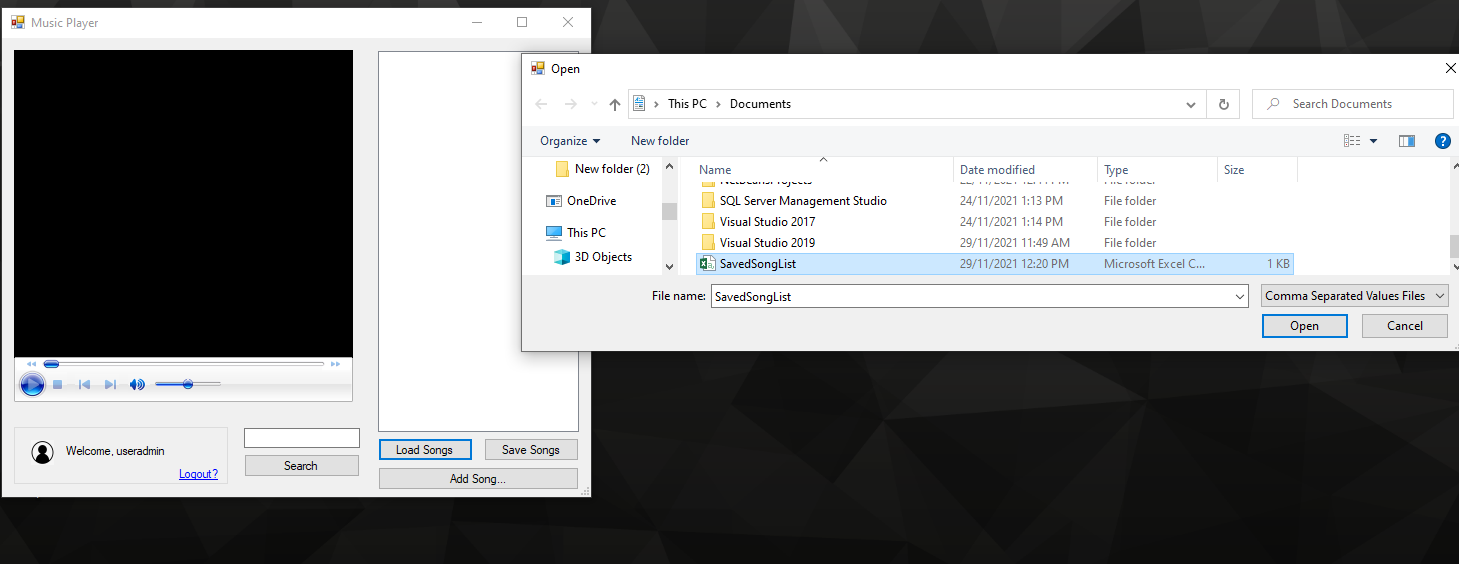
Screenshot 11



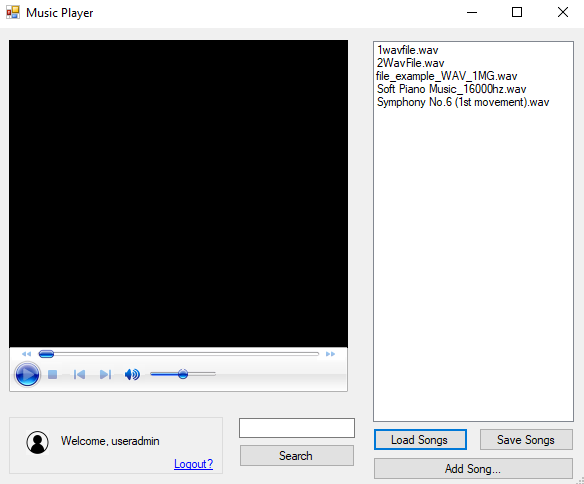
Screenshot 12



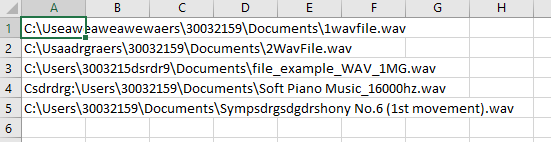
Screenshot 13



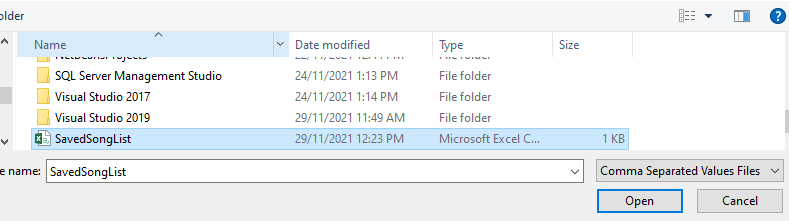
Screenshot 14



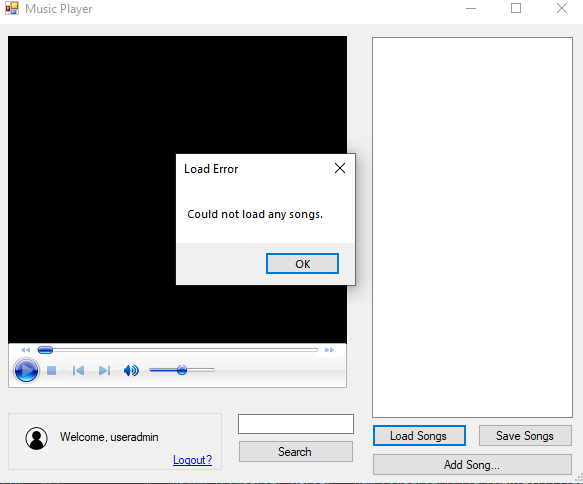
Screenshot 15



Screenshot 16



Screenshot 17



Screenshot 18