|  |
| --- |
| **Project Documentation** |
| **JMC MUSIC PLAYER** |
| Programming III |

|  |
| --- |
| Jose Rico Imbang  26/11/2020 |

# TABLE OF CONTENTS

[Introduction 1](#_Toc57298001)

[UML 2](#_Toc57298002)

[Test Data 3](#_Toc57298003)

[Method Signature for MusicPlayerForm 8](#_Toc57298004)

[Method Signature for FormServer 8](#_Toc57298005)

[List of Requirements 9](#_Toc57298006)

[Must contain dynamic data structures 9](#_Toc57298007)

[Must contain hashing techniques 9](#_Toc57298008)

[Must contain sorting algorithm 9](#_Toc57298009)

[Must contain searching technique 9](#_Toc57298010)

[Must contain 3rd party library 9](#_Toc57298011)

[Must have a GUI 9](#_Toc57298012)

[Must adhere to coding standards 10](#_Toc57298013)

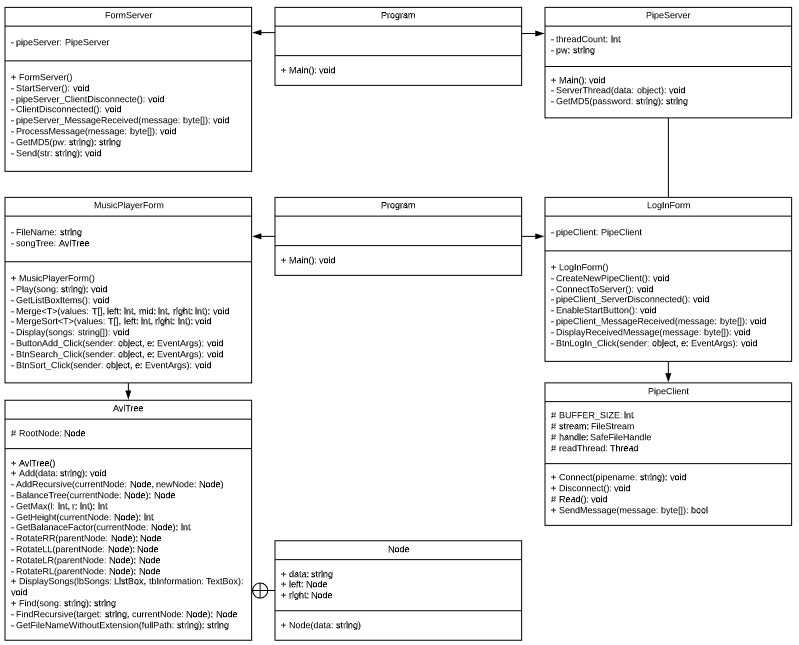
[References](#_Toc57298014)

## Introduction

There are two applications for this project – the Server and the Client. The former serves as the server on which the Client application can connect through Inter Process Communication (IPC) using Named Pipes, as long as they are on the same network. The Server application needs to be running before the Client application can request to establish a connection.

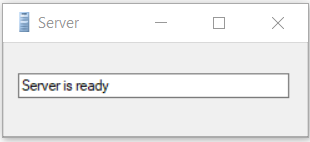
Once the Server is up and running, the user can open the Client application. The user will be prompted to enter his login details, and once entered, he can then click the Log In button and the Client application will send the information to the Server. The Server verifies the hashes the password then verifies the username and the hashed password. The user is prompted if the login details are incorrect. But if it is correct, the login window hides, and the music player window is shown. The user can add songs using the Add Songs button. The full path of the songs is saved in a CSV file so a third-party library is used to read those information. They are then saved into an AVL Tree. The unsorted names of the songs are then listed on the ListBox. There is a TextBox that the user can use to search for a specific song. Once the user typed in the name of the song into the TextBox, he then can click the Search button. The algorithm that is used to search is Binary Search. If the song is found, Windows Media Player plays it and the song’s name that is on the ListBox is highlighted. There is also a button to Sort the songs and it uses Merge Sort.

## UML

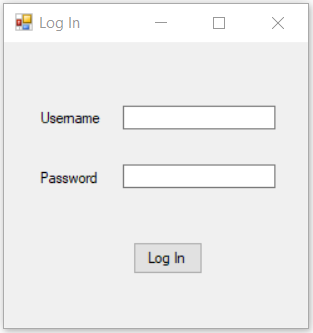


## Test Data

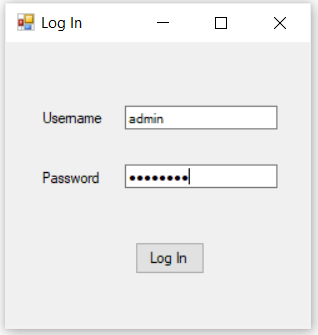
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Expected Outcome** | **Evidence** |
| Case 1 | Run the Server | Server up and ready. | Ref 1 |
| Case 2 | Run the Client | Log In form shows up. | Ref 2 |
| Case 3 | Enter Login details | The password’s characters are hidden. | Ref 3 |
| Case 4 | Click Log In button | The Log In form hides and the Music Player form is shown. The Server app shows a message that the connection with a Client is established. | Ref 4 |
| Case 5 | Click Add Songs button | A dialog box opens up and the user can select a CSV file. | Ref 5 |
| Case 6 | Click Open button from the dialog box | The songs are listed on the ListBox and a TextBox shows the Root Node. | Ref 6 |
| Case 7 | Type the name of the song and the click Search button | The song plays if it is found and the name is on the ListBox is highlighted. | Ref 7 |
| Case 8 | Click Sort button | The unsorted song names on the ListBox are sorted. | Ref 8 |



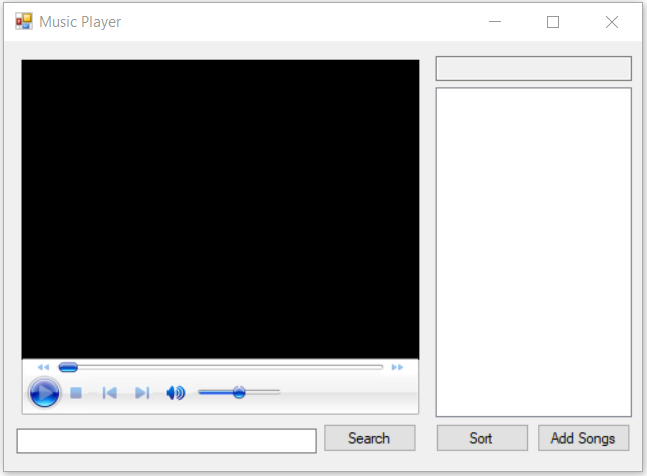
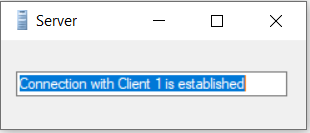
Ref 1



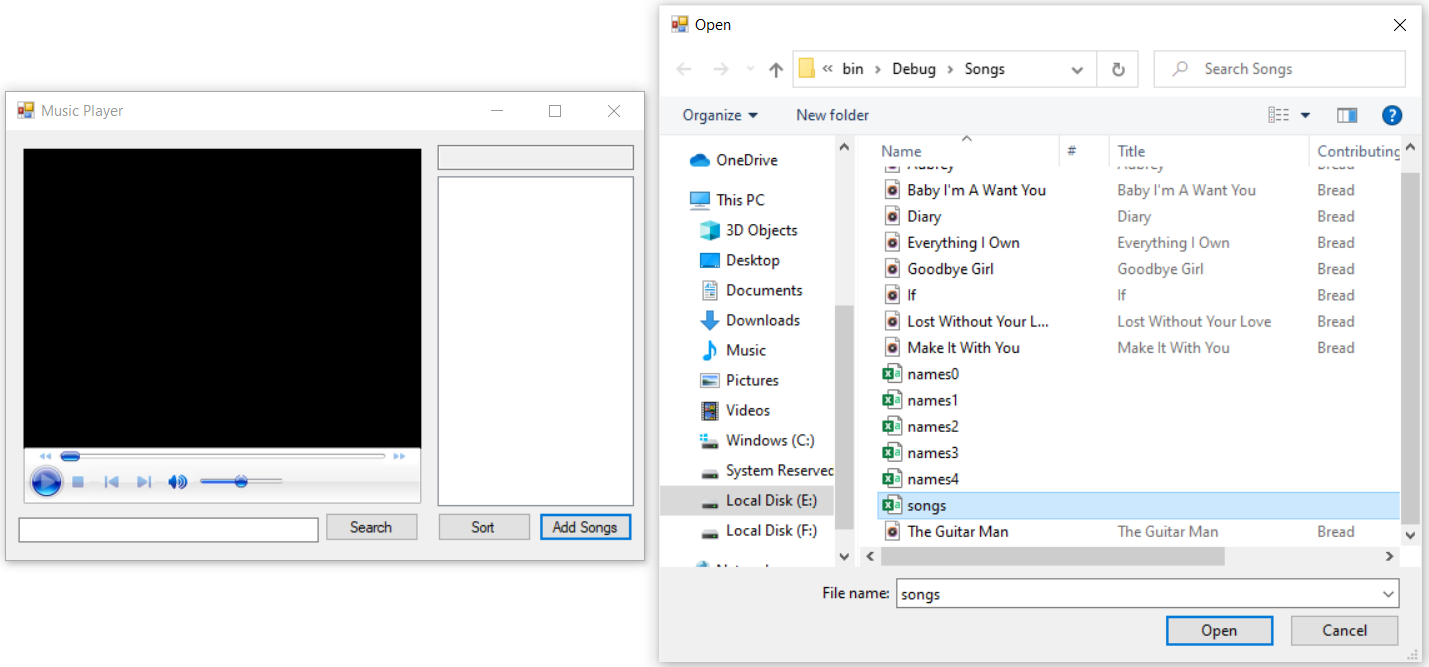
Ref 2



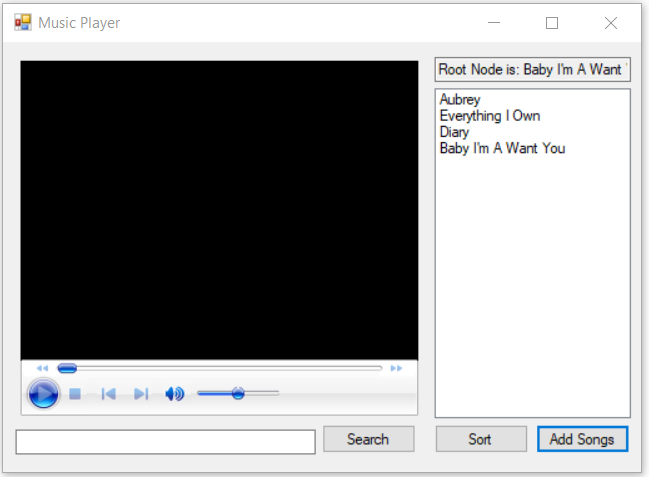
Ref 3



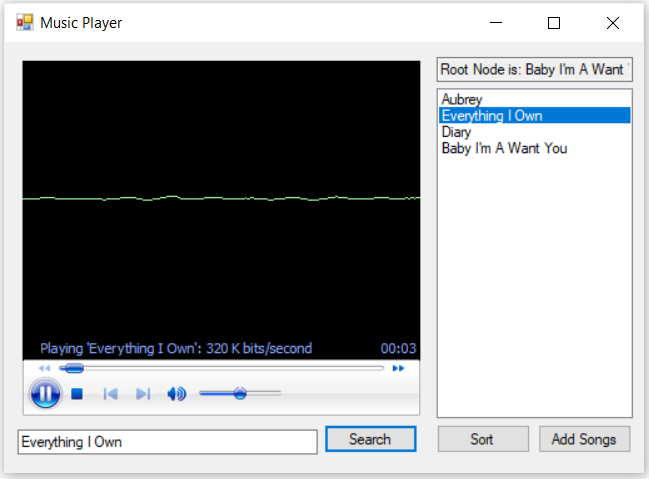
Ref 4



Ref 5



Ref 6



Ref 7



Ref 8

## Method Signature for MusicPlayerForm

private void ResetInformation() {}

private void Play(string song) {}

private void GetListBoxItems() {}

private void Merge<T>(T[] values, int left, int mid, int right) {}

private void MergeSort<T>(T[] values, int left, int right) {}

private void Display(string[] songs) {}

private void ButtonAdd\_Click(object sender, EventArgs e) {}

private void BtnSearch\_Click(object sender, EventArgs e) {}

private void BtnSort\_Click(object sender, EventArgs e) () {}

## Method Signature for FormServer

public FormServer() {}

private void StartServer() {}

private void pipeServer\_ClientDisconnected() {}

private void ClientDisconnected() {}

private void pipeServer\_MessageReceived(byte[] message) {}

private void ProcessMessage(byte[] message) {}

private static string GetMD5(string pw) {}

private void Send(string str) {}

## List of Requirements

Regions are created in the code to easily find the requirements.

Example:

#region \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*THIRD-PARTY LIBRARY\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### Must contain dynamic data structures

The dynamic data structure that is implemented for this project is AvlTree and a separate class is created to demonstrated this. The class can be found in the MusicPlayer project.

### Must contain hashing techniques

Hashing is implemented in the PipeServer Project in FormServer partial class. GetMD5 is the specific method where it is implemented. This method is used to hash the default password and the password from the MusicPlayer Project.

### Must contain sorting algorithm

The sorting algorithm that is used for this project is Merge Sort and it is in MusicPlayer Project specifically inside MusicPlayerForm.cs. The names of the methods where it is implemented are MergeSort and Merge.

### Must contain searching technique

The algorithm that is used for searching is Binary Search and it is in MusicPlayer Project inside the AvlTree class. The names of the methods are Find and FindRecursive.

### Must contain 3rd party library

The 3rd party library is implemented in MusicPlayer Project inside MusicPlayerForm. The name of the specific method is Display.

### Must have a GUI

Windows Forms are implemented for the GUI.

### Must adhere to coding standards

The coding standard that is used for this project is C# Coding Conventions to create a consistent look to the code so that readers can focus on content a not on layout. This also helps facilitate copying, changing, and maintaining the code. Click the link to know more about [C# Coding Conventions](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions).

## References

[3rd Party Library](https://www.codeproject.com/Articles/9258/A-Fast-CSV-Reader)

[C# Coding Conventions](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions)