Programming III Project

AT3 Project

Kyle Cleofe - 30000250

2019

Table of Contents

[**Planning (Question 1)** 3](#_Toc26454600)

[1. What data structures are you using? 3](#_Toc26454601)

[2. Where are you using hashing techniques? 3](#_Toc26454602)

[3. What sorting algorithm are you using how this is different from selection and bubble sort? 3](#_Toc26454603)

[4. What search technique are you using? 4](#_Toc26454604)

[5. What third party libraries are you using? 4](#_Toc26454605)

[6. Where can I find the documentation for this? 4](#_Toc26454606)

[7. A mock-up of the GUI. 4](#_Toc26454607)

[8. What source control are you using? 5](#_Toc26454608)

[9. What are your coding standards you are enforcing? 6](#_Toc26454609)

[10. What tests are you going to run? 6](#_Toc26454610)

[**Product Design Specification (SRS)** 7](#_Toc26454611)

[1 INTRODUCTION 7](#_Toc26454612)

[1.1 PURPOSE OF THE PRODUCT DESIGN SPECIFICATION DOCUMENT 7](#_Toc26454613)

[1.2 DOCUMENT CONVENTIONS 7](#_Toc26454614)

[1.3 INTENDED AUDIENCE AND READING SUGGESTIONS 8](#_Toc26454615)

[1.4 PROJECT SCOPE 8](#_Toc26454616)

[1.5 REFERENCES 8](#_Toc26454617)

[2 OVERALL DESCRIPTION 8](#_Toc26454618)

[2.1 PRODUCT PERSPECTIVE 8](#_Toc26454619)

[2.2 Product Features 8](#_Toc26454620)

[2.3 USER CLASS and CHARACTERISTICS 9](#_Toc26454621)

[2.4 OPERATING ENVIRONMENT 9](#_Toc26454622)

[2.5 DESIGN and IMPLEMENTATION CONSTRAINTS 10](#_Toc26454623)

[2.6 ASSUMPTION DEPENDENCIES 10](#_Toc26454624)

[3. SYSTEM FEATURES 10](#_Toc26454625)

[3.1 Functional Requirements 10](#_Toc26454626)

[4. EXTERNAL INTERFACE REQUIREMENTS 11](#_Toc26454627)

[4.1 USER INTERFACES 11](#_Toc26454628)

[4.2 HARDWARE INTERFACES 11](#_Toc26454629)

[4.3 SOFTWARE INTERFACES 11](#_Toc26454630)

[4.4 COMMUNICATION INTERFACES 11](#_Toc26454631)

[5. NONFUNCTIONAL REQUIREMENTS 11](#_Toc26454632)

[5.1 PERFORMANCE REQUIREMENTS 11](#_Toc26454633)

[5.2 SAFETY REQUIREMENTS 12](#_Toc26454634)

[5.3 SECURITY REQUIREMENTS 12](#_Toc26454635)

[5.4 SOFTWARE QUALITY ATTRIBUTES 12](#_Toc26454636)

# **Planning (Question 1)**

## What data structures are you using?

The data structures that I am using for this project is a linked list to create, read, update and delete elements. I will also use List<T> if it is applicable to add and read panels.

## Where are you using hashing techniques?

I am using hashing techniques on my starting login page that logs in a user to use the main component of the program. The type of hashing technique I am using is using the named pipes client stream and named pipes server stream to communicate between server and client.

<https://www.c-sharpcorner.com/article/compute-sha256-hash-in-c-sharp/>

## What sorting algorithm are you using how this is different from selection and bubble sort?

The sorting algorithm that I am using in this project is an insertion sort to sort the music.

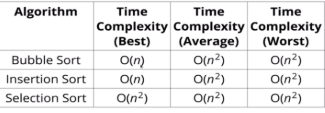
Insertion sort is different from selection because it sorts in its place by searching 2 elements and sorting it through the array unlike selection sort it searches the whole array for the smallest element to select then puts it in the starting index of the array. Insertion sort is different from bubble sort because bubble sort compares the largest number in the array from iterating two elements each time in the array and sends it to the last index of the array.

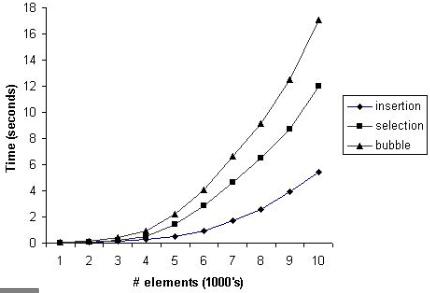
<https://pediaa.com/what-is-the-difference-between-insertion-sort-and-selection-sort/>

<http://en.wikipedia.org/wiki/Insertion_sort>

<https://en.wikipedia.org/wiki/Bubble_sort>

Time complexity of the different sorting algorithms that include worst, best and average.





## What search technique are you using?

The search technique I am using is the linear. I chose the linear search algorithm because binary search will be harder to do on linked lists and unnecessary in this project as it will not contain a lot of items in the linked list.

<https://www.geeksforgeeks.org/binary-search/>

## What third party libraries are you using?

The third party library that I’m using is CSV helper for making the program for the audio player. This will be used to read the csv files of the contained music file paths.

<https://joshclose.github.io/CsvHelper/>

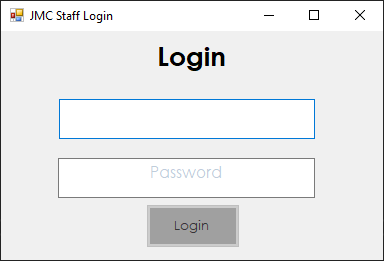
## Where can I find the documentation for this?

The documentation for this project can be found in my source control (github). Link: <https://github.com/kyle623/Project-AT3/>

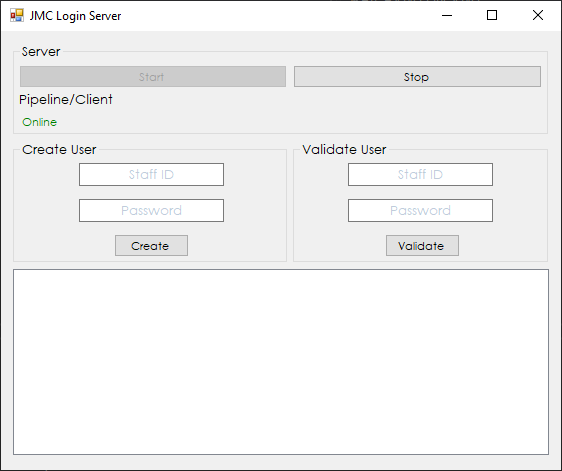
Link to the documentation of the CSV helper: <https://joshclose.github.io/CsvHelper/>

## A mock-up of the GUI.

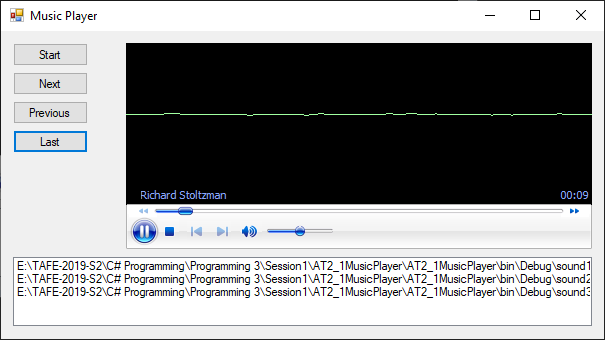
Login GUI



Server GUI



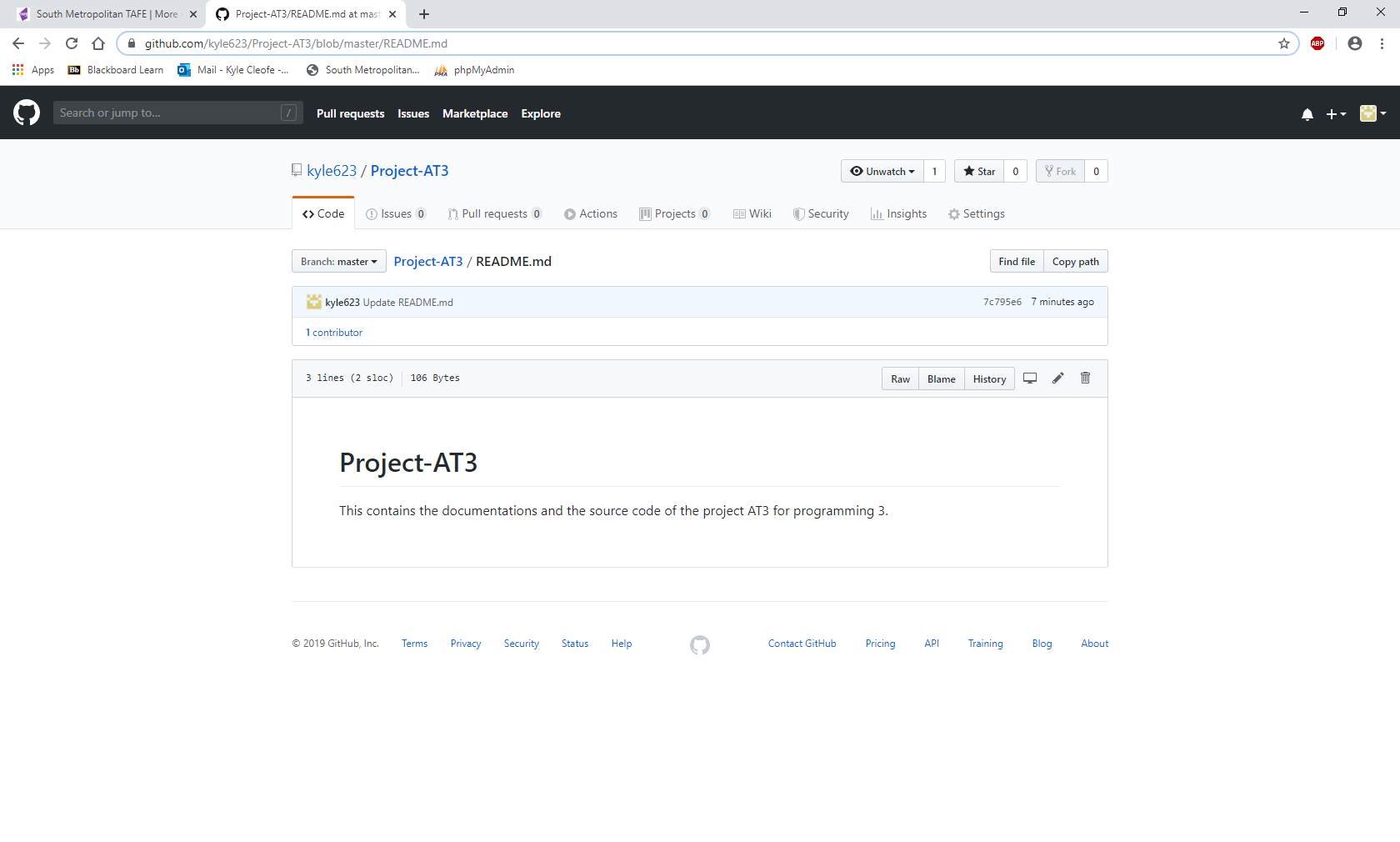
Music player



## What source control are you using?

The source control that I’m using for this program is GitHub link to my source control is here

<https://github.com/kyle623/Project-AT3/>



## What are your coding standards you are enforcing?

The coding standards I am enforcing are:

* Commenting and Documentation
* Consistent Indention
* Avoid Obvious Comments
* Code Grouping
* Consistent Naming Scheme
* DRY Principle (Don’t Repeat Yourself)
* Try Avoid Deep Nesting
* File and Folder Organization
* Consistent Temporary Names
* Separation of Code and Data
* Alternate Syntax Inside Templates
* Object Oriented vs. Procedural
* Read Open Source Code

## What tests are you going to run?

The tests that I am mostly going to run are grey box testing. I am using grey box testing because it has a mix of both black and white box testing. Using grey box testing is great for testing the input and outputs of the programs to find bugs.

# **Product Design Specification (SRS)**

## 1 INTRODUCTION

### 1.1 PURPOSE OF THE PRODUCT DESIGN SPECIFICATION DOCUMENT

This Product Design specification document documents the information needed by the stakeholders that are using this program to effectively define architecture and system design to give the developers a guidance on the code and system needed to be developed. This document is created during the planning phase of the development cycle. The audience is pointed towards the stakeholders, which includes the project manager, development team, client and testing team. Some portions of this document such as the user interface (UI) may on occasion be shared with the client/user, and other stakeholder whose input/approval into the UI is needed.

### 1.2 DOCUMENT CONVENTIONS

**This document uses the following conventions.**

|  |  |
| --- | --- |
| Acronym | Definition |
| DB | A database (DB), in the most general sense, is an organized collection of data. More specifically, a database is an electronic system that allows data to be easily accessed, manipulated and updated. ... Modern databases are managed using a database management system (DBMS). |
| C# | C# is a general-purpose, multi-paradigm programming language encompassing strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines. |
| .NET Framework | .NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library named as Framework Class Library and provides language interoperability across several programming languages. |
| CSV | A comma-separated values file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. |
| WMP | Windows Media Player (WMP) is a media player and media library application developed by Microsoft that is used for playing audio, video. |
| SRS | A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. |
| Mp3, mp4, wav | is a coding format for digital audio |
| Salt | In cryptography, a salt is random data that is used as an additional input to a one-way function that "hashes" data, a password or passphrase |

### 1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

The intended audience for this project is for the end users to listen to their favourite music, mp3, mp4 and wav files.

The audience is pointed towards the stakeholders, which includes the project manager, development team, client and testing team. Some portions of this document such as the user interface (UI) may on occasion be shared with the client/user, and other stakeholder whose input/approval into the UI is needed.

### 1.4 PROJECT SCOPE

The purpose of this Project is to deliver a music player for the end user to listen and enjoy their music to add music files into the program. The user can log in (given by the server) to the music player and users can create, read, update and delete the music files to their satisfaction.

The server will provide the user the login username and password for them to access the music player. It will have sorting, searching for music files to create a convenient and easy to use application.

### 1.5 REFERENCES

<https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database>

<https://docs.microsoft.com/en-us/dotnet/framework/>

<https://joshclose.github.io/CsvHelper/>

<https://github.com/kyle623/Project-AT3/>

## 2 OVERALL DESCRIPTION

### 2.1 PRODUCT PERSPECTIVE

This music player program stores the following

User:

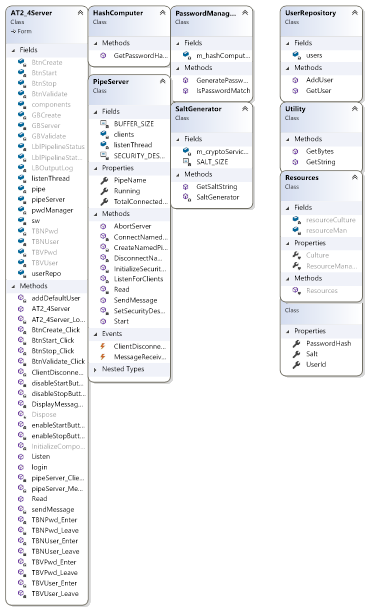
* Stores username from the server
* Stores the hashed password and salt from the server

Program:

* Stores the song path URL to a csv file.

### 2.2 Product Features

This is a sample of the class diagram of the program



### 2.3 USER CLASS and CHARACTERISTICS

Users that are given the username and password must be able to log into the music player. Users should be able to add, read, update and delete song files to play. They should be able to play, skip to next, previous and pause the music that is on the program.

Once the users have added music files into the program, they can sort the song names and search for the songs they want to search.

### 2.4 OPERATING ENVIRONMENT

The operating environment that is needed for the program includes:

* .NET Framework 4.7.2 or higher
* Running server for users
* OS: Microsoft Windows 7
* Processor: Intel Core i5-4590 CPU @3.30GHz Quad core
* storage that has a free space of greater than 200MBs
* RAM: greater than 2GBs
* Separate GPU is not needed for this game as not much graphics processing is needed and some CPUs already integrate graphics the processor.

For development (operating environment):

* .NET Framework 4.7.2 or higher
* Running server for users
* OS: Microsoft Windows 7
* Visual Studio 2017 or higher
* Csv third party library (CSVHelper)
* Windows form app

### 2.5 DESIGN and IMPLEMENTATION CONSTRAINTS

The implementation constraints for this program can be if the server is not open, the users will not be able to login, therefore they cannot use the music player. Another constraint of this application can be if the song files are deleted, the program cannot load the songs and the user will need to add the songs again to be able to play the music.

### 2.6 ASSUMPTION DEPENDENCIES

An assumption dependencies that this program can have are listed below:

* If the program tries to open and doesn’t have or the music files have relocated, there will be no music to play.
* Dependency on the server if the client (user) tries to run the program.

## 3. SYSTEM FEATURES

### 3.1 Functional Requirements

Main minimum functional 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

The functional requirements that will be implemented in this application will have a login system from a supported server, users can create, play, update and delete the music files to their satisfaction. Other features include sorting by insertion sort and searching using binary search for music files, client and server system for log in and user creations. The music player can only be accessed from client side and needs the server to be running to be able to log in and use the program.

## 4. EXTERNAL INTERFACE REQUIREMENTS

### 4.1 USER INTERFACES

* Front-end software: Windows Form app from Microsoft’s Visual Studio with .NET Framework v4.7.2 or higher
* Back-end software: .NET Framework v4.7.2 or higher.

### 4.2 HARDWARE INTERFACES

* Windows 7 or higher
* .NET Framework v4.7.2 or higher.

### 4.3 SOFTWARE INTERFACES

|  |  |
| --- | --- |
| Software used | Description |
| Operating system | We have chosen Windows operating system for its best support and user-friendliness. It also includes the requirements to make a GUI and you can only use it on windows and linux. |
| NET Framework v4.7.2 or higher. | This framework is needed to create the code and run the application. |
| Visual studio | This application is needed to create the code and UI and run the application. |

### 4.4 COMMUNICATION INTERFACES

* .NET Framework v4.7.2 or higher. For using the server to client pipe namespaces
* NamedPipeClientStream and NamedPipeServerStream classes used for communication on login system

## 5. NONFUNCTIONAL REQUIREMENTS

### 5.1 PERFORMANCE REQUIREMENTS

Normalization- The basic objective of normalization is to reduce redundancy which means that information is to be stored only once. Storing information several times leads to wastage of storage space and increase in the total size of the data stored. This is used in the application to prevent users to make duplicates of a song

Other non-functional requirements include:

* Tooltips
* Loading speed
* User interface design
* Flexibility

### 5.2 SAFETY REQUIREMENTS

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

### 5.3 SECURITY REQUIREMENTS

Security systems needs a login system where the users can get their login information given by the server and log in safely using hashing techniques.

### 5.4 SOFTWARE QUALITY ATTRIBUTES

AVAILABILITY: The Music player should be available when the server is online.

CORRECTNESS: The program follows all requirements.

MAINTAINABILITY: The code can be maintained easily as it follows the coding convention. Music can be maintained by using the features.

USABILITY: The Music player should satisfy the user’s needs