Ryley Bilson

SM TAFE

**Programming 3**

**Assessment 3 – Project (AT3)**

24/06/2021

Table of Contents

[1 Planning Document 2](#_Toc89770810)

[1.1 What data structures are you using? 2](#_Toc89770811)

[1.2 Where are you using hashing techniques? 2](#_Toc89770812)

[1.3 What sorting algorithm are you using how this is different from selection and bubble sort? 2](#_Toc89770813)

[1.4 What search technique are you using? 2](#_Toc89770814)

[1.5 What third party libraries are you using? 2](#_Toc89770815)

[1.6 Where can I find the documentation for this? 2](#_Toc89770816)

[1.7 GUI Mock-up 2](#_Toc89770817)

[1.7.1 Window – Employee Mode 2](#_Toc89770818)

[1.7.2 Window – Launch Mode Selector 3](#_Toc89770819)

[1.7.3 Window – Manager Mode 3](#_Toc89770820)

[1.8 What source control are you using? 3](#_Toc89770821)

[1.9 What are your coding standards you are enforcing? 3](#_Toc89770822)

[1.10 What tests are you going to run? 3](#_Toc89770823)

[2 Product Specification Design Document 4](#_Toc89770824)

[2.1.1 Quality Assurance Practices 4](#_Toc89770825)

[2.2 System Architecture & Design 4](#_Toc89770826)

[2.2.1 Requirements 4](#_Toc89770827)

[2.3 Forms/Windows 4](#_Toc89770828)

[2.4 Class Design 4](#_Toc89770829)

[3 Testing Documentation 5](#_Toc89770830)

[3.1 Test Table 5](#_Toc89770831)

[3.2 Debug 5](#_Toc89770832)

# Planning Document

## What data structures are you using?

A queue.

## Where are you using hashing techniques?

Data will be saved alongside a hash. Hash match required to edit loaded file.

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

Quick sort will be used in this solution. Selection and bubble sort both look for a value that belongs at a respective end and slowly fill each element onwards with the correct value. Quick sort quickly divides a collection into weakly sorted partitions then repeats on each partition until partitions are of size one.

## What search technique are you using?

Binary search.

## What third party libraries are you using?

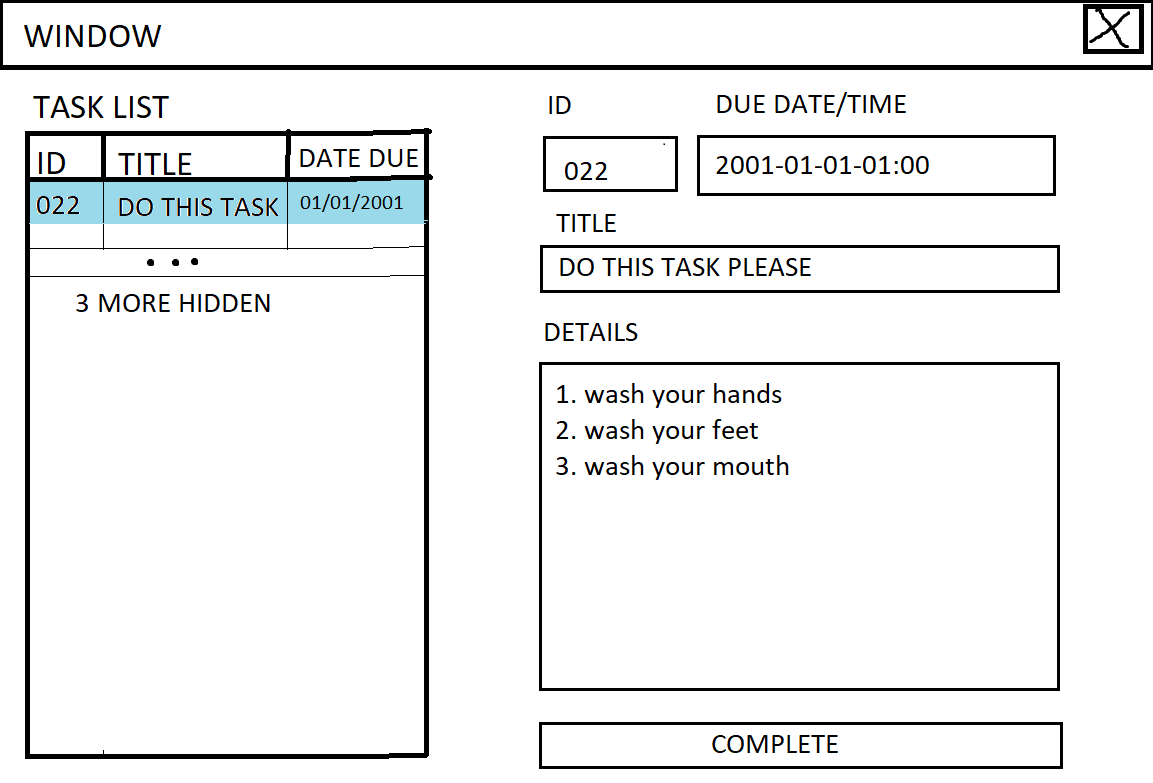
(In-House) Binary File Writer.

## Where can I find the documentation for this?

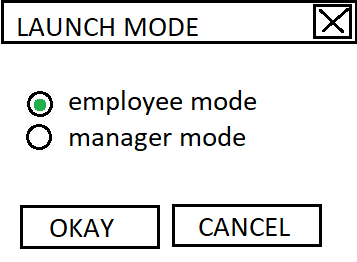
N/A. Feed a filepath and object, writes the object to a .dat file at the given location.

## GUI Mock-up

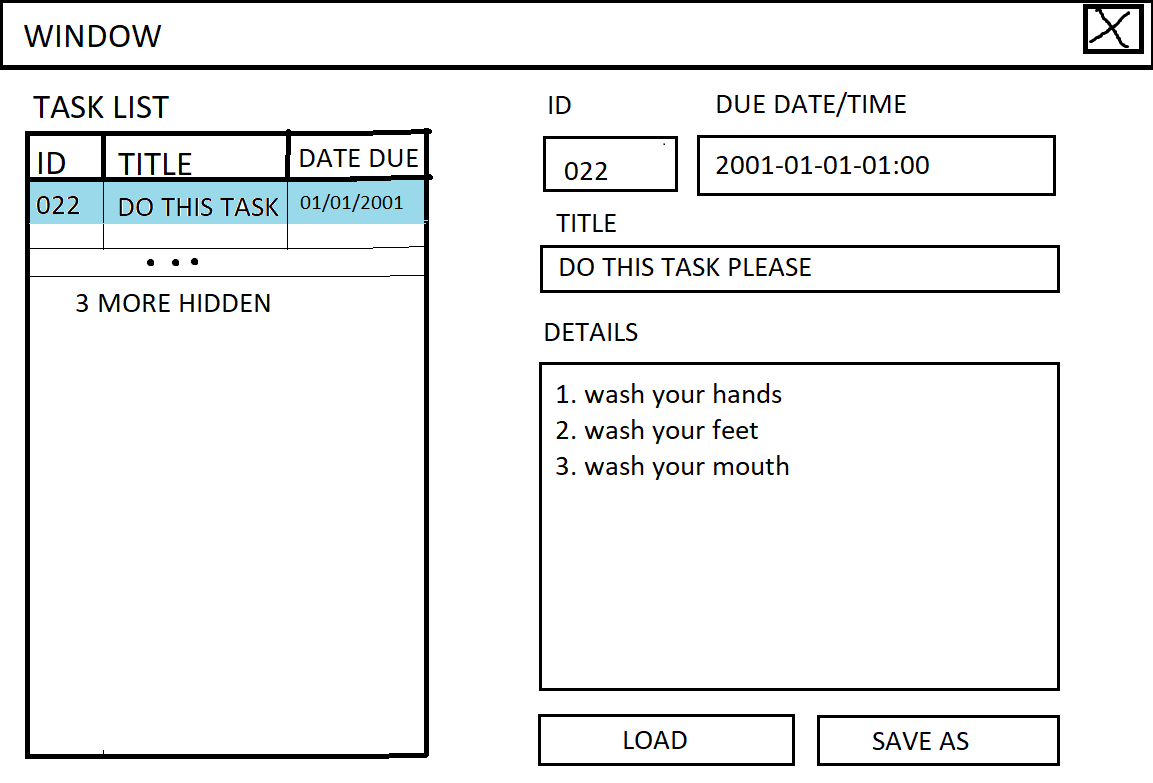
### Window – Employee Mode



### Window – Launch Mode Selector



### Window – Manager Mode



## What source control are you using?

Git via github.com.

## What are your coding standards you are enforcing?

Microsoft Docs C# Coding Conventions: <https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>

## What tests are you going to run?

* Save file
* Load File
* Enter correct password
* Enter incorrect password
* Sort object
* Search object
* Add to object
* Remove from object

# Product Specification Design Document

### Quality Assurance Practices

**Coding Standards**

Microsoft Docs C# Coding Conventions: <https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>

## System Architecture & Design

### Requirements

* Must contain dynamic data structures
  + Data structure will be a queue
* Must contain hashing techniques
  + Data will be saved alongside a hash. Hash match required to edit loaded file.
* Must contain sorting algorithm
  + Sorting algorithm will be quick sort
* Must contain searching technique
  + Searching algorithm will be binary search
* Must contain 3rd party library
  + 3rd party library will be a binary file writer/reader
* Must have a GUI
  + GUI will be .NET WPF
* Must adhere to coding standards
  + Microsoft Docs C# Coding Conventions: <https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>

## Forms/Windows

Note: see 1.7 for mock-ups

* Launch Mode Selector (initial)
  + Employee Mode
  + Manager Mode

## Class Design

|  |
| --- |
| Queue |
| * nodeList : List<QueueNode> * head : QueueNode |
|  |

|  |
| --- |
| QueueNode |
| * data : Task * nextNode : QueueNode |
|  |

|  |
| --- |
| Task |
| * id : int * dueDateTime : DateTime * title : String * details : String |
| + DueDateTimeString(void) : String |

|  |
| --- |
| ProtectedFileObject |
| * fileData : Object * hash : byte[] |
|  |

# Testing Documentation

## Test Table

Note: images may be found in file named “screenshots” in project’s containing folder.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Case | Input | Expected | Actual |
| 1 | Save to file | Filename=”test\_case.dat” | Save success | As expected |
| 2 | Load from file | Filename=”test\_case.dat” | Load success | As expected |
| 3 | Enter correct password | n/a | Success message, editing enabled | As expected |
| 4 | Enter incorrect password | n/a | Failure message | As expected |
| 5 | Sort queue | n/a | Queue sorted success | As expected |
| 6 | Search queue (success) | Target ID=9 | Search success, row highlighted | As expected |
| 7 | Add to queue | n/a | New row added | As expected |
| 8 | Remove from queue | n/a | Row removed | As expected |

## Debug

