A screenshot of a computer

Description automatically generated with medium confidence

**Coursework: 1 Library Management System**

**2023-24 CST2550 Software Engineering Management and Development**

**Coursework 1: Library Management System**

**Module Teacher:**

**Ahmed Eissa - Module leader (Hendon), Adam Philpot (Hendon), Olugbenga Oluwagbemi (Hendon)**

**Created by Arbaaz Ali Hussain - M00872279**

**Submission date: 22/01/24, 11:30pm**

**Middlesex London University Computer Science School**

**Word Count:**

Table of Contents

[Introduction 3](#_Toc156836597)

[Design: 3](#_Toc156836598)

[Implementation, including: 3](#_Toc156836599)

[Testing approach: 3](#_Toc156836600)

[Software demonstration: 3](#_Toc156836601)

[Conclusion: 3](#_Toc156836602)

[The Game Design 4](#_Toc156836603)

[Wireframe’s screenshots 1 & 2 the “Game page”. 4](#_Toc156836604)

[Wireframe’s screenshots 4 the “Login page”. 4](#_Toc156836605)

[Wireframe’s screenshots 6 the “Leader board page”. 4](#_Toc156836606)

[Wireframe’s screenshots 8 the “Contact us page”. 5](#_Toc156836607)

# Introduction

Within the project I created a Library management System for Coursework 1, the way this program is for the librarian to access the library management system, where they will be accomplishing numerous responsibilities & task that will need regularly updated and preserved within the Library Management System itself. The various tasks that would be accomplished by, adding members, issuing books, returning books, displaying borrowed books, and calculating find which have been overdue for returning and as well the system constantly and consistently tracing all books, which are in use, and in the library system or borrowed by members.

Also, this presentation will consist of relevant information on the library management system, explaining in detail and overall functionalities. The first part will consist of the design, where will be discussing various diagrams such as the UML class diagrams, use case, diagram, and activity diagram, and where will explain it utilisation and is relevancy to the code. The reason for using the time is to provide a plan and a visual reputation to have an idea to create the library management system easy to understand plan of action. Second part consist of the implementation as will be discussing, it is useful to use and Makefile, and discussing on the important version control to avoid error and the possibility to rehash old code or replaced with new existing code this done on GitHub in the repository. The third part will consist of the testing approach that has been used and implemented within the code, further, discussing how various test cases are utilised for preventing error handling. Fourth part consist of software demonstration where this is conducted in a video discussing how the library management system works and looks like in real life. Finally, the conclude will the discuss whole the project and is limitations of that I faced when created the system and how would I approach a similar project in the future.

# 

# Design:

A diagram of a library system

Description automatically generated

Figure 1 – UML Use Case Diagram

The diagram is the Use Case Diagram for Library Management System with one actor that is called “Librarian”, that can execute various actions such as the following: Add Member, Issue Book, Return Book, Display Borrowed Books & Calculate Fine. Here the diagram shows how the Librarian is able to accesses all functions and the method do so.

A screenshot of a computer

Description automatically generated

Figure 2 – UML Class Diagram

Here is a Class Diagram which display the blueprints/schematics of the system, within the diagram it portrays all the methods and its attributes that is used in the system. It also elaborates the relationships between the different objects’ relationship, such as parent and child. Inherits of the person class is a parent class to both the librarian and member which inherit the attributes from person class. As librarian and member inherit from person class, they each have their own attributes & method to their own specific functions within their own class. There’s another relationship, which is related to the classes librarian and member it is called book class. Hence, librarian and member both interact with book object when relating to the book function within the library management system.

A screenshot of a computer

Description automatically generated

Figure 3 – UML Activity Diagram

The Activity Diagram it is created to demonstrate/clarify the way that the functions flow throughout the library management system. The first prompt for starting program, here code will start the program or end program depending on the user’s input. Then the program prompts the user to enter a filename for example, a .csv file etc, if file is in the repository or file, it will then execute the program and read its data, if not, it will prompt user to enter a file which is in the repository.

For example. When the csv file has been entered into the system, the system/program will display a menu to the user displaying various functions that can hence be executed by the user if chosen. If the user wants to end their program, they must be in menu selection of the code, then it is possible to exit the system at once.

The way to add member is by selecting add member, they the program will prompt user to enter all the relevant data regarding of creating a new member. Furthermore, when the user has entered all the data, the code will save the new member and display. It’s information that has been created to user. Also once created it will provide a new member a ID.

The way to issue book is by selecting issue book, when the user has entered the function, the issue book will prompt user to enter member ID will be prompt again to select their desired book through its book ID.

The way to return book is by selecting return book, user is required to enter them and ID which is necessary for the program, once they have entered the member ID, they use a must select the book they want to return through entering it book ID into the system. if the user issued the book more than it required days, the user will receive a fine, therefore if the user returns the book in its proper due date, then a user will not receive any fine. Also, there’s another function that is used in return book which is called calculate fine here it will calculate the number of days that the user has kept the book for the length of the book was kept in the due date. There will be no fine applied if the user returned the book after his due date for example, three days anything exceeding is due date will incur a fine of £1 per day.

The way to display the board books is by selecting display borrowed books, when the user entrance enters their member ID, the system will check if the user has borrowed any books currently so if yes, the program will display all the books as borrowed by the user if not, the process of the program will end.

And finally, if the user wants to exit the program, they can exit the program in the menu selection.

# 

# Implementation, including:

The method that has followed is through utilising the class diagram, here we can improvement the base of the code. In the diagram, it has shown that the person class is the parent thus we can use it as a foundation for a code with attributes of the name address and email with its consistent accessor mutator methods. Librarian and member are child to the parent class, meaning that they are a subclass of person, this return allows both classes to inherit all attributes and method from person class. In addition to this, the librarian member class has their own Ribes and methods that can be used with the code. Book class is also utilised alongside librarian and member class, it too, has its own attributes and methods which is shown in the UML Class diagram.

Regarding the Use Case diagram, it is used to visualise and demonstrate how the visualisation in the program in the perception of how the user will interact as the librarian within the system itself.

Furthermore, the activity diagram is principally a method that is robust to comprehend and discuss with ease, allowing for each action to be understood peace by peace how the program will operate if so, action has occurred.

The Makefile is file which is utilise for automation of the compiler and the program build process. With the Makefile it has identified the way to compile the code and allow it to link with the program, thus its commonly has an established of instructions and particular requirements for the program. This certifies that it only the crucial aspects of the project that are recompiled when changes have occurred this return, resources and reduces latency.

The way that the version control where different types of same the variation of code is tracked and managed through GitHub. This drastically provides improvement overall on the performance and quality of code and its scalability of code. If a problem/mistake has occurred, it is easily avoidable and can be returned to previous code version. Finally, through committing various code/files it allowed for seeing all coding processes and development cycles,

Here is my commit history for the project in GitHub:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a black screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Testing approach:

The way that I have tested my library management system is via, implementing a testing method, through making sure that all the classes are appropriately working without error. Thus, this method allows for every individual function of its class to be tested before implementing it in its final system.

In the code I have created a test case for every single class for testing separately each function. Therefore, it is comprised of testing both its functionality and handling errors regarding its data. Through this, we can ensure that each part of the system is operating as expected in its specific function that is assigned to.

Test case further aspects in detail:

* The Person class: The test case is developed to test it getter and setter method for their attributes such as name, address, and email. This involves not only confirming accurate data retrieval and attribute updates but also scrutinising incorrect data to ensure robustness.
* The Librarian class: Testing of this class focused on specific Librarian attributes such as staff ID and salary, as well as attributes inherited from the Person class. Both inherited getter and setter methods and methods specific to the Librarian class were tested.
* The Member class: Similar to the Librarian class, Member inherits from Person. The tests developed for the Member class were intended to evaluate both native and inherited functionality. This comprises of getter and setter methods for member-specific attributes. Member ID and attributes inherited from the Person class, such as name, address, and email. These tests verified the correct functionality of these methods in relations of precise data retrieval and data updates. Additionally, the tests also proven how the class handles malformed data.
* The Book class: Testing the Book class was aimed at evaluating the functionality of the getter methods for attributes such as, bookID, BookName, BookAuthorFirstName, BookAuthorlastName, bookType, and isIssued This includes testing the getter method for the attribute.

# Software demonstration:

For the software demonstration, this will be demonstrated in the video that will be posted in the repository, further describing the program in detail, and showing the live version of the library management system.

# Conclusion:

The in this conclusion, I developed and create the Library Management System which is develop for the of librarians to a system that robust and practical, for them to manage numerous tasks that is a daily occurrence. As mentioned before the librarian can add members issue, books, return books, calculate fine and display borrowed books of each member within the system. In regard to this project, I have initially developed a plan. Which consist of the following diagrams which are use case diagram, activity, diagram, and UML class diagram. As for the creation of the code, is consisted of developing classes and methods.

Implementation includes developing classes and methods based on current diagrams, automating the compilation of files using the Makefiles, and version control to manage and track updates in the code. We also included detailed test cases to evaluate the class and provided software demonstrations in video format to demonstrate the functionality of the code.

One of the limitations of my code was that I struggled to make my Mac Terminal to run the testcase file, I have it run for Windows but not for mac I found this difficult implement, and which resulted in me not completing this task. Due to a lack of knowledge and experience of multiple platforms operating systems. Hopefully this can be addressed in the future.

In the future, I plan to use a different testing approach and spend more time fully understanding the testing approach & researching different library that can be used for the code. Finally demonstrate my knowledge for demonstration in a more efficient way and understanding way.