

A logo with a shield and text

Description automatically generated

**CST2550**

**Software Engineering**

**Management and Development**

**Coursework 1**

**Project: Library System**

**Date Of Submission:15/01/24**

A red and black rectangle

Description automatically generated

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

A red and black rectangle

Description automatically generated

**M00935155**

Table of Contents

[Introduction 3](#_Toc156027499)

[Development Process + Design Decisions 3](#_Toc156027500)

[Early development stage 3](#_Toc156027501)

[Mid Stage Development 8](#_Toc156027502)

[Late-Stage Development 9](#_Toc156027503)

[Lessons Learned 10](#_Toc156027504)

[Issues encountered 10](#_Toc156027505)

[Future Works 11](#_Toc156027506)

[Conclusion 11](#_Toc156027507)

Table of Figures

[Figure 1 Use Case Diagram 4](#_Toc156027438)

[Figure 2 Create Account Activity Diagram 5](#_Toc156027439)

[Figure 3 Borrow Book Activity Diagram 6](#_Toc156027440)

[Figure 4 Return Book Activity Diagram 7](#_Toc156027441)

[Figure 5 Display Book Activity Diagram 8](#_Toc156027442)

# Introduction

This project is a fully functional Library System tailored for a librarian’s use. It allows the librarian to add new members, issue books to members, handle fines calculations upon return of books and display the all of a member’s book.

This report will cover the project’s development process, design decisions, limitations and future aspects for development

# Development Process + Design Decisions

## Early development stage

At this stage, the diagrams such as the use case diagram and activity diagrams were being created with respects to the project specifications and UML class diagrams.

Below are the diagrams which were designed for the system.

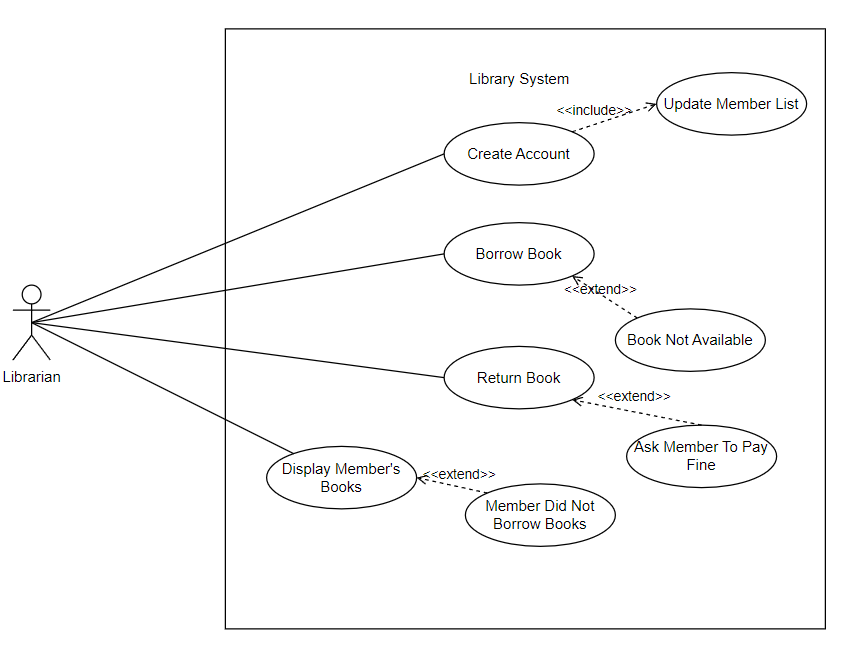


Figure Use Case Diagram

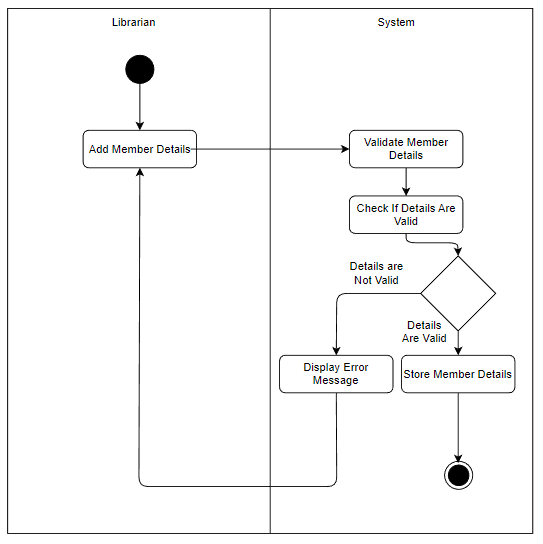


Figure Create Account Activity Diagram

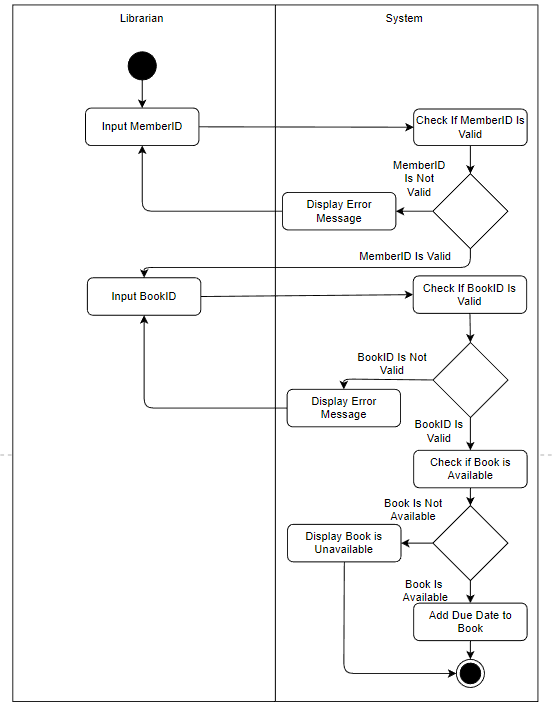


Figure Borrow Book Activity Diagram

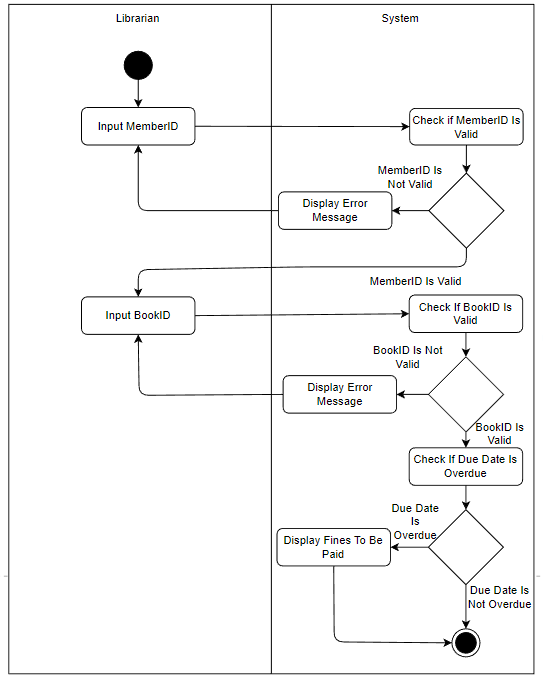


Figure Return Book Activity Diagram

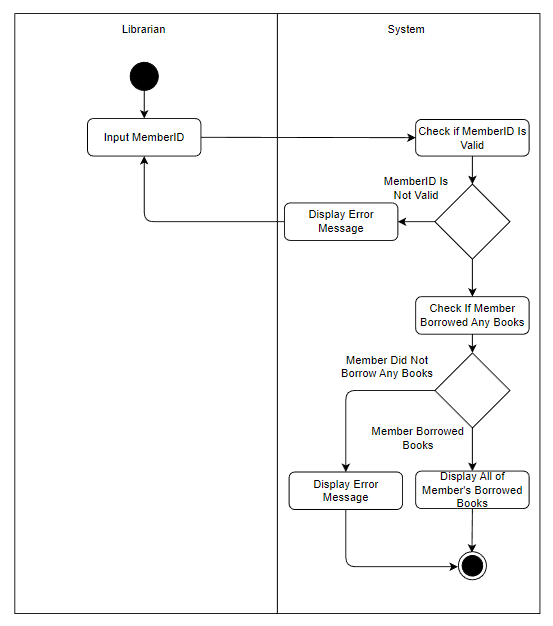


Figure Display Book Activity Diagram

After the design has been set in stone, an empty git remote repository was setup on Bitbucket and cloned on to a local folder which was used as a local repository.

Following this, a singular header file which contained all of the classes declaration as well as their methods was coded. Moreover, the classes’ constructors, getters and setters were also being coded at this point since they were relatively effortless to implement.

## Mid Stage Development

Whilst not directly in the specifications, work had been directed into allowing the Librarian to enter their details before being able to access the system. Functions related to validating the Librarian’s details such as, name, address, email, StaffID and salary were also implemented in a modular design which allowed them to be re used later on for validating details of members.

Following through, focus was directed at trying to manipulate the CSV file. The first thing which was worked on was an input which allows the user to input the path to the CSV file. Proper validations were put in place to ensure that the file existed and could be read.

After ensuring that a proper file was being read, an algorithm to extract data from the CSV file was coded.

At this point, it was decided that a vector will be used to store Member and Book objects.

After that, a menu system was designed to allow the librarian to easily perform their tasks. But during that time, it was discovered that some Books had commas within their titles which broke the algorithm which was used to extract the book data. As such, the code for extracting book data was modified to account for this.

It was also decided that the vectors which stored Member and Book objects became global variables which allowed the different methods to access them to adhere by the UML Class diagram.

Moreover, work on the first functionality, Create Member, begun and it was implemented without much issue as validations from the Librarian class were re used.

The second functionality, Issue Book, required a bit more effort since date related functions were needed to add a due date to books. As such, a function with the capability to return the current day and the date in 3 days was coded.

Since Issue Book, had been completed, the next functionality to be coded was Display All Members Books, which was used to test if a book did in fact get assigned to a member.

Finally, the Return Book functionality was coded last, however at first there were no fines calculations. Another function which made use of the aforementioned date calculating function was created which allowed for the difference in days between two different days to be obtained. This allowed for fines to be calculated.

## Late-Stage Development

After all of the functionalities were added, the next step was to comment the code to ensure it adhered to the Code Guidelines. Furthermore, adding more quality-of-life features as well as enhancing previous functionalities were in order.

Some notable features which were added are:

* Pressing 0 to exit an action
* Display Book displaying the BookID instead of only seeing the book title
* Ensuring that “clear screen characters” were present where necessary
* Changing all the STD::CIN into STD::GETLINE which allowed spaces to be input in between texts
* Adding a Regex to check for non-alphabets characters in a string

Work had also begun in creating the Makefile using automatic variables and proper compiler flags.

However, all of the work above was implemented using only one .h file and one .cpp file which made the code non-modular and was bad practice. Hence, a lot of effort was needed to refactor the code into different .h and .cpp files.

The Makefile had to be reworked as well to account for this.

After completing this, more cleanup work was in order in terms of fixing bugs and removing unnecessary code and comments.

Finally, the remaining piece of the code, the Test File, was worked on last as it required for the entire codebase to be fully functional before testing.

The test cases followed the principle of unit testing, whereby individual pieces of code were tested separately.

A final update to the Makefile was made to make it compile the test file.

# Lessons Learned

The following are what I have learnt throughout the project:

* Proper commenting practice and coding standards
* Catch2 testing
* OOP principles such as abstraction, encapsulation and inheritance
* Code organization
* Reading CSV files
* Automating Compilation using Makefile
* Regex
* Version Control using Git
* Designing Use Case and Activity Case Diagrams

# Issues encountered

Below are some of the issues which were encountered while coding the project:

* Refactoring the code into different files created a lot of incompatibilities between the different files. It created a lot of linker issues during compilation which had to be resolved.
* Some Books had commas within their titles which had to be accounted for
* The UML Class diagram was strict on the variables we could pass as parameters, so some work arounds were needed to respect the diagram.
* No specifications for Date data type led to use of a Struct data type to store date info.
* Naming convention were sometimes ignored and inconsistent in the UML class diagram.
* Use of global variable vectors to store objects of Books and Members which is not good practice.
* Some details about the books were ignored which made them redundant (e.g. page number, genre, etc).
* If the program is closed, all data is loss and has to be input once more.
* Some variables, such as MemberID, were stored as integer but were returned as string, which lead to type casting before returning the value.

# Future Works

With the experience gained from this project, there are a few improvements which could be made to tackle future works which can be found below.

* Lessen use of global variables.
* Use of a consistent and appropriate naming scheme for all of the code of the program
* Use of a database or other storing platform to ensure that data is not loss upon exit of the program.
* Performance optimisation such as proper memory management.
* Implementation of a GUI which makes the software more user friendly.

# Conclusion

To conclude, this project showcased a library system tailored for a librarian’s use as well as the design and implementation of the system while also elaborating on the limitations and future approach on future projects aspect of the Library System.