CST2550 Software Engineering Management and Development

Report for Library System

Student ID: M00912754

Email: PH526@live.mdx.ac.uk

Date of Submission: 15 January 2024

Table of Contents

Table of Figures

Introduction

The project is a library system which will be used by librarians only. The librarian will be able to issue books, return borrowed books, add a new member and view all the books borrowed by a member. The main aim of the project is to design and implement a system which can be used by someone without any knowledge of the implementation of the system and to facilitate the daily activity of the library.

The waterfall model was used for the development of the software. This involves various stages which need to be carried out in chronological order. The stages are:

- Requirements engineering
- Design
- Implementation
- Testing
- Maintenance

To be able to move onto the next stage of the project, the previous stages must be completed. In case, any changes have to be made to the software, the whole process has to be repeated starting from the previous stages.

Software Design

The software was designed using UML diagrams, mainly class diagram, use case diagram and activity diagram.

Class diagram

The class diagram below shows the different classes present in the library software and how they relate to each other.

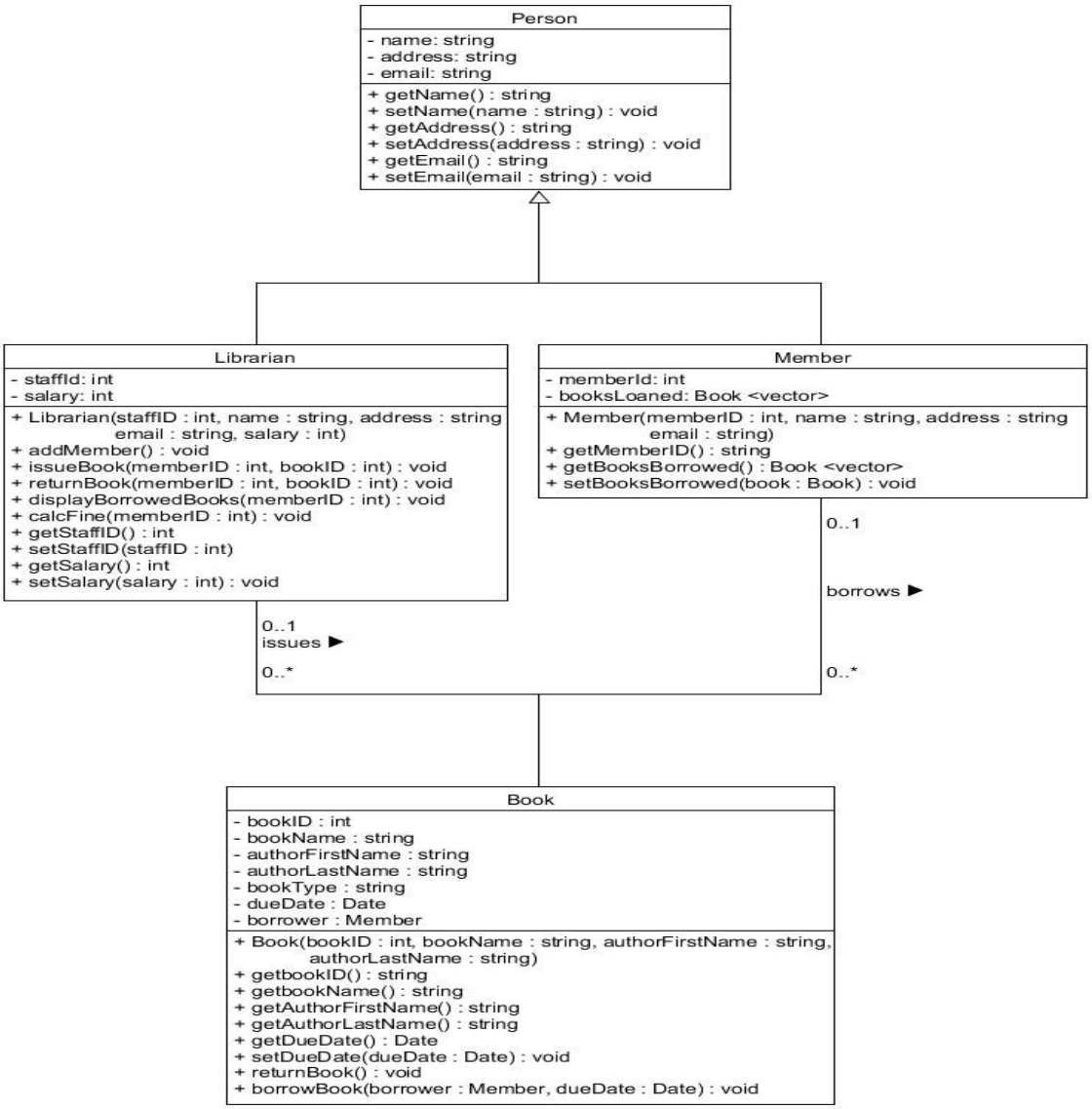

Figure 1 - Class diagram

Use case diagram

The use case diagram below illustrates the different functionalities of the library system and shows who will use the functionalities.

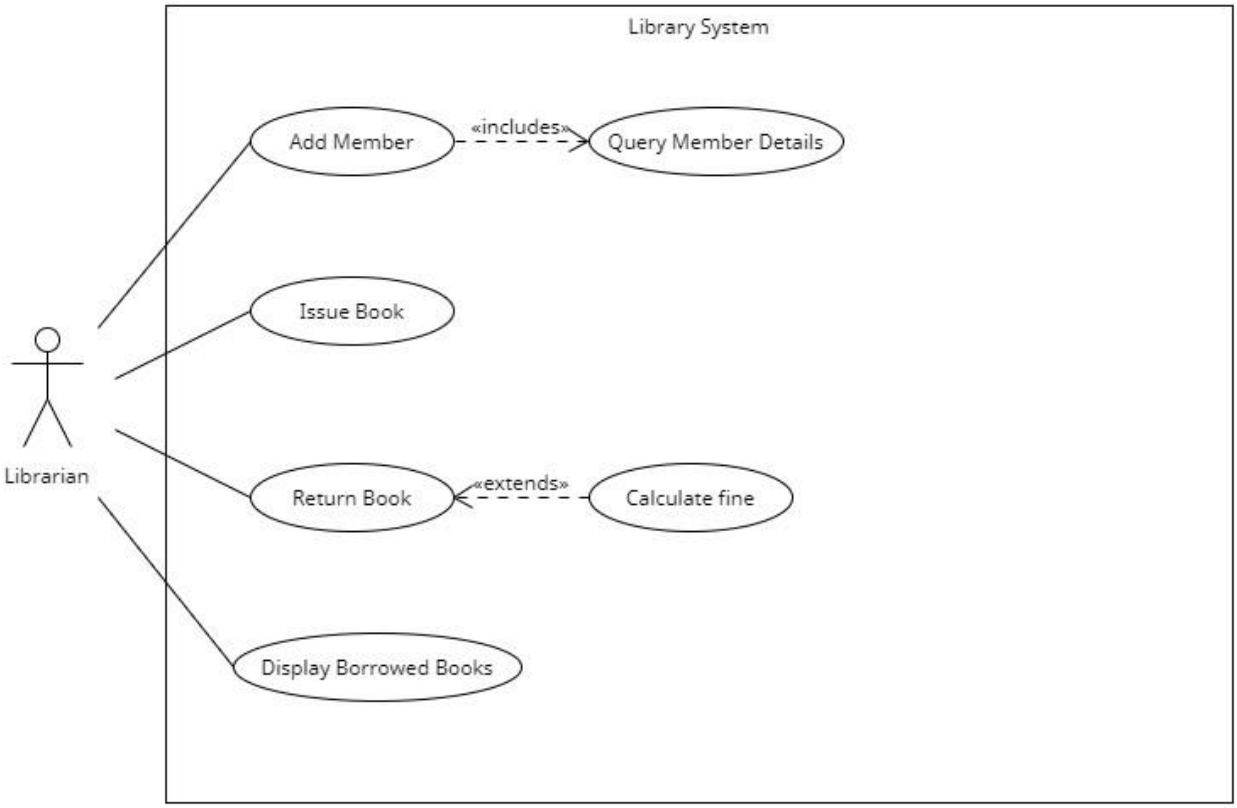

Figure 2 - Use case diagram

Activity diagrams

The activity diagrams below show the flow of actions for the different activities in the system. The different activities in the system represent the functionalities as shown in the use case diagram.

Add member

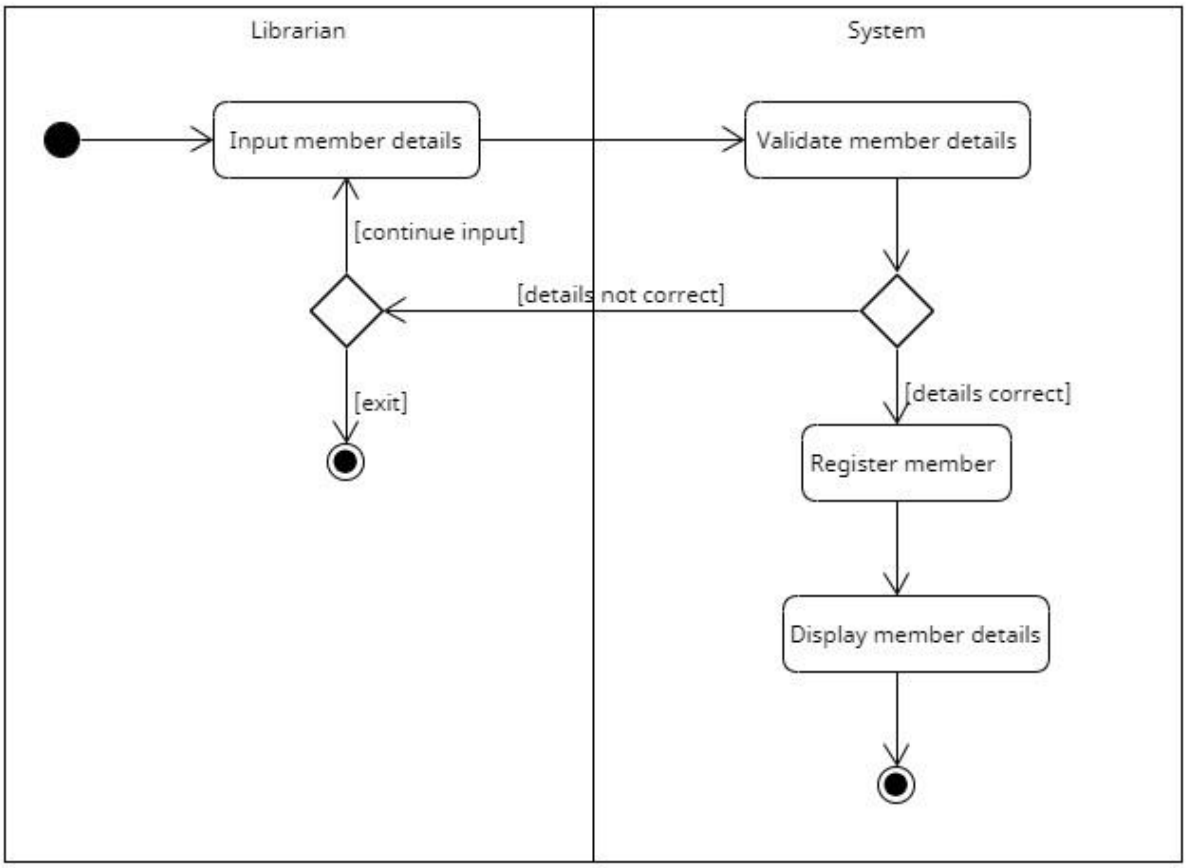

Figure 3 - Activity diagram (Add member)

Issue book

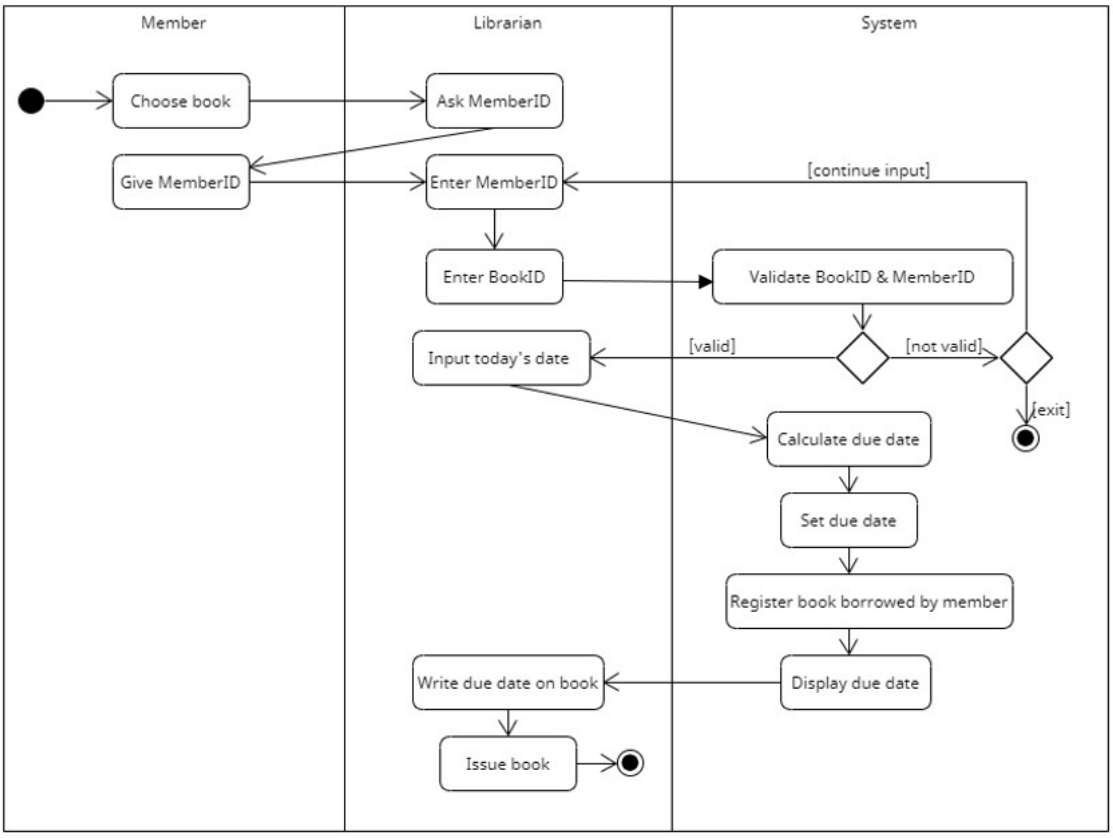

Figure 4 - Activity diagram (Issue book)

Return book

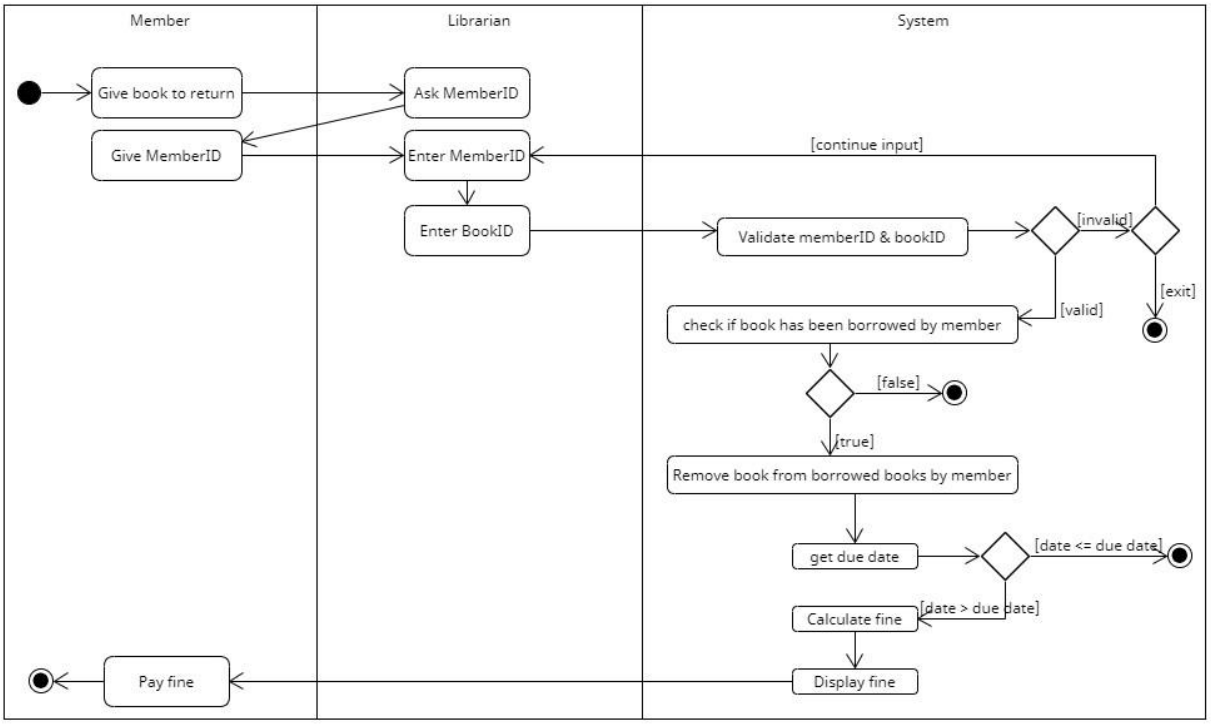

Figure 5 - Activity diagram (Return book)

View borrowed books

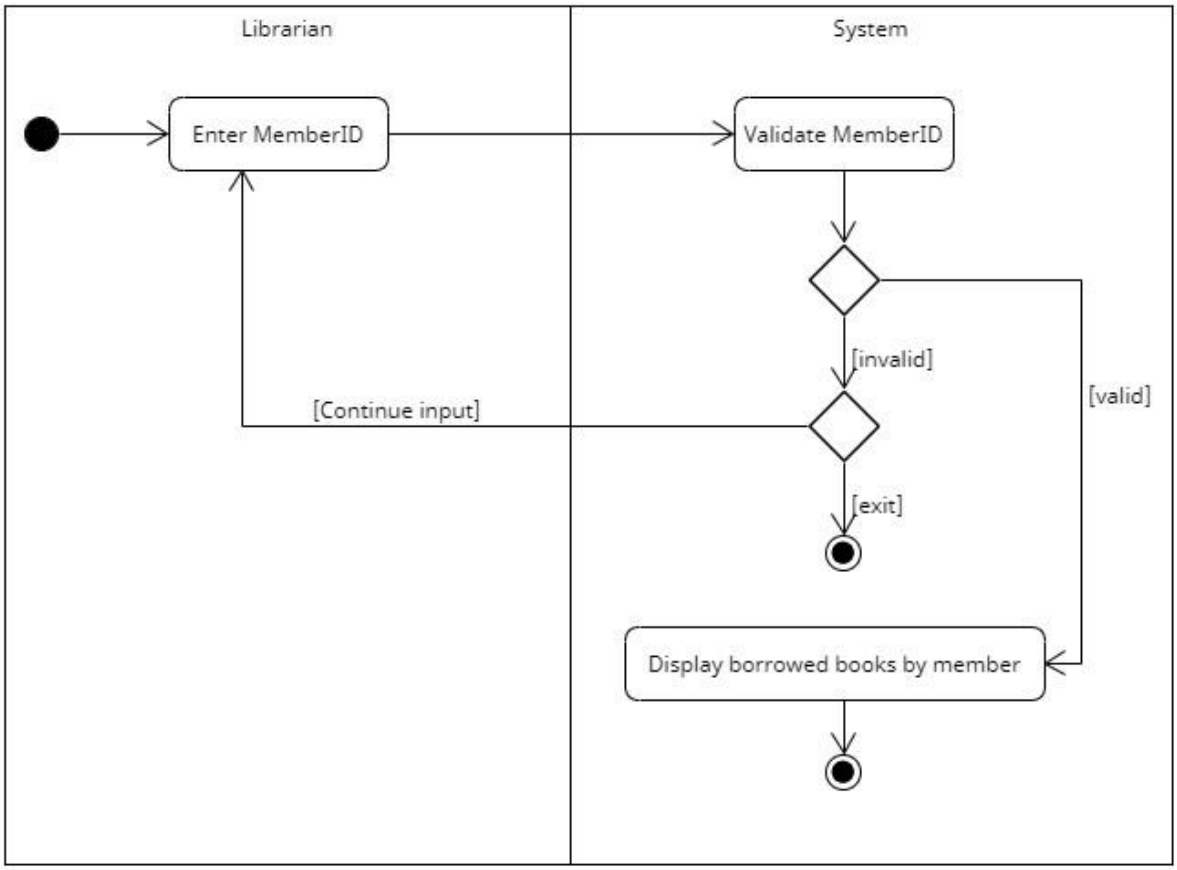

Figure 6 - Activity diagram (View borrowed books)

Software Implementation

Implementation approach

To translate the design into working software, the following steps were carried out:

1. Making the header and source files for each class present.
2. Making the makefile which is the build tool used.
3. Writing some basic test cases for the classes so that other parts of the program can be coded.
4. Coding the remaining parts of the software.

To keep track of all the changes made, git version control was used.

Build tool

As mentioned earlier, the build tool used is make. The make utility program determines which piece of the program needs to be recompiled. All the targets along with their dependencies are defined in the file named Makefile in the directory where the software is being developed. The

file consists different phony targets to achieve executable files for different targets. The phony targets and their description are as follows:

- all : produce an executable file for the software.
- test : produce an executable file to testing the software.
- clean : removes all executable files and object files from the directory.

Only the files that have changes are recompiled when make is used. Make helps automate the build process.

Version control

Git version control has been used during implementation of the system. Version control helps track all the changes made and we have access to different versions of the software during implementation. If ever a newer version of the software fails, we can always revert to the previous stable version. Git also allows software development without affecting the current stable version of the software. The developer can make a branch and add new features without affecting the main branch.

Commits

Figure to figure show all the commits made during the development of the software.

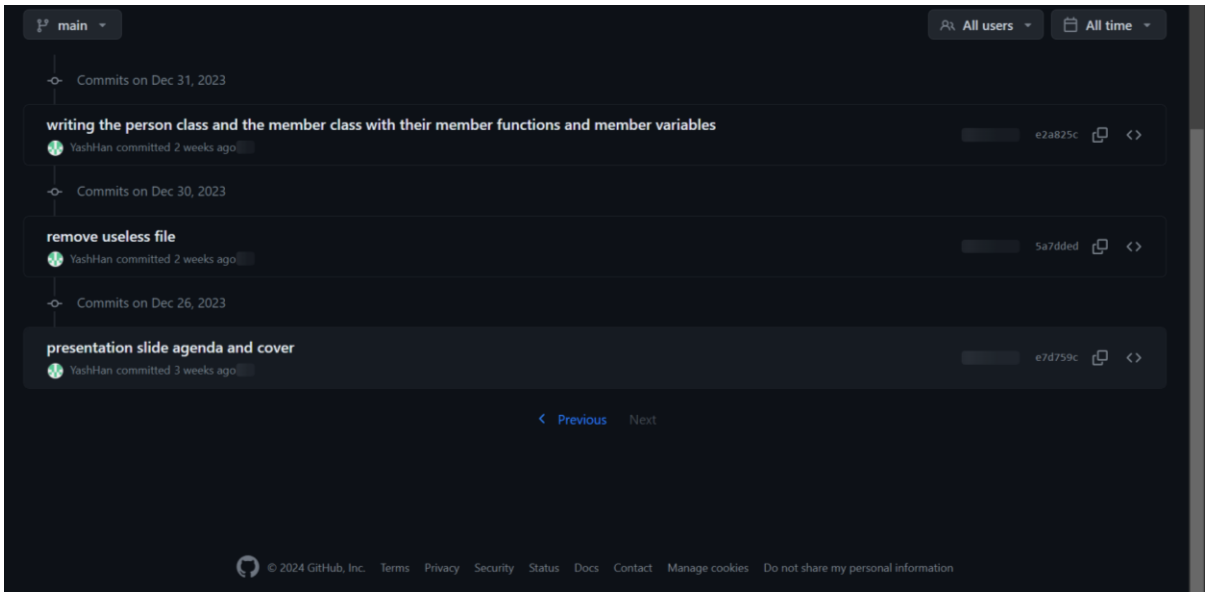

Figure 7 - commits 1

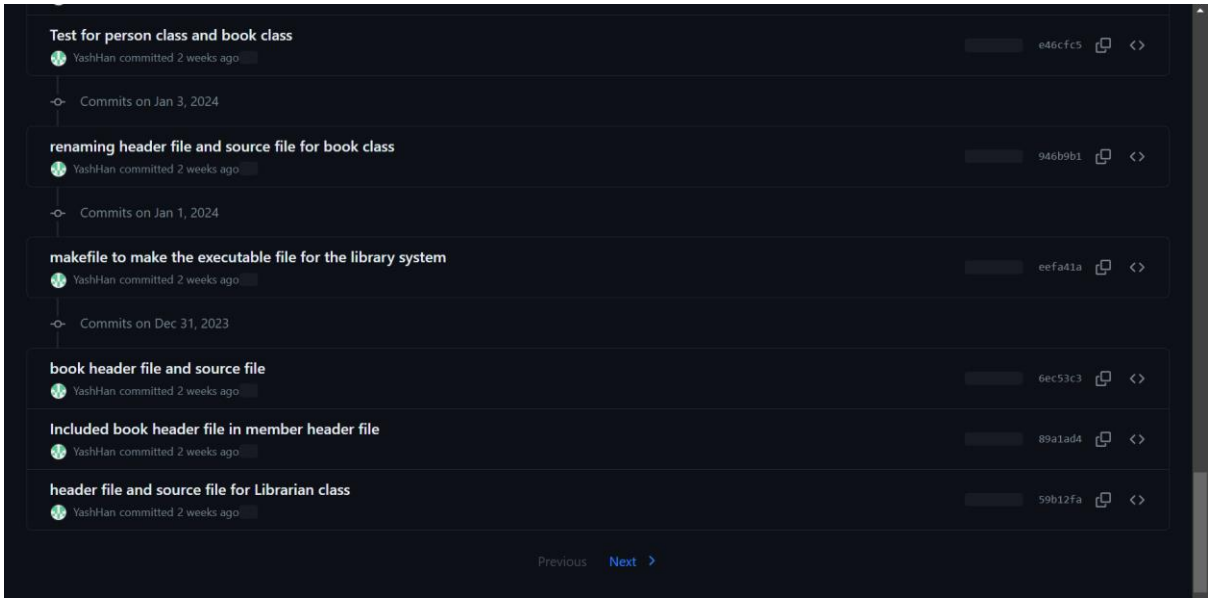

Figure 8 - Commits 2

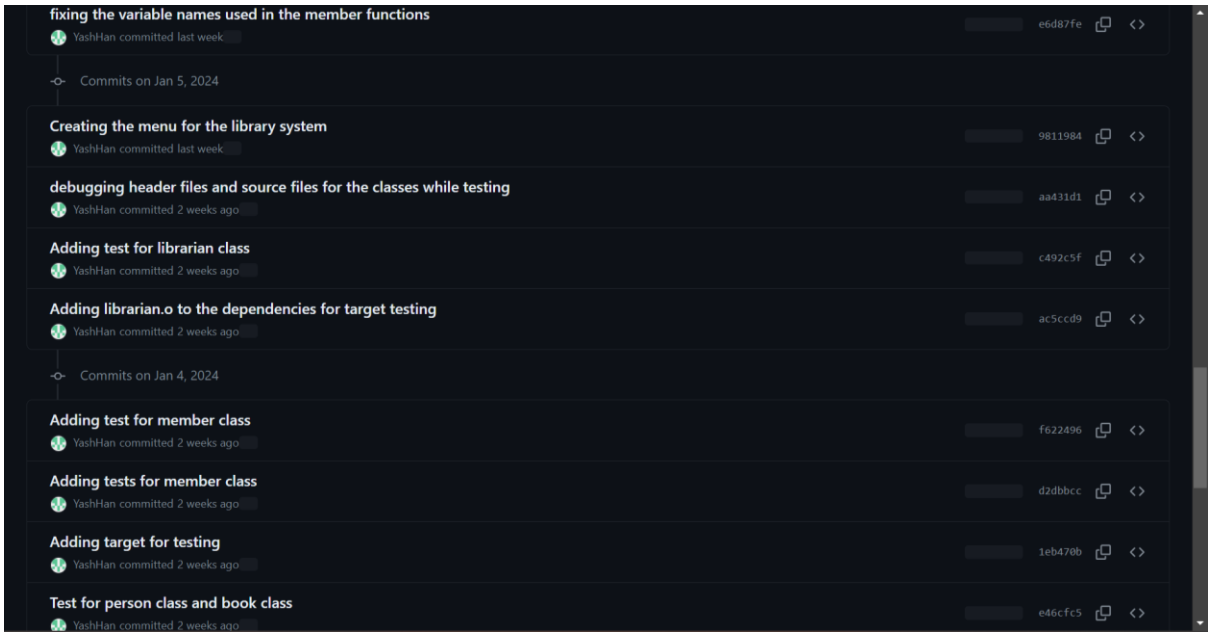

Figure 9 - commits 3

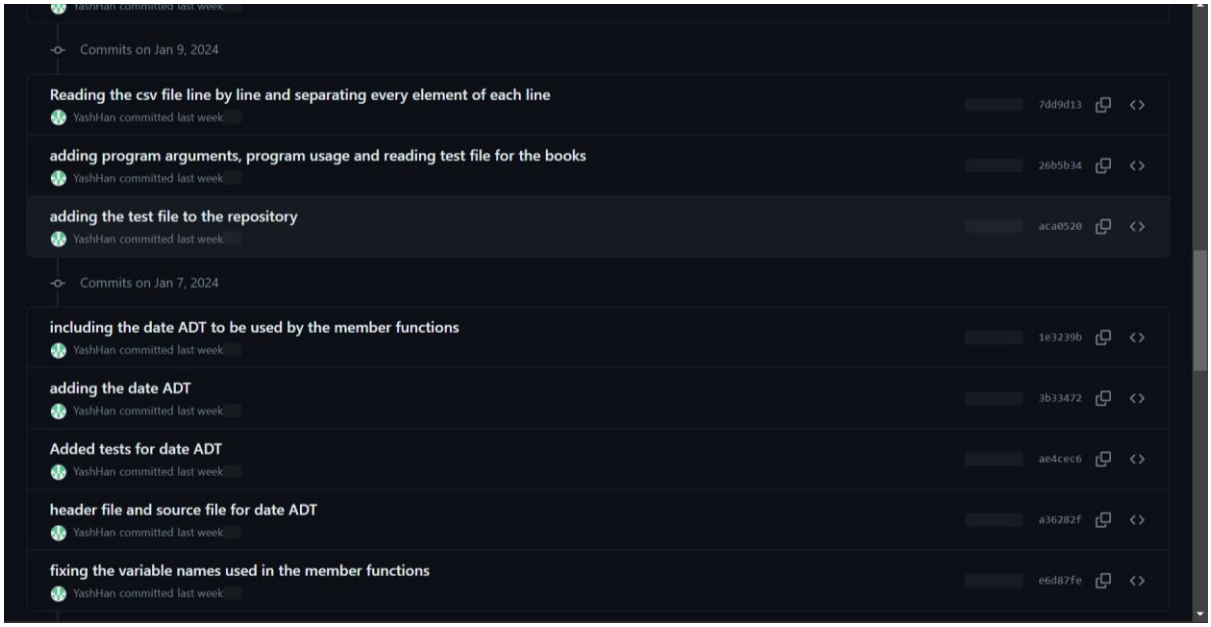

Figure 10 - commits 4

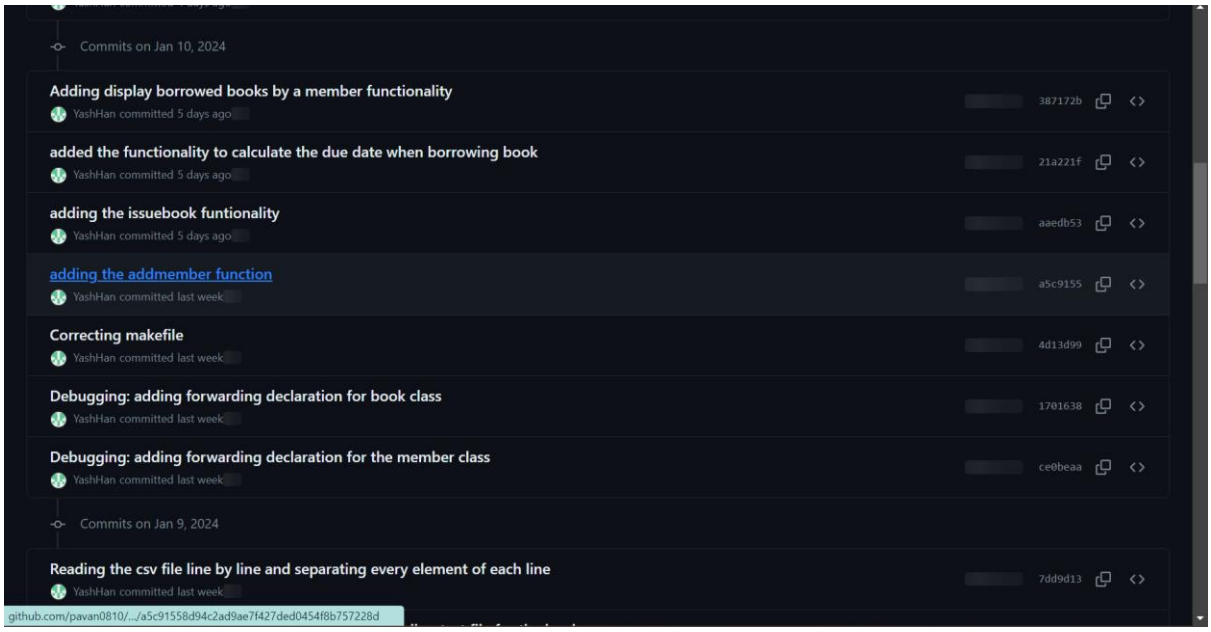

Figure 11 - commits 5

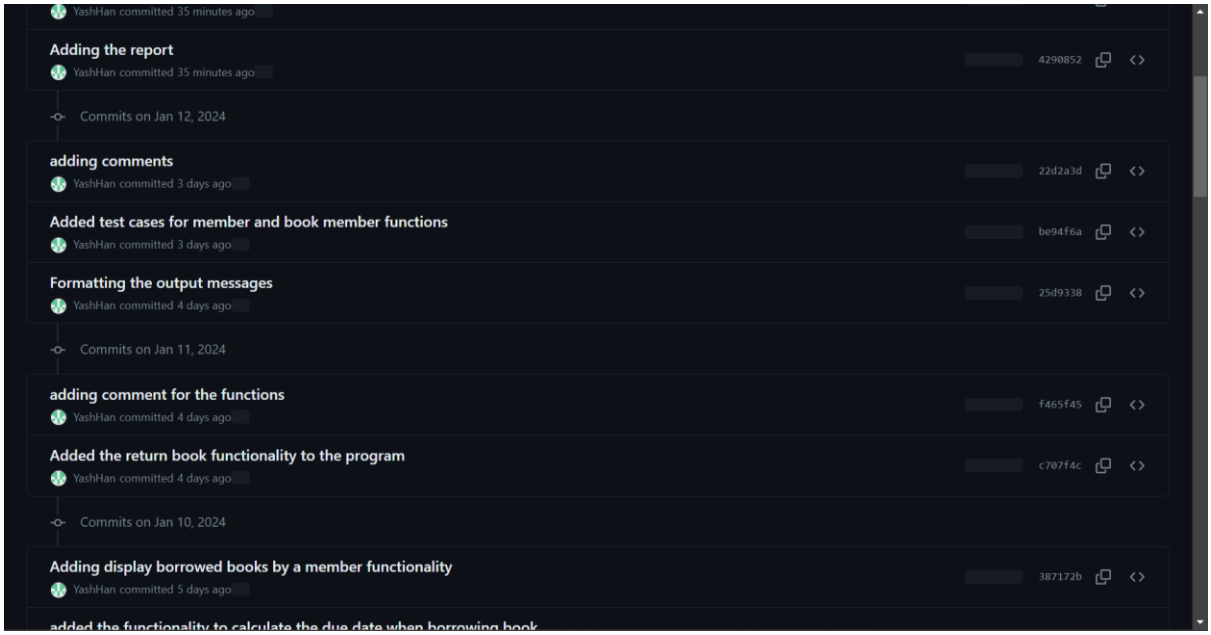

Figure 12 - Commits 6

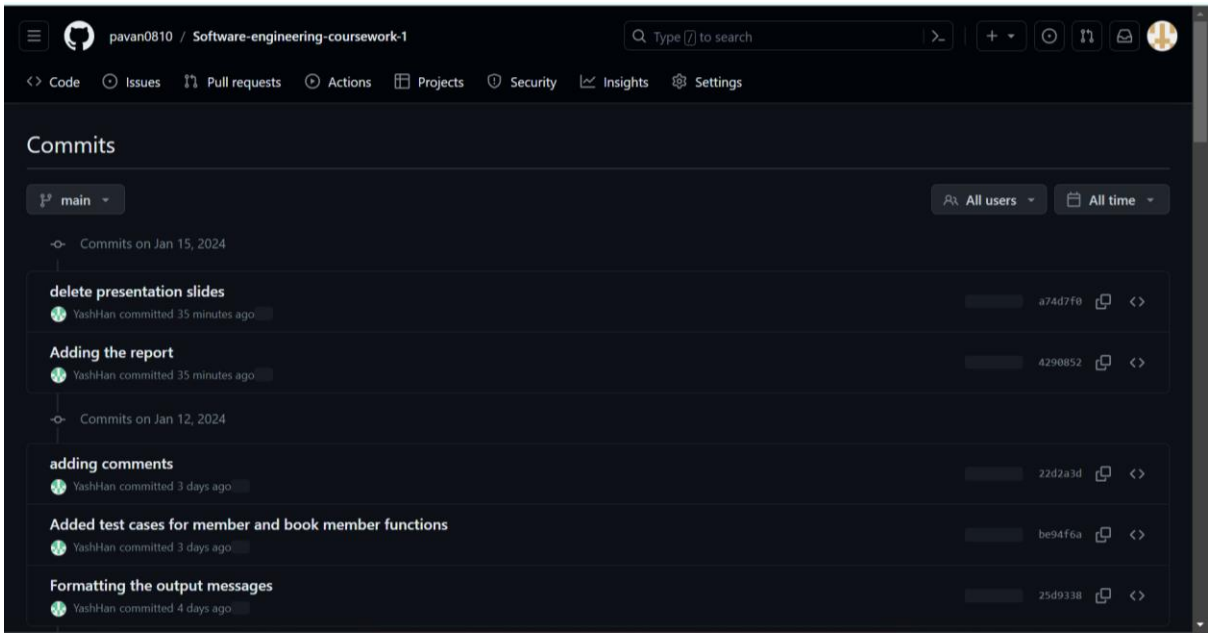

Figure 13 - Commits 7

Software Testing

Testing was carried out in three stages, mainly unit testing, component testing and system testing.

Unit testing

The unit testing was carried out to test the member functions of the different classes using Catch2 framework. Different test cases were used for each class and each test case was divided into different sections to test the different member functions. The functions which are

used for validation were tested with test data to both fail and pass the tests to ensure that the functions work as intended.

Component testing

Component testing was performed every time two functionalities that relate to each other was completed. For example, when the view borrowed book functionality was implemented, it was tested along with the borrow book functionality to check if the books borrowed by a member is displayed in his/her borrowed books list.

System testing

When all the functionalities of the system were completed, I tested the system as a whole to check whether all the functionalities of the system worked correctly as intended. All the different validations and paths of the system was tested, and the system worked as intended. Error messages were displayed when needed.

Conclusion

Summary

To summarise, many difficulties were faced during the implementation of the project and adjustments had to be made to follow the design plan. Even with the obstacles, all the functionalities of the library system were implemented successfully while respecting the guidelines and design.

Limitations and difficulties

As mentioned earlier, the project had some limitations which lead to some difficulties.

The limitations and difficulties associated are listed below:

- The classes had to be implemented according to the class diagram provided.
- Global variables were not allowed to be used to store the member objects and book objects which would have made it easier to use the objects in the member functions of the different classes. As a result, all the functionalities had to be implemented in the main program.
- A file used to store the member and book objects was not allowed to be used and the same difficulty as faced with the global variables was faced.
- Difficulties were faced when implementing the classes, that is there was a circular dependency error when trying to link the member class and the book class. Forward declaration had to be used to overcome this issue.
- Difficulties were faced when trying to test the functions in the main program.
- Despite successful implementation of the functionalities, more tests have to be carried out to improve the software as it stills contains some deficiency.

Future Improvements

For future projects, some improvements must be made to decrease the limitations and difficulties faced. The improvements are as follows:

- A better design for the software must be made prior to the implementation stage. All the possible limitations with their possible solutions must be planned before implementation.
- Different paths to design the software must be planned and the most efficient path must be chosen.
- Thorough testing must be carried out to further improve the software testing all the possible scenarios. Thus, a better version of the software will be produced.