Student Number: M00917263

A brief descripƟon of your project

**Introduction (Overview of the project)**

The goal of this project was to develop a comprehensive library management system tailored to the specific needs of a small library. I was told to create a library system that must be designed to efficiently track details related to available books and library members, providing the librarian with the ability to manage book borrowing, returns, and handle and input additional member information. The project was created in response to the library's requirement for a reliable software solution to streamline their daily functions.

To meet these objectives, we had been given a UML diagram that served as a blueprint for the system's design. I intended to adhere to this design, as written in the project requirements, as closely as possible to ensure the highest marks.

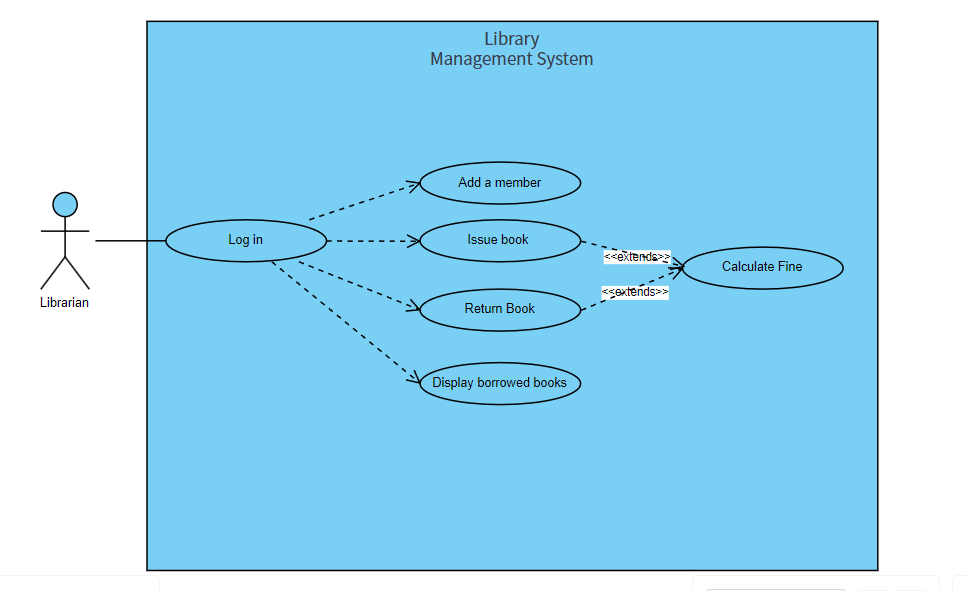
I intended to have my project allow a librarian to be able to carry out the following functionalities, as stated in the coursework brief: Being able to write down member details and store them and to obtain them when required, to be able to issue books out to a member and get given a due date from when it was issued, to show all books borrowed by the member, and calculate fines for overdue books using the due date at a rate of £1 for every day the book is past the due date. (Reference: CW1\_Final.docx, 2023)

Moreover, another requirement of the project was in utilisation of version control using Git, with regular commits pushed to a repository on GitHub, and clear concise messages detailing the changes. This practice enabled me to ensure an efficient and easy way of tracking changes, and a much more structured development process.

Design

**• Design: – Clearly show each of your UML diagrams. – You should briefly describe what the diagrams are showing (do not just read the diagrams).**

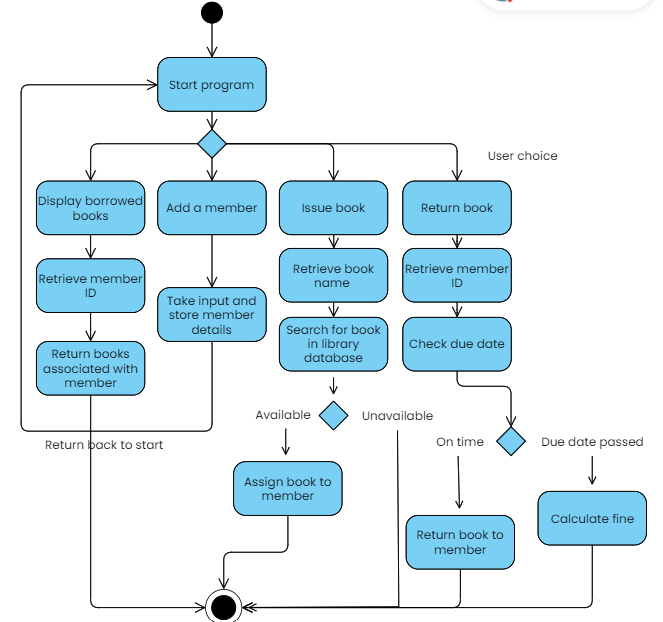
Use case diagram



A use case diagram can be explained as a graphical representation of all the user's steps throughout the program, in their interaction with the system.

* The actor, in this case the librarian, has the ability to first enter the information for the system to verify the librarian’s identity as an actual member of the library (through ID). I later decided to omit this idea in the creation of my software in order to prevent inconveniences for the user and to ensure smooth and prompt accessibility of the library management system’s features. This change effectively demonstrates the ability to empathise with the user’s wants and needs and to alter my way of thinking based on ease of access and the end goal, which in this scenario is a smooth running and efficient library system.
* The actor then has the option to choose their next steps, as prompted by the code.
* The extend relationship signifies the users optional or conditional behaviour, following both choices they would have the ability to receive a calculated fine from the program, and in the case of the “Issue Book” choice, if a book is issued to a member with another book not having been handed in, it would prompt a calculation of the previous book’s fine.

UML Activity Diagram



A UML activity diagram can be described as a flowchart showcasing all of the activities and actions performed within the system, and demonstrates the relationship and interaction between the different activities.

In my UML activity diagram, I have written all the actions the system will ideally undertake throughout the program in order to carry out the end goal represented by the final node.

* The user is first presented with a choice, which the system awaits the response for, and after is taken through a series of steps in order to carry out the final goal of that user choice like assigning a book to a member, or returning all the books associated with a member after obtaining their id.
* For Return Book, the system has an option of two possible routes after verifying whether the return date is before or after the book’s due date. If the book is on time and as per the stored due date then it will carry out the functionality to return the book to the member, and if not then a fine will be calculated.
* For issuing a book, the system has two routes it can take, either to assign the book to the associated member, if the required book is found in the library database, or to return a message signifying that the book is unavailable or not in the library’s storage and then end the program.

Implementation:

**• ImplementaƟon, including: – Your approach, i.e., how you translated the design into working soŌware. – Explain what the Makefile was used for. – Explain how and why version control was used. – Include a screenshot of the Bitbucket/GitHub repository which clearly shows all commits and commit messages.**

**ImplementaƟon (descripƟon of approach and how the makefile and version control were used)**

DESCRIPTION OF APPROACH

IMPLEMENTATION OF UML CLASS DIAGRAMS INTO CODE

My approach was an attempt at following the provided UML class diagram as closely as possible, as I wanted to ensure that the classes, methods and class relationships were translated accurately into my project. The UML class diagram served as my blueprint, which allowed for the creation of the Book, Person, Member, and Librarian classes. I also intended to have each class's attributes and methods implemented align with the specified/required functionalities as mentioned above in my introduction. While I attempted to adhere to the UML class design, there were instances where I encountered challenges or had to deviate from the diagram due to unforeseen difficulties in recreating them.

One area where I faced challenges was in the implementation of the due date logic within the Book class. The UML gave me a straightforward representation, but the practicalities of handling due dates required additional considerations, leading to a slight deviation from the original design in order to have my code run.

Another area was in implementing the issue book and return book functionalities, after several attempts I unfortunately could not find a successful way of doing so as every attempt would lead to multiple errors and so decided to omit the two attempts in order to have my code run successfully without the problem of bugs.

CREATION OF MY MAKEFILE

What was my Makefile used for

The purpose of my makefile is to facilitate the compilation or construction of my program’s source code files and classes, via a text file containing instructions in Shell on the construction of my program. Moreover, by utilising a makefile, it helped in streamlining the compilation process by allowing it to be automated and minimised the potential for manual errors by ensuring a smooth build.

I wrote instructions for compiling the source code file, and linked them to be able to compile my library management system project.

In the creation of my makefile, I created an empty text file in the directory of my source file (main.cpp) and named it Makefile. I wrote the shell script for my makefile and navigated to the directory using cd. I then ran it by entering make -f Makefile.txt and specified the makefile for increased accuracy. When using just the make command, I faced some problems so resorted to this more specific command in order to execute my makefile.

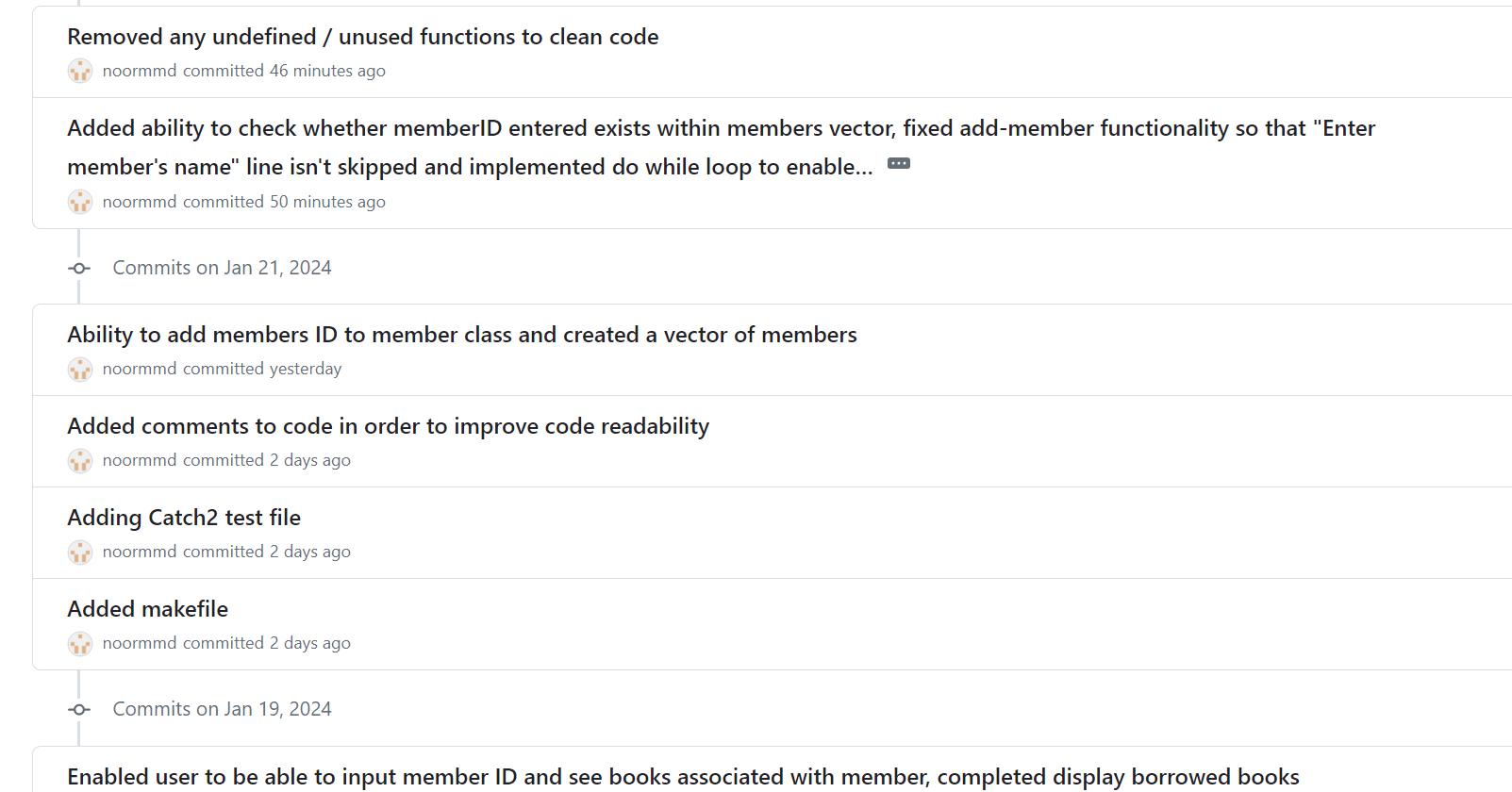
The makefile allowed me to be able to see all the errors with class ordering that were other not visible. I intended to fix these following the creation of my makefile to the best of my ability.

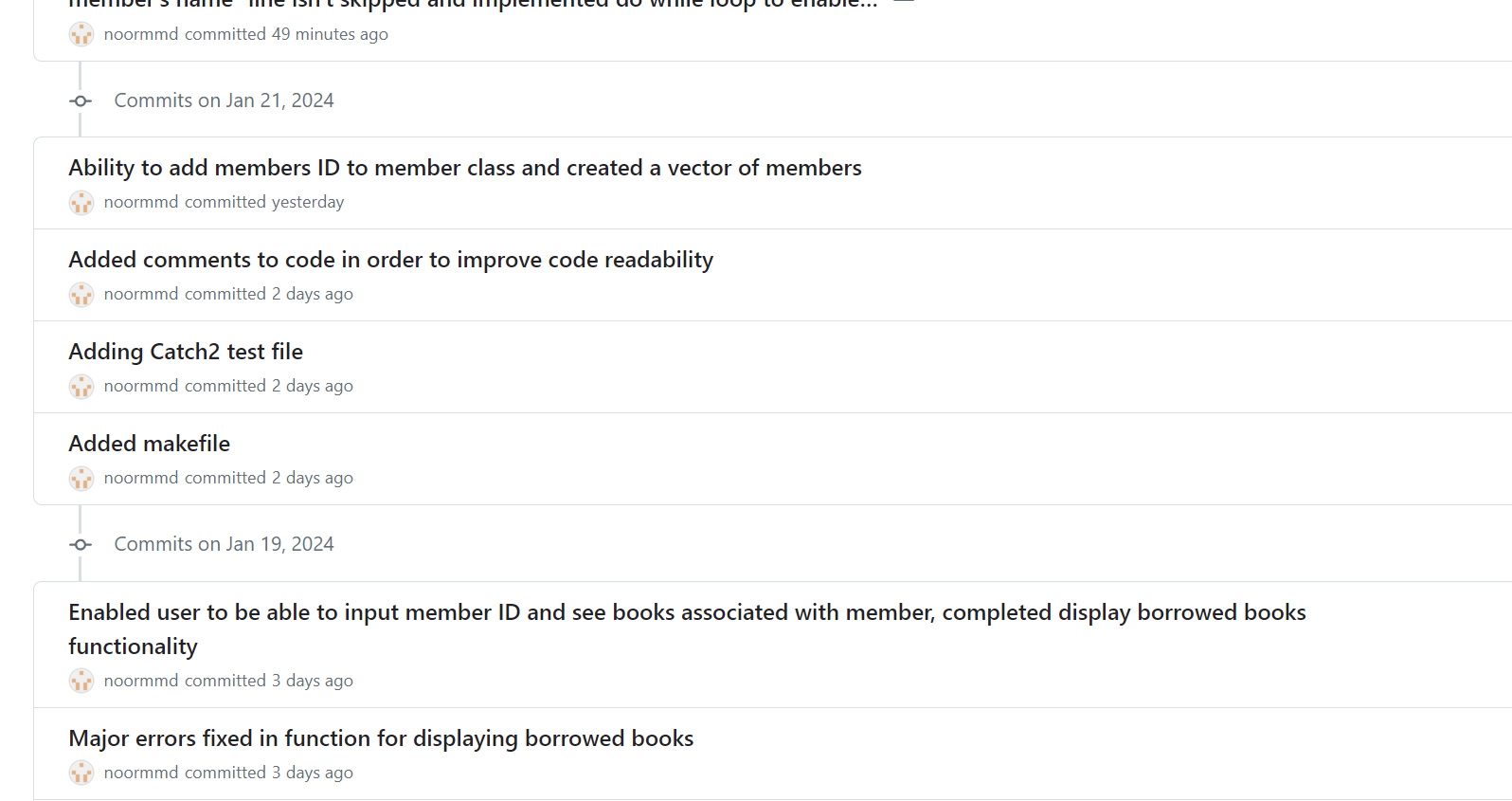
VERSION CONTROL

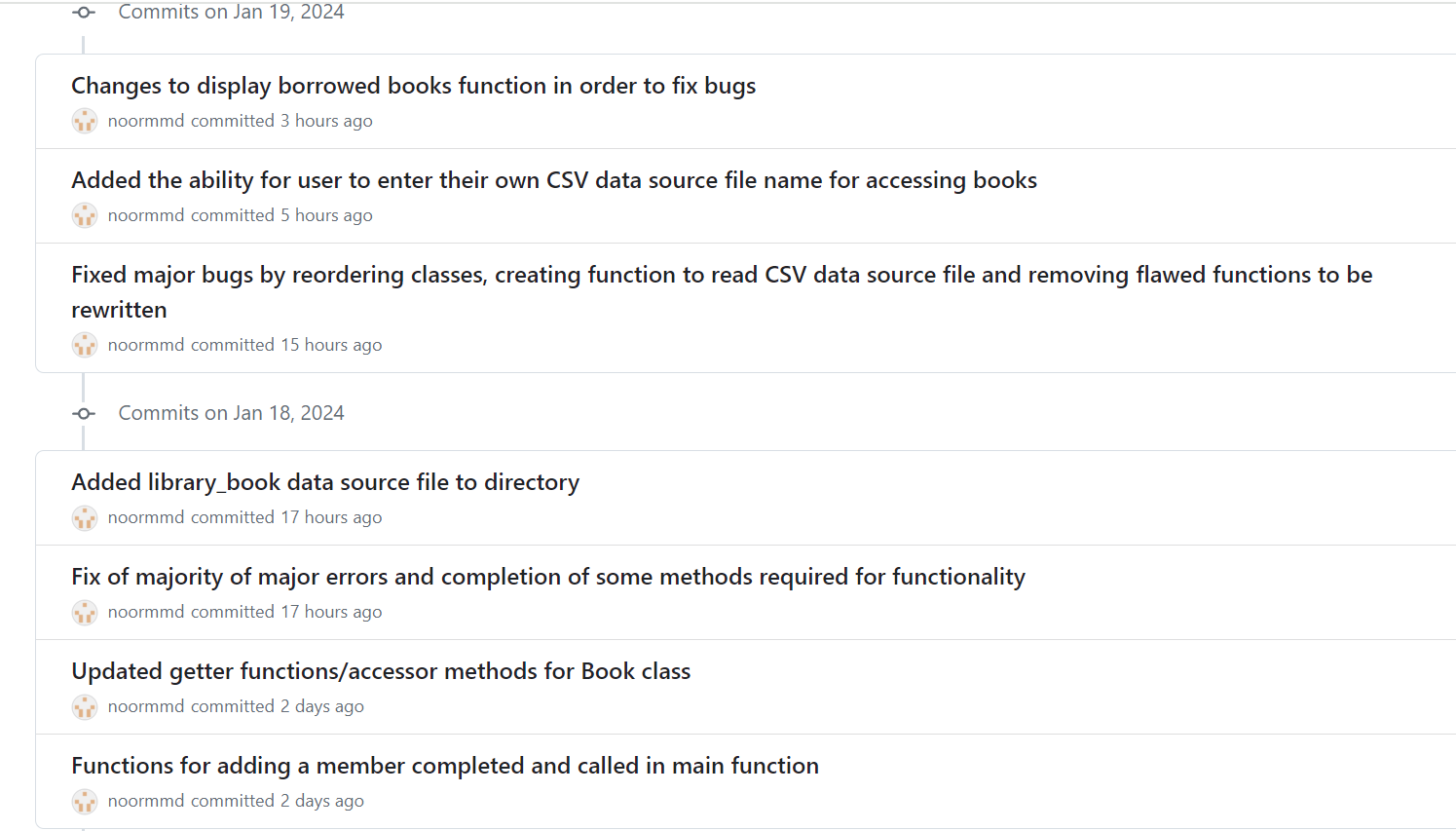
**Explain how and why version control was used.**

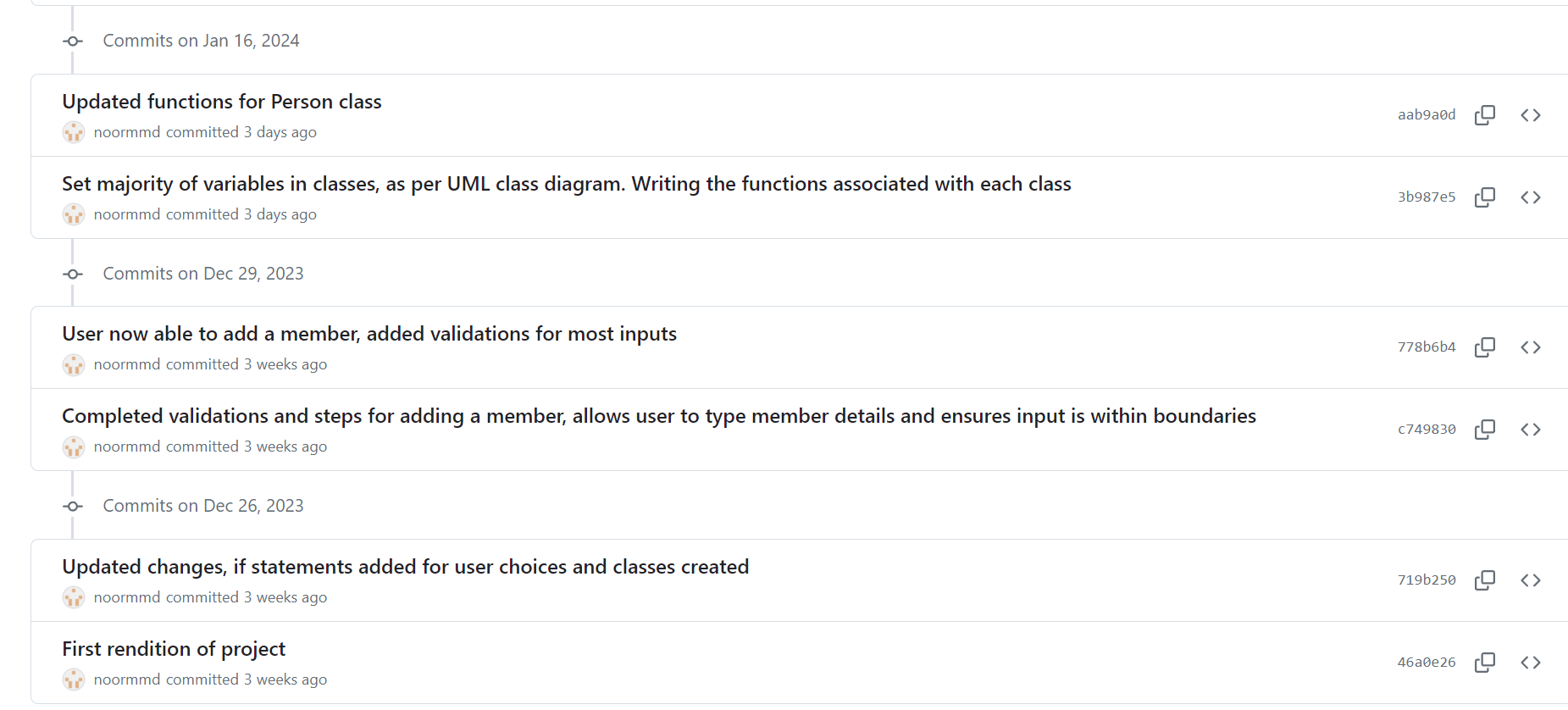
Version control was implemented using Git to facilitate the development of my project, track my changes, and maintain a more organised development history, that I can resort to whenever needed. I made sure to make regular commits, each demonstrating clearly the change that has been made and what has been worked on. This approach allowed for incredibly easy identification of changes made, bug fixing as I could resort to previous versions of my code whenever needed, and a record of the development of my project for future reference. I used commands like git commit -m "then my comment" and git push origin main in order to push and commit my changes to my github repository.

**Screenshot of the GitHub repository which clearly shows all commits and commit messages:**

****

****



****

As seen above, my commit messages were written in a clear and concise manner, describing the purpose of each commit. This ensures that the commit history is informative and aids in understanding the evolution of the project.

Testing approach:

**• TesƟng approach: – A statement of the approach used. – How you applied this approach. – Details of test cases, i.e., what was being tested (don’t show the code)**

I faced a few problems in setting up my Catch2 tests however it enabled me to see what to work on in my next coursework and that is ensuring sufficient testing. I wrote the test cases for my code regardless in an attempt to test out my testing ability and put all my tests under my library\_tests file. In this file I tested the entering member details functionality by testing the name inputs, address inputs, email inputs with both correct and incorrect inputs (matching or not matching the validation), this enabled me to practise my testing capabilities and learn the process software engineers undergo in producing and assisting in software development.

**Conclusion**

**Conclusion, including: - summary of work done - limitaƟons and criƟcal reflecƟon - how would change approach on similar task in future**

Summary of work done

In conclusion, I intended to create a Library Management System, for use by the librarians, that is easy to use, simple to follow through with and functionable, using an already provided UML Class Diagram. I created the remaining UML diagrams to help with planning and design, and intended to create the code from those diagrams, representing a professional and properly executed software development life cycle, as would be in a professional environment. I unfortunately, however, faced some setbacks as detailed below.

Limitations and Critical Reflection

POINTER CREATION AND USE, SIGNIFYING RELATIONS EFFECTIVELY INTO CODE

-Some of the limitations I faced included understanding how to demonstrate or translate relationships from the UML class diagram into my code, as well as in initialising and effectively using pointers. I faced a few problems in trying to declare pointers throughout my work, like in signifying the relationship between Member\* borrower in code.

BEING ABLE TO TRANSLATE CLASSES AND METHODS FROM UML DIAGRAM ACCURATELY, WITH AS LITTLE USE AS POSSIBLE OF EXTRA METHODS

-Most of my problems were around implementing my work in the UML diagrams into actual code, and translating what has been given on the UML class diagram accurately into my code with as little differences as possible. This was the most challenging part of the project as it required paying an incredible amount of attention to detail and working out creative ways of implementing the functions listed without adding additional methods to support it or being able to creatively find my own way of implementing the functionalities. I, however, learnt a lot throughout the task as it enabled me to practise and implement a lot of new and obtained knowledge, like a function that enables reading of an imported CSV file, based on the name of the file returned by the user.

Critical Reflection

As a critical reflection of my work, I understand the setbacks I faced in creation caused me to produce a rough attempt of a fully functional library management system with the 4 major functionalities properly executable.

It's noticeable that some of the major functionalities outlined in the assignment brief are not fully implemented or are missing. For example, the implementation of book borrowing, returning, and due date management are not present, although my attempts in another IDE might have been suitable, but were omitted for the reason that it wouldn’t allow my code to run. My code seems to lack the details required for a comprehensive library management system and I intend to fix and improve my C++ coding ability in future assignments through many more project attempts. One of the things I particularly struggled with is in coordinating the interactions between classes (e.g., Librarian, Member, Book, and Person) to properly issue and return books. I struggled with managing the relationships between these classes and hope to improve on this moving forward. I also want to pay greater attention to writing effective Catch2 test cases as I feel my attempt was not at the standard I wanted considering I had some issues in executing it.

In my attempts, I also struggled with managing book availability, borrower information, and due dates. I found it quite difficult to link these pieces of information together in a way that accurately reflects the state of the library in code, which I hope to be able to improve through development of my C++ ability.

Approaching a similar project in the future:

ATTEMPTING A SIMILAR PROJECT IN A BETTER ORDER

-I would make sure to start in order of the tasks listed instead of jumping from section to section as it made it a lot more difficult to successfully plan and recreate. Doing things in the correct order (chronologically) would allow for successful planning and iteration and would enable me to focus properly on each task at hand. This is what I intend to fix on my next project including proper planning of my coursework before development. Next time, my focus will be first on ensuring I have fully understood the assignment brief, either by summarising it in the creation of my report so that it is more easily digestible and easier to be converted into working code, then that all UML diagrams are created, fully understood and adhered to.

TRANSLATING UML DIAGRAMS INTO FUNCTIONAL WORKING CODE

-I would start with a top down approach and initialise all the classes, moving onto the functions, then combining it all together, as my approach of jumping from one piece of the project to another, or from functionality to functionality, made implementation and the production of my work quite difficult.

-Another problem I faced was in understanding time handling. I found it quite challenging to work with date and time representations knowing that properly setting and comparing due dates is crucial for managing book return deadlines in an effective and properly working library management system.