

**Software Engineering Management and Development CST2550**

**Course work 1 (Library system)**

Module Leader/Tutor: Ahmed Eissa

Tutor: Adam Philpot

Tutor: Olugbenga Oluwagbemi

Student Name: - Abhi jagdishkumar Patel

Student ID: - M00882901

**...INDEX...**

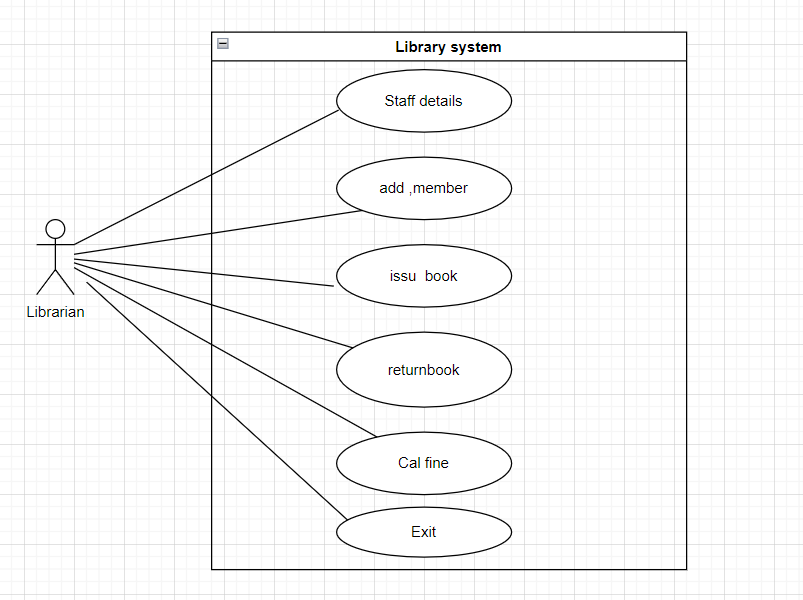
|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Objective** | **Pg.no** |
| 1 | Introduction | 3 |
| 2 | UML design | 3 |
| * Use case diagram * Class diagram * Activity diagram |
| 3 | Implementation | 5 |
| * Approach * About Make file * Why version control |
| 4 | Testing approach | 6 |
| * A statement of the approach * Details of test cases |
| 5 | Software demonstration | 7 |
| 6 | Conclusion | 7 |

1. **Introduction**

In this report, I introduce a simple and effective Library Management System created using C++. This system is designed to make everyday tasks in a library much easier, such as keeping track of which books are borrowed and returned. It has special sections for managing books, librarians, and library members. The Book part deals with all the details about the books, while the Librarian and Member parts help manage the people using the library. Our aim with this system is to help librarians organize their work better and make the library experience more enjoyable for its members. The following sections of the report will explain more about how the system works, its special features, and the code that makes it all happen.

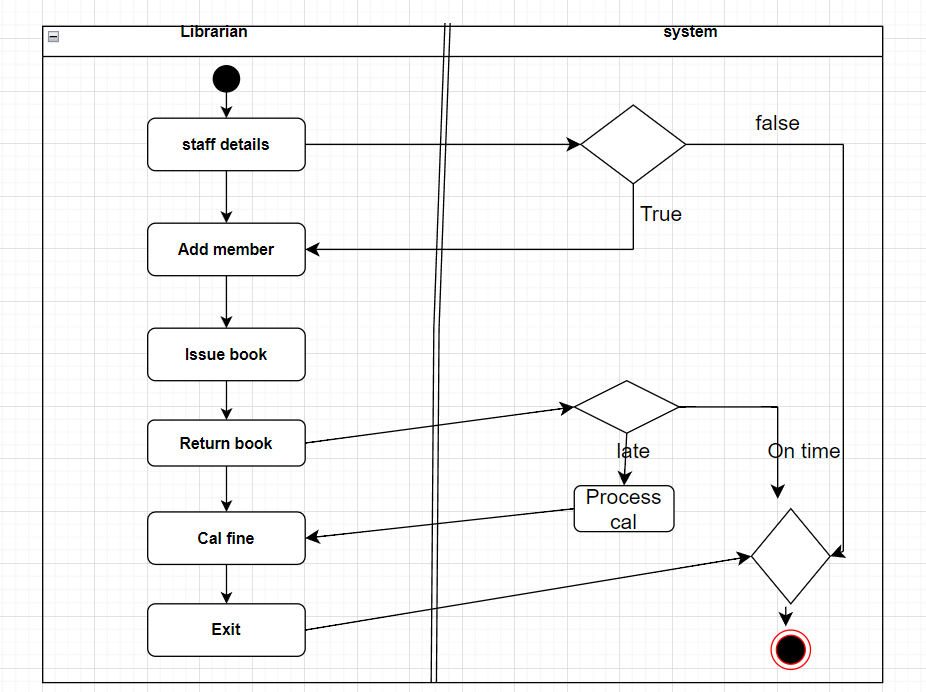
1. **UML Design**

* **Use case diagram: -**



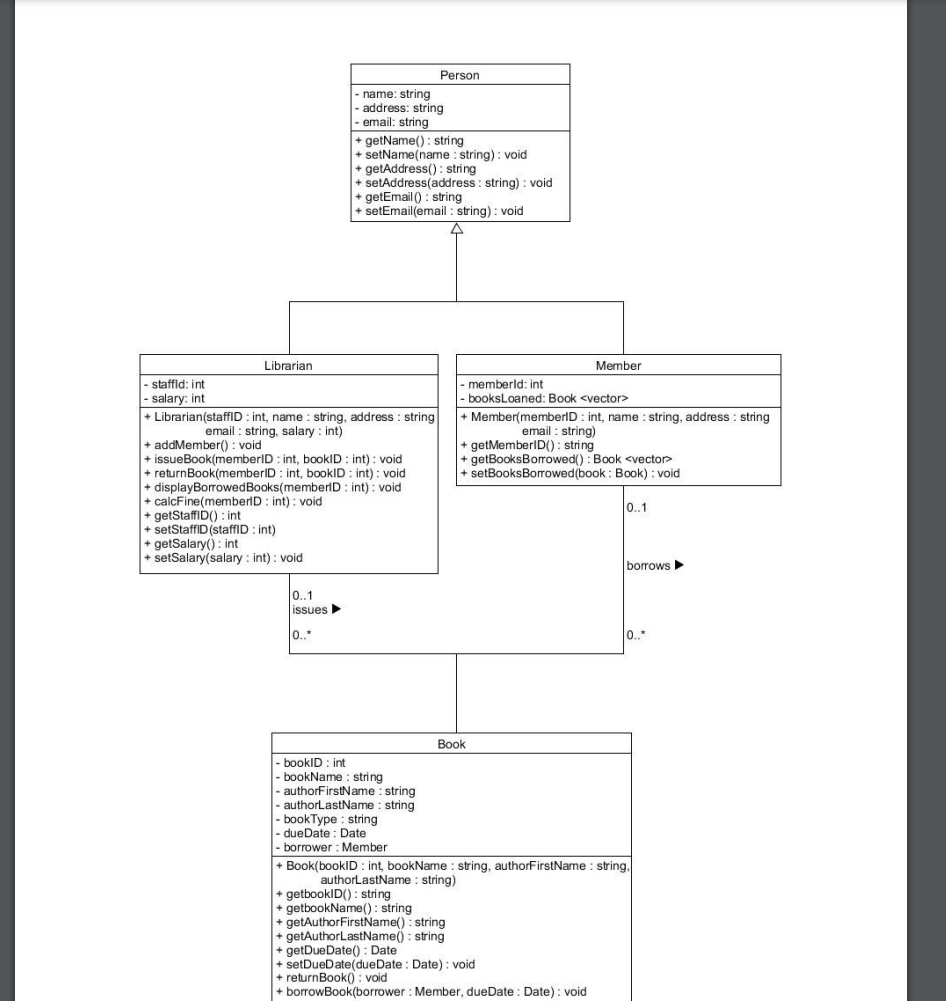
Here, you can see the use case diagram, which shows different use case that an actor (librarian) can perform with in area called library system. Here, the actor is librarian who stick in figure. Actor can do different task like he can view or updates staff details, he can add new member, he can issue book, take return book and calculate fine if books returned late. And, also close the system as well. Librarian can do all tasks within an area called library system.

* **Activity diagram: -**



the above image shows activity diagram in UML, which describe the sequence of activities performed by a librarian within a library system. A black circle shows the starting point of library. Then, the librarian starts by accessing or updating staff details. They can add a new member form “Add member”. After adding a member, the librarian can issue book from “Issue book” and return book from “Return book”. The diamond called decision point, if the book is returned late, it goes to a process labelled “Cal fine”, which likely stands for calculating the fine. If the book is returned on time, the process moves directly to the end. System calculates fine that the member needs to pay from “Cal fine”. All librarian’s activities and at exit which is shown in figure.

* **Class diagram: -**



Above picture demonstrates the class diagram which I describe below…

Person Class: This is a general class that has basic information about a person, like their name, address, and email. It has functions to get and set this information.

Librarian Class: Inherits from the Person class, meaning it has all the properties of a Person plus some specific to a librarian, such as staff ID and salary. Librarians can perform actions like adding members, issuing and returning books, and setting their own information.

Member Class: Also, a type of Person, but focused on library members. Members have an ID and a list of books they've borrowed. They can get their own information and the list of books they've borrowed, and they can set which books they've borrowed.

Book Class: Represents a book in the library. It includes details like book ID, name, author, and due date. It also keeps track of who has borrowed the book. There are functions to get and set book details, including who borrowed it and when it's due back.

**Relationships**:

The arrow from the Librarian and Member classes pointing back to Person indicates that they are both specialized types of Persons.

The "borrows" line between Member and Book shows that a member can borrow zero or more books, and each book can be borrowed by one member at a time.

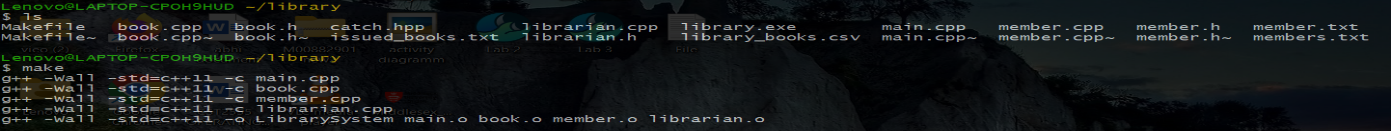
1. **Implementation**

In this course work my approach to make code for library system is easy to under stable between librarian to manage books and library’s member. The librarian could do important tasks like adding new members, giving out books, and taking them back when members return them. If a book comes back late, the system helps the librarian figure out the fine. The main is to make model easy for everyone.

* **About Make file: -**

Developing this system, I use Make file, which is very useful for building and testing my software. It’s a special file that tells the computer how to put together our program from the different classes that we had written. We don’t need remember all stuff like how execute different file together, using this file just make then show magic all file execute together I upload image how my terminal looks like when I type make and make clean. make clean command removes all the generated file, allowing for a clean build state.

When I type make command then execute all my file automatically



And, when I type make clean command, remove all the generated file.



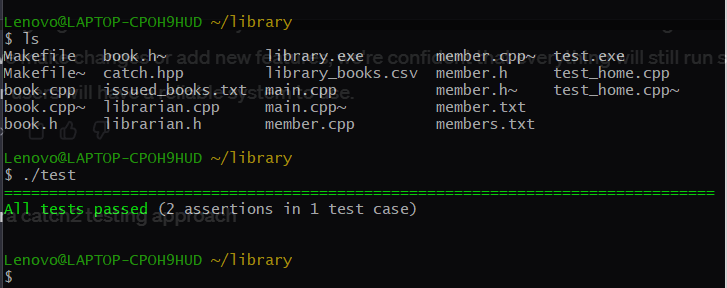
* **How and why version control was used: -**

Version control is a very useful system that we used. it helps us to keep track of changes we make to our file over the time. It saved and records our works who made changes, what changes were made, and when they were made. We use version control because it allows us to look back at earlier versions of our work, which is really helpful if we need to find out when a problem started or if we want to undo changes without losing any work. It also makes it easier for many people to work on the same project at the same time because it manages everyone's updates and makes sure they don't conflict with each other.

1. **Testing approach**

In our project, we employ Catch2, a modern C++ testing framework, to validate the functionality and reliability of our library management system. This framework allows us to write test cases that mimic a wide range of scenarios a librarian or a member might encounter, such as registering a new member or checking out a book. With Catch2, we write simple yet comprehensive tests that can automatically check whether each part of our system is performing as expected. By integrating Catch2 into our development process, we've established a robust safety net that helps us quickly identify and resolve issues, ensuring our system remains stable and user-friendly. This proactive testing approach underscores our commitment to quality and provides peace of mind that our software meets the high standards our users deserve.

I also use this approach here is my all-test output



1. **Software demonstration**

First, I understand your given class UML diagram I compare this diagram as real life then, set up one algorithm like what I add in different class like member, librarian, book then I implement different UML diagram and test my code one by one class. By breaking down the system into smaller, manageable pieces, we were able to focus on one component at a time, ensuring each part was built correctly and worked well on its own before integrating it with the rest of the system.

1. **Conclusion**

At the end, I made one software which is control by librarian. Also, the librarian can manage and interact with member easily with in different case. they can add member, issue book, return book and calculate fine. But, at a first glance there are some limitations in program. in future want to overcome this limitation surly. The limitation is build strong login function for librarian so that only librarian uses this program and also my effort is going for everyone use this software (joining with bank details and automatic cut off payment.

I would like to work in future on student management system and banking loan system may be these are similar to this project.