Middlesex University Mauritius

CST2550

Software Engineering Management and

Library Management System
Coursework 1 – Project Report

Summer term
2023/2024

Date of Submission: 15th January 2024

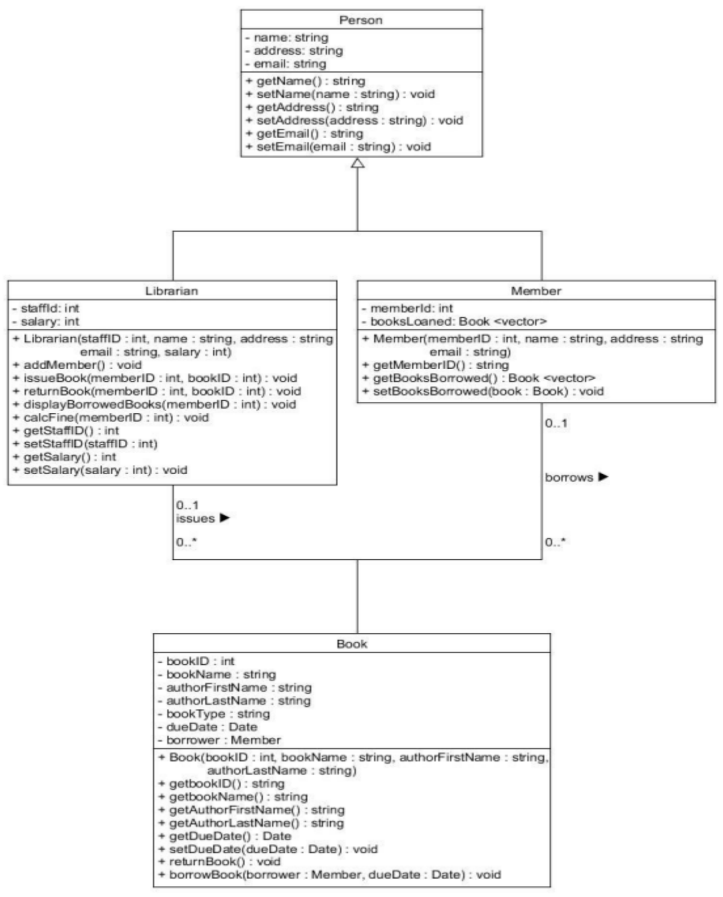
Student ID Number: M00857430

Lab Tutor: Aditya Santokhe

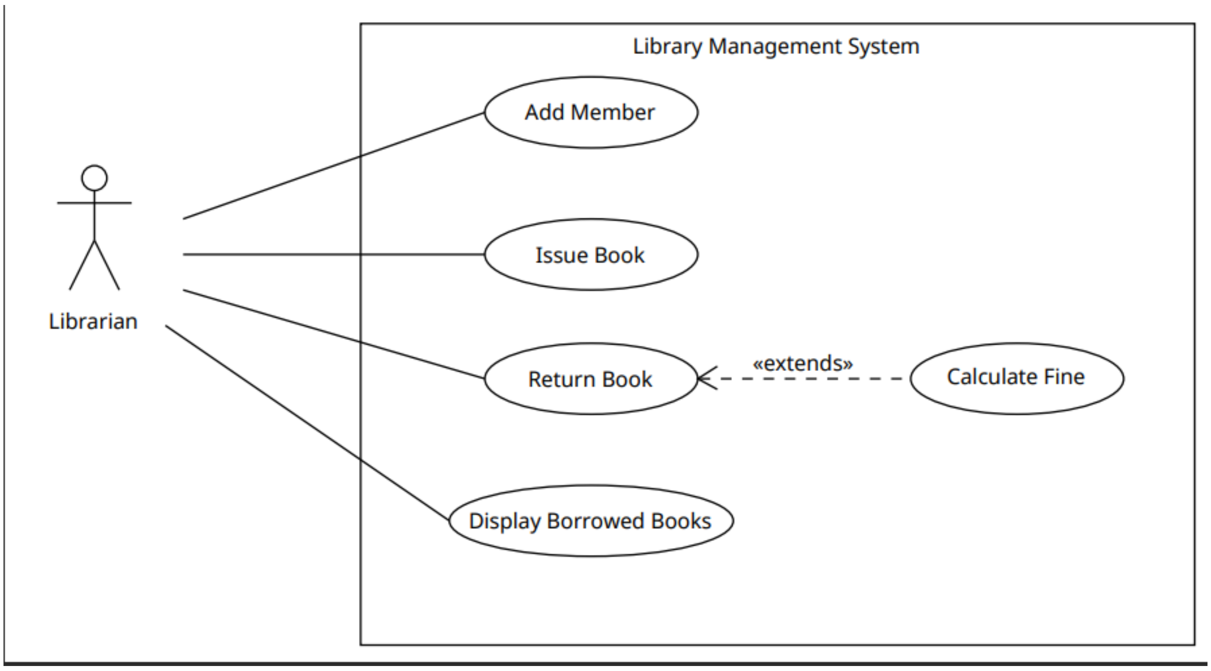
Table of Contents

1. System Design Approach

1.1 Library Management System Class Diagram

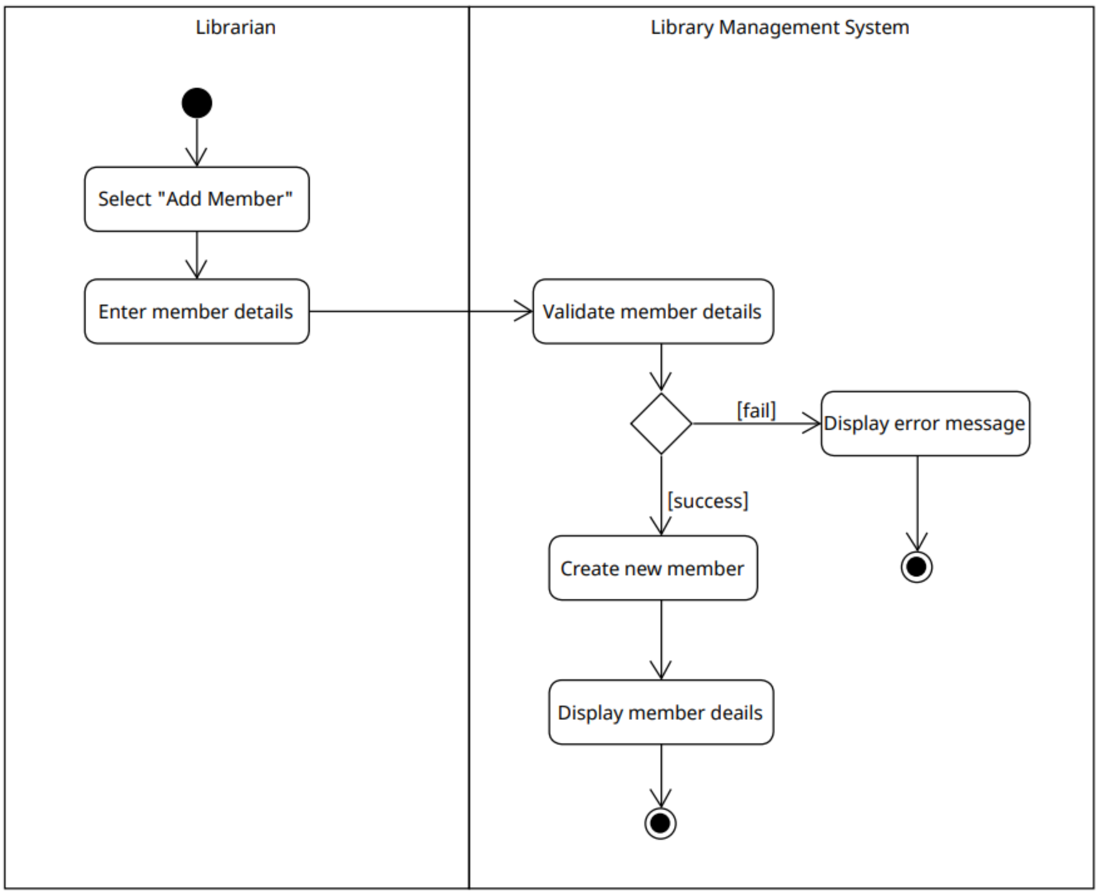

Person
- name: string
- address: string
- email: string
+ getName() : string
+ setName(name : string) : void
+ getAddress() : string
+ setAddress(address : string) : void
+ getEmail() : string
+ setEmail(email : string) : void
Librarian
- staffId: int
- salary: int
+ Librarian(staffID : int, name : string, address : string email : string, salary : int)
+ addMember() : void
+ issueBook(memberID : int, bookID : int) : void
+ returnBook(memberID : int, bookID : int) : void
+ displayBorrowedBooks(memberID : int) : void
+ calcFine(memberID : int) : void
+ getStaffID() : int
+ setStaffID(staffID : int)
+ getSalary() : int
+ setSalary(salary : int) : void
Member
- memberId: int
- booksLoaned: Book <vector>
+ Member(memberID : int, name : string, address : string email : string)
+ getMemberID() : string
+ getBooksBorrowed() : Book <vector>
+ setBooksBorrowed(book : Book) : void
0..1
issues ▶
0..*
0..1
borrows ▶
0..*
Book
- bookID : int
- bookName : string
- authorFirstName : string
- authorLastName : string
- bookType : string
- dueDate : Date
- borrower : Member
+ Book(bookID : int, bookName : string, authorFirstName : string, authorLastName : string)
+ getbookID() : string
+ getbookName() : string
+ getAuthorFirstName() : string
+ getAuthorLastName() : string
+ getDueDate() : Date
+ setDueDate(dueDate : Date) : void
+ returnBook() : void
+ borrowBook(borrower : Member, dueDate : Date) : void

1.2 Library Management System Use case Diagram



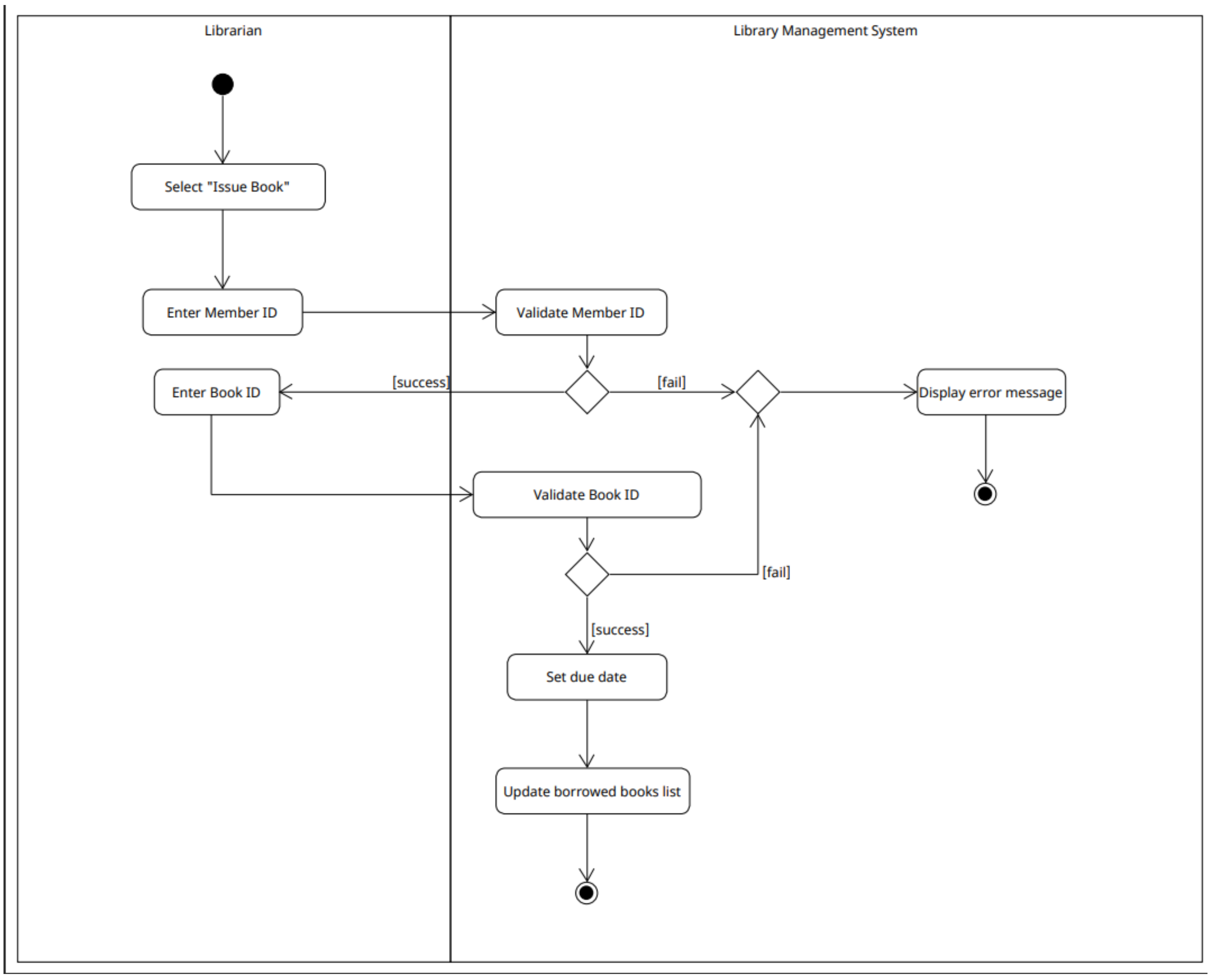
The system Use case Diagram Features the main functionalities of the system which are the Add member, Issue book, Return Book and Display borrowed book with the calculate fine use case extending from the return book use case because the calculate fine method is only called when a book being returned is overdue. The Use case diagram has a librarian as the only actor because the system is to be used by the librarian and no one else.

1.3 Add member Activity Diagram



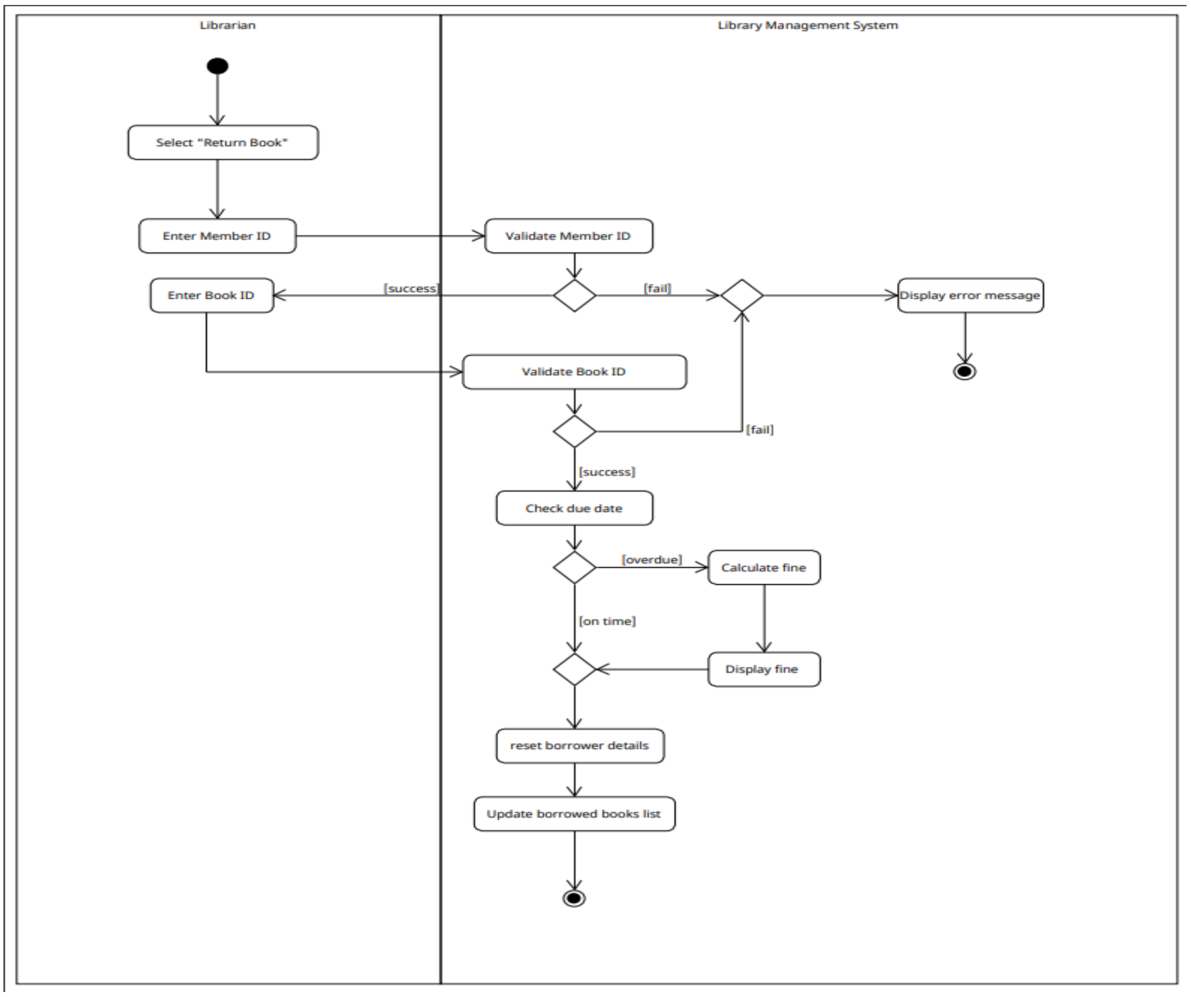
This activity diagram shows the process flow for the Add member use case. When the librarian selects add Member from the system menu, a prompt is initiated for the librarian to enter the new member details which are the member ID, name, address, and email address. An error message will be displayed if any of the fields entered are invalid. If all entries are valid, the member details are displayed on the console.

1.4 Issue Book Activity Diagram



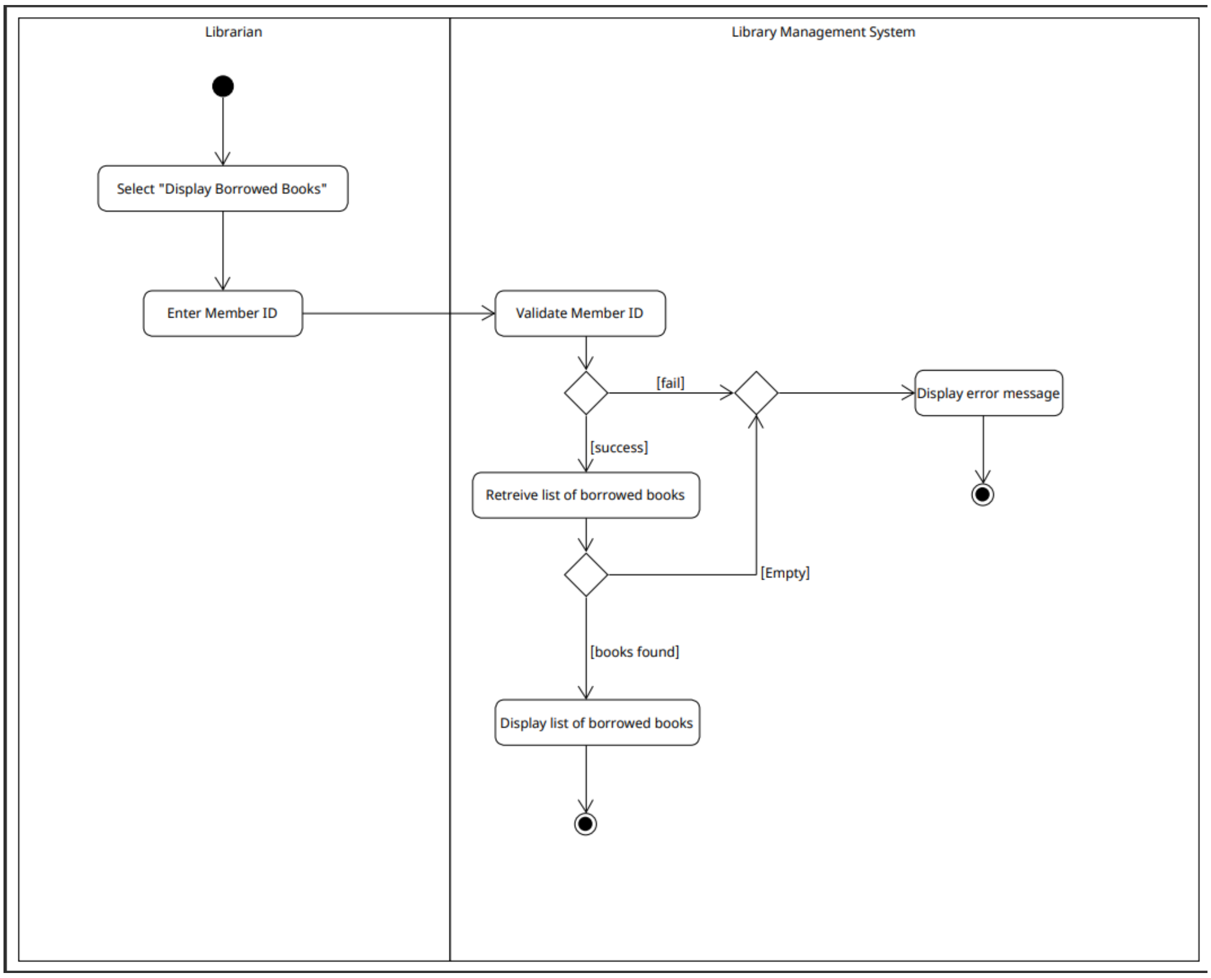
This activity diagram shows the process flow for the Issue book use case. When the librarian selects issue book from the system menu, a prompt is initiated for the librarian to enter an existing member ID. If the member doesn't exist in the system, an error message is displayed. If the member ID exist, a prompt is initiated for the librarian to enter the book ID of the book to be issued. Id the book doesn't exist in the collection of books, an error message is displayed. If the book exists, the due date is set, and the book is added to a collection of loaned books.

1.5 Return Book Activity Diagram



This activity diagram shows the process flow for the return book use case. When the librarian selects return book from the system menu, a prompt is initiated for the librarian to enter an existing member ID. If the member doesn't exist in the system, an error message is displayed. If the member ID exist, a prompt is initiated for the librarian to enter the book ID of the book to be issued. Id the book doesn't exist in the collection of books loaned by the member, an error message is displayed. If the book exists, the due date is checked to determine if the book is overdue. If the book is overdue, the calculate fine method is called and the fine calculated is displayed on the console. If the book is not overdue, the method call is skipped. The borrower details of the book are reset, and the book is then deleted from the book loaned list

1.6 Display Borrowed Books Activity Diagram


Librarian
Library Management System
Select "Display Borrowed Books"
Enter Member ID
Validate Member ID
[fail]
Display error message
[success]
Retreive list of borrowed books
[Empty]
[books found]
Display list of borrowed books

This activity diagram shows the process flow for the Display borrowed book use case. When the librarian selects Display borrowed book from the system menu, a prompt is initiated for the librarian to enter an existing member ID. If the member doesn't exist in the system, an error message is displayed. If the member ID exist, the collection of books borrowed by the member is then retrieved. If no books have been borrowed by the member, an error message is displayed. If books have been borrowed by the member, the content of the collection is then displayed on the console.

2. System Implementation

The UML class diagram was used to generate a class declaration for the system in a header file- library.h. The system classes declared in the header file were implemented in a separate file- library.cpp. An abstract data type was created for the date data type used in the system which also has a header file- date.h for member variable and function declarations and a separate file- date.cpp for the Implementation. In the library.cpp system implementation file, a members vector was declared to store the member details being added to the system.

2.1 Add member Functionality

The add member functionality was implemented by validating the member details entered by the librarian to check if the member ID entered consists of only digits, if the name and address entered consists of only alphabets and if the email address enters was entered in a valid email format with an @ symbol. If not valid, the system continuously loops until a valid entry is made. Once verified, an instance of the member class was created and added to the member vector, the member details entered are then output to the console with a success message and the method is terminated.

2.2 Issue Book Functionality

The issue book functionality was implemented by validating the member id by looping through the members vector to find a member with the member ID entered. If not found, an error message is displayed, and the method is terminated. If found, a prompt is initiated to enter the filename to retrieve the books. Once the book ID, name and author mane is retrieved from the file, an instance of the Book class is created. a prompt is then initiated to enter the date of issue. An instance of the Date, issue date is then created. Using methods from the Date abstract data type like get day, get month, and get year, 3 days are added to the issue date to get the due date and an instance of Date, due date is created. the set due date method from the book class is then called by the boo instance to set the due date by passing the due date instance. The borrow book method of the book class is also called by the book instance and the borrower which is the member whose member id was retrieved from the members vector and the due date instance is then passed into the method. The set borrowed books of the member class which adds books borrowed by a member to the book loaned vector is called by the borrower and passed the book instance into the method. A success message is then displayed and the methos is terminated.

2.3 Return Book Functionality

The return book functionality was implemented by validating the member id by looping through the members vector to find a member with the member ID entered. If not found, an error message is displayed, and the method is terminated. If found, the book loaned vector is then looped through to find the book which matches the book ID entered. If the book is not found, an error message is displayed, and the method is terminated. The book is found, the due date of the books borrowed by the member are checked by comparing the due date which is gotten by the book calling the get due date method and the current date which is gotten from the currentDate method of the Date data type using the compare method of the date data type to find out if the member has any overdue books. The calculate fine method is called to calculate the total fine of the overdue books. If there are no overdue books, the function call is skipped. The book to be returned is then removed from the member’s collection of loaned books and a success message is displayed and the method is terminated.

2.4 Calculate Fine Functionality

The calculate fine method was implemented by looping through the members vector to find the member whose ID was passed into the return books method. Once the member is found, an instance of the current date is then created using the current date method of the Date data type. The member's collection of loaned books is then iterated through, and the due date of each book is checked and compared to the current date to determine if the book is overdue. If the book is overdue, 1£ is added to the fine previously initialized to 0 for each day overdue. The compare method returns the difference of the number of days between the current date and the due date. The total fine of the member for all borrowed books is then displayed on the console when called by the return book method.

2.5 Display Borrowed Book Functionality

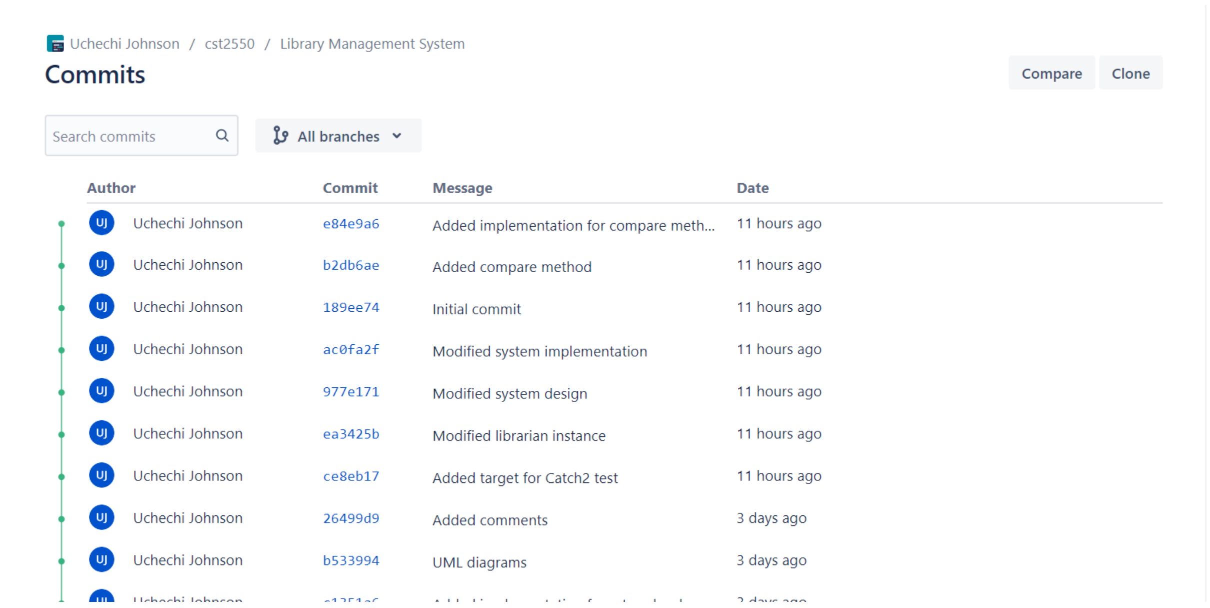
The display borrowed books method was implemented by validating the member ID by looping through the members vector to find a member with the member ID entered. If not found, an error message is displayed, and the method is terminated. If found, the book loaned vector of the member is iterated through and the content is displayed on the console and the method is terminated. If the vector is empty, an error message is displayed, and the method is terminated.

2.6 Makefile

A make file was used to simplify the execution of the system. A rule to generate a library object file- library.o is when library.cpp and library.h file which are the main files of the system is written. Another rule to generate a date object file date.o file when date.cpp and date.h which are the files for the Date abstract data type is written. A rule to generate an executable file for the system when the main.cpp, library.o and date.o files are compiled is written. These rules make the compilation of these numerous files less complicates and reduce compilation errors.

An 'all' target is made to generate the object files and the executable file from the rules. A 'test target is also made to generate a test executable file from a rule written to generate the executable file from the compilation of library_tests.cpp, library.o and date.o. A 'clean' target is also made to execute a clean rule which removes all object files and executable files.

2.7 Version Control



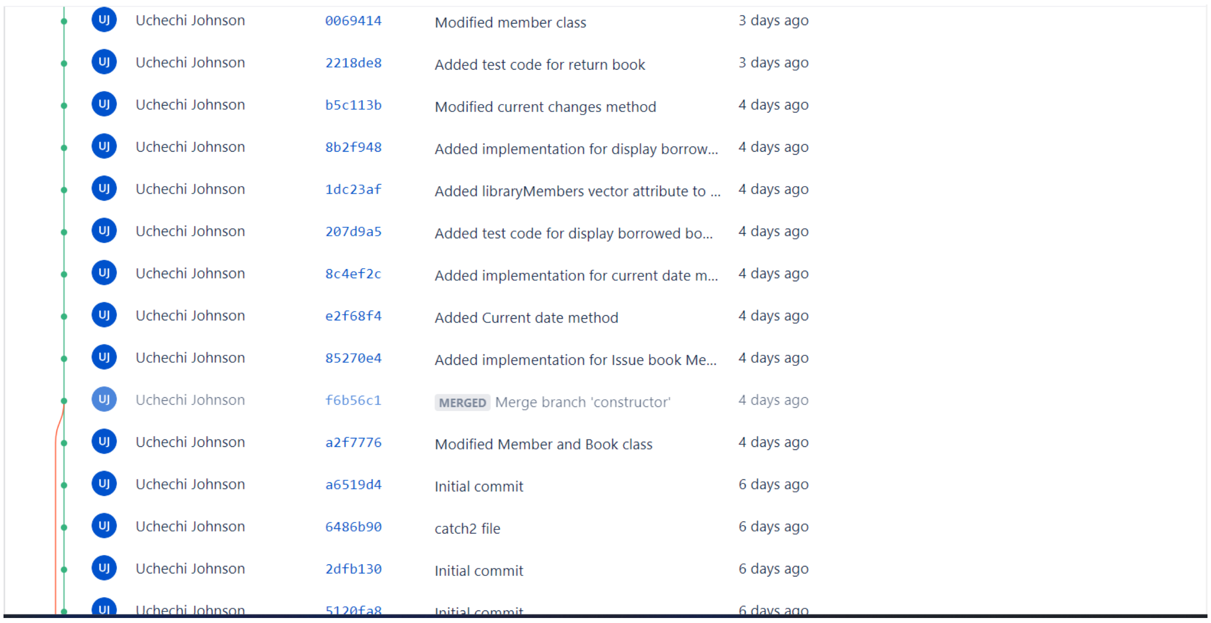
Uchechi Johnson / cst2550 / Library Management System

Commits

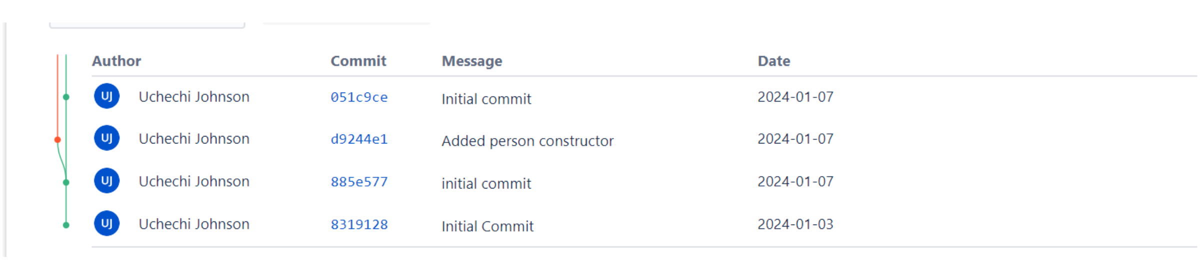
Compare Clone

Search commits All branches

Author	Commit	Message	Date
Uchechi Johnson	e84e9a6	Added implementation for compare meth...	11 hours ago
Uchechi Johnson	b2db6ae	Added compare method	11 hours ago
Uchechi Johnson	189ee74	Initial commit	11 hours ago
Uchechi Johnson	ac0fa2f	Modified system implementation	11 hours ago
Uchechi Johnson	977e171	Modified system design	11 hours ago
Uchechi Johnson	ea3425b	Modified librarian instance	11 hours ago
Uchechi Johnson	ce8eb17	Added target for Catch2 test	11 hours ago
Uchechi Johnson	26499d9	Added comments	3 days ago
Uchechi Johnson	b533994	UML diagrams	3 days ago
Uchechi Johnson	[illegible]	[illegible]	3 days ago



Author	Commit	Message	Date
Uchechi Johnson	0069414	Modified member class	3 days ago
Uchechi Johnson	2218de8	Added test code for return book	3 days ago
Uchechi Johnson	b5c113b	Modified current changes method	4 days ago
Uchechi Johnson	8b2f948	Added implementation for display borrow...	4 days ago
Uchechi Johnson	1dc23af	Added libraryMembers vector attribute to ...	4 days ago
Uchechi Johnson	207d9a5	Added test code for display borrowed bo...	4 days ago
Uchechi Johnson	8c4ef2c	Added implementation for current date m...	4 days ago
Uchechi Johnson	e2f68f4	Added Current date method	4 days ago
Uchechi Johnson	85270e4	Added implementation for Issue book Me...	4 days ago
Uchechi Johnson	f6b56c1	MERGED Merge branch 'constructor'	4 days ago
Uchechi Johnson	a2f7776	Modified Member and Book class	4 days ago
Uchechi Johnson	a6519d4	Initial commit	6 days ago
Uchechi Johnson	6486b90	catch2 file	6 days ago
Uchechi Johnson	2dfb130	Initial commit	6 days ago
Uchechi Johnson	5120fa8	Initial commit	6 days ago



Author	Commit	Message	Date
Uchechi Johnson	051c9ce	Initial commit	2024-01-07
Uchechi Johnson	d9244e1	Added person constructor	2024-01-07
Uchechi Johnson	885e577	initial commit	2024-01-07
Uchechi Johnson	8319128	Initial Commit	2024-01-03

3. Limitations

A limitation that was encountered in developing the system was implementing the calculate fine method. The calculate fine method takes only the member ID as a parameter and not a book ID. For every book a member tries to return, the total fine of all the books borrowed is calculated even when the book being returned is not overdue.

4. Lessons

Creating a global variable to store the members being added to the system worked with system seamlessly and was helpful in the implementation of other methods of the system.

Retrieving the books borrowed by a member from the vector of loaned books did not work out and was returning the reference to the elements of the vector. A dereferencing operator had to be used with the get borrowed books method to get the actual details of books borrowed.

5. Future Works

In a future project, I will avoid the use of void methods with no parameters because they were difficult to test. I will also pass a book id as a parameter of the calculate fine function together with the member ID so when the method is called, only the fine for the book to be returned is displayed.