COURSE WORK 1
LIBRARY MANAGEMENT SYSTEM

CST2550

M00872834

3 Jan 2024

OVERVIEW

INTRODUCTION

- **Project Overview:**
- Purpose: Developing a user-friendly Library Management System for a small library
- Focus: Easy management of book collections and member records

- **Key features:**
- Book management across different genres
- Member management with a user-friendly interface
- Features include adding members, issuing/returning books, and fine calculation

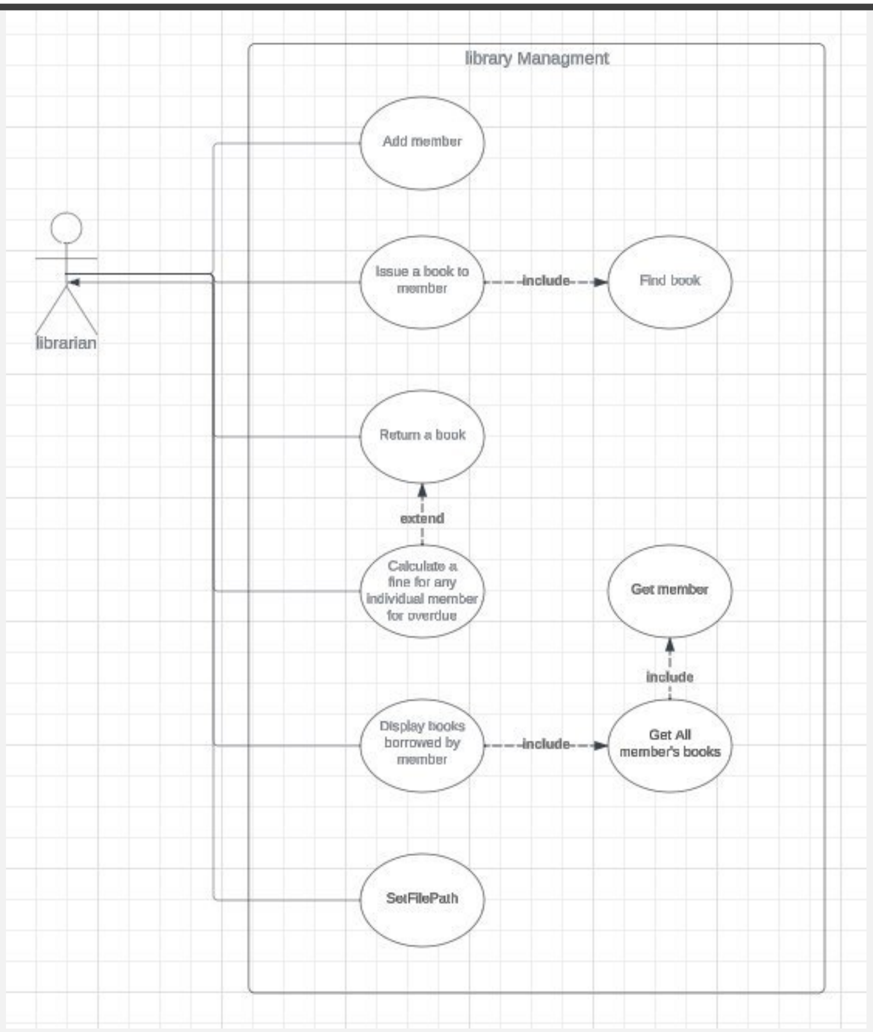
INTRODUCTION

- **Technical Highlights:**
- Developed in C++ without third-party libraries
- Efficient tracking of books by ID, name, author, type, and page count
- The use of Git for version control

- **Goal:**
- Optimizing library operations and improving the efficiency of book management

UML

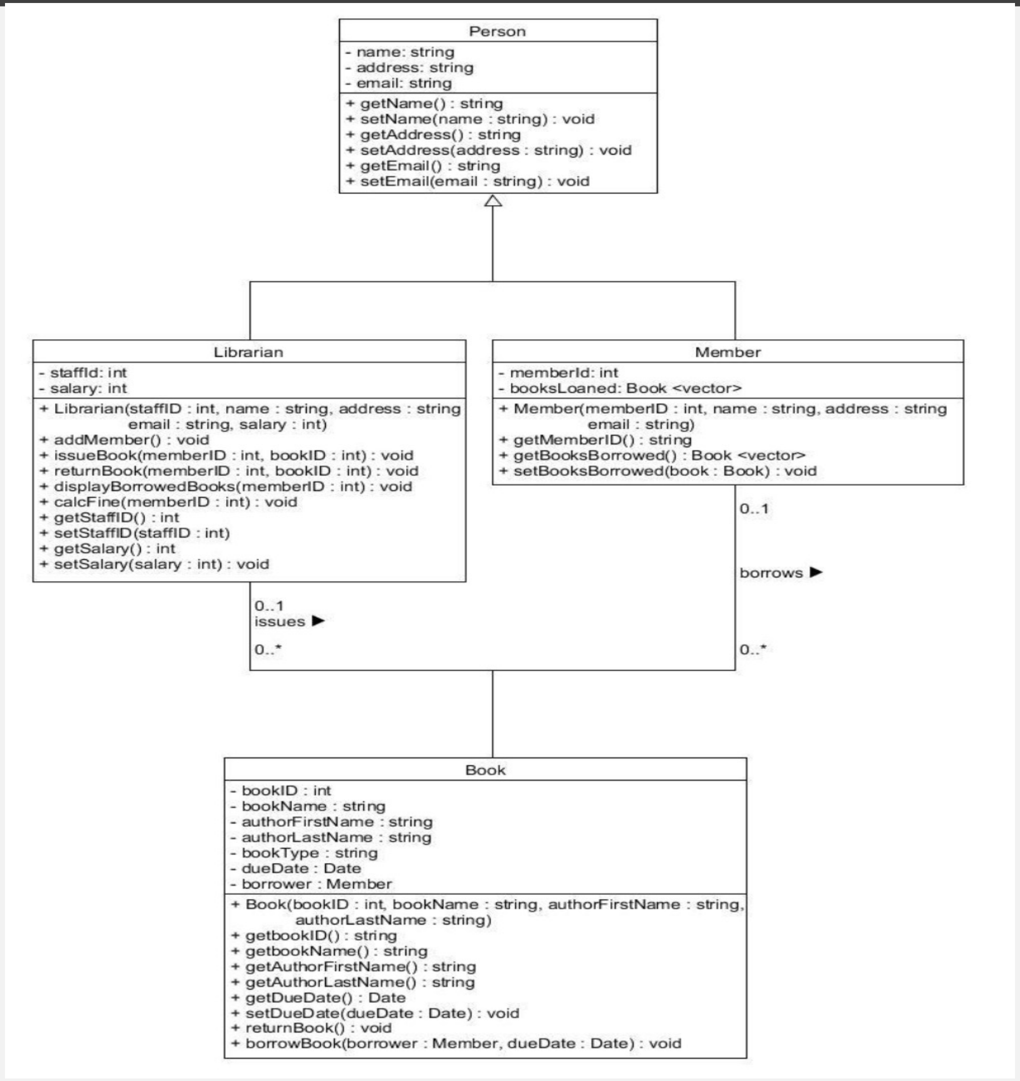
USE CASE DIAGRAM



M00872834

UML

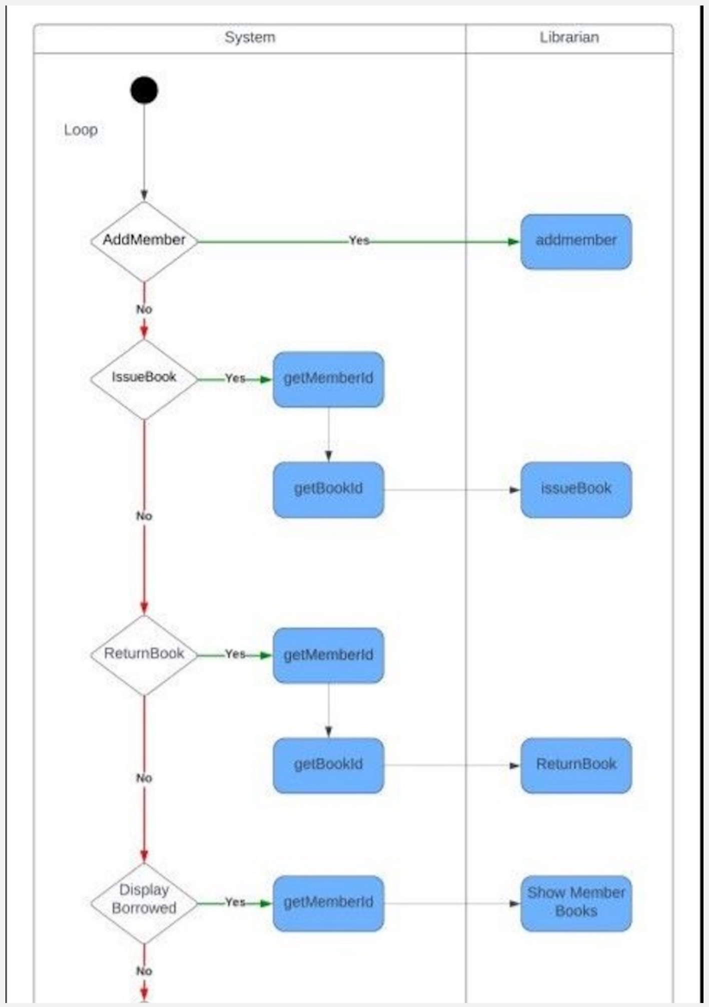
CLASS DIAGRAM

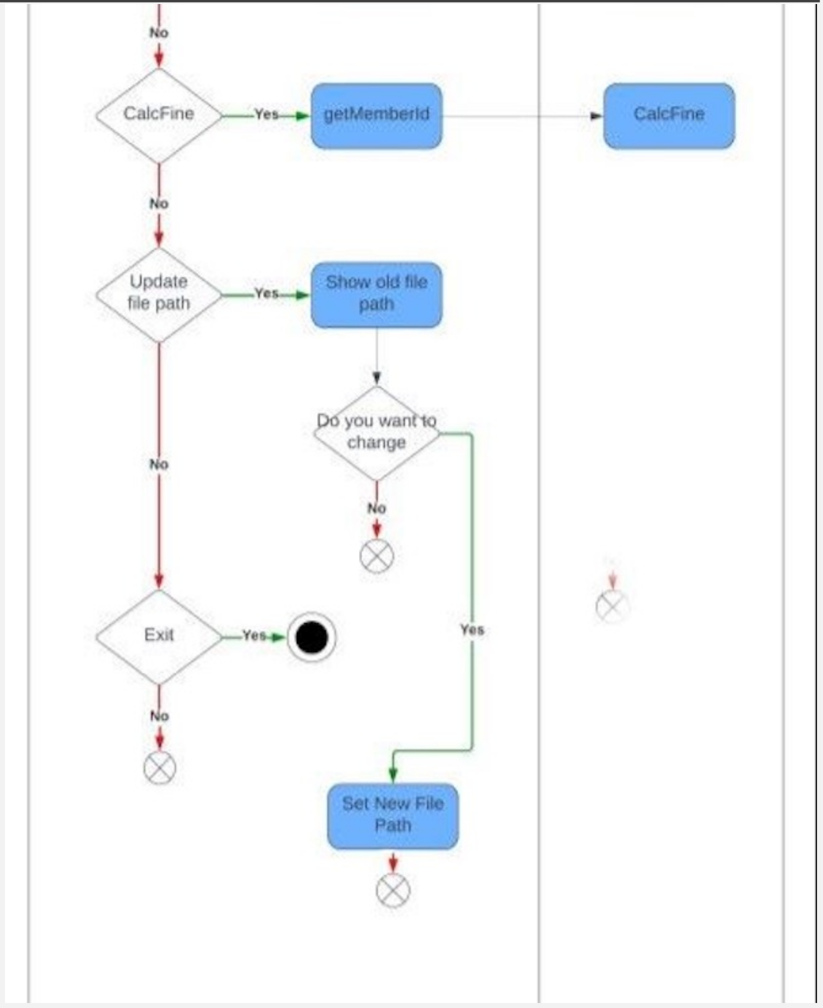

Person
- name: string
- address: string
- email: string
+ getName() : string
+ setName(name : string) : void
+ getAddress() : string
+ setAddress(address : string) : void
+ getEmail() : string
+ setEmail(email : string) : void
Librarian
- staffId: int
- salary: int
+ Librarian(staffID : int, name : string, 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

M00872834

UML

ACTIVITY DIAGRAM (MAIN)

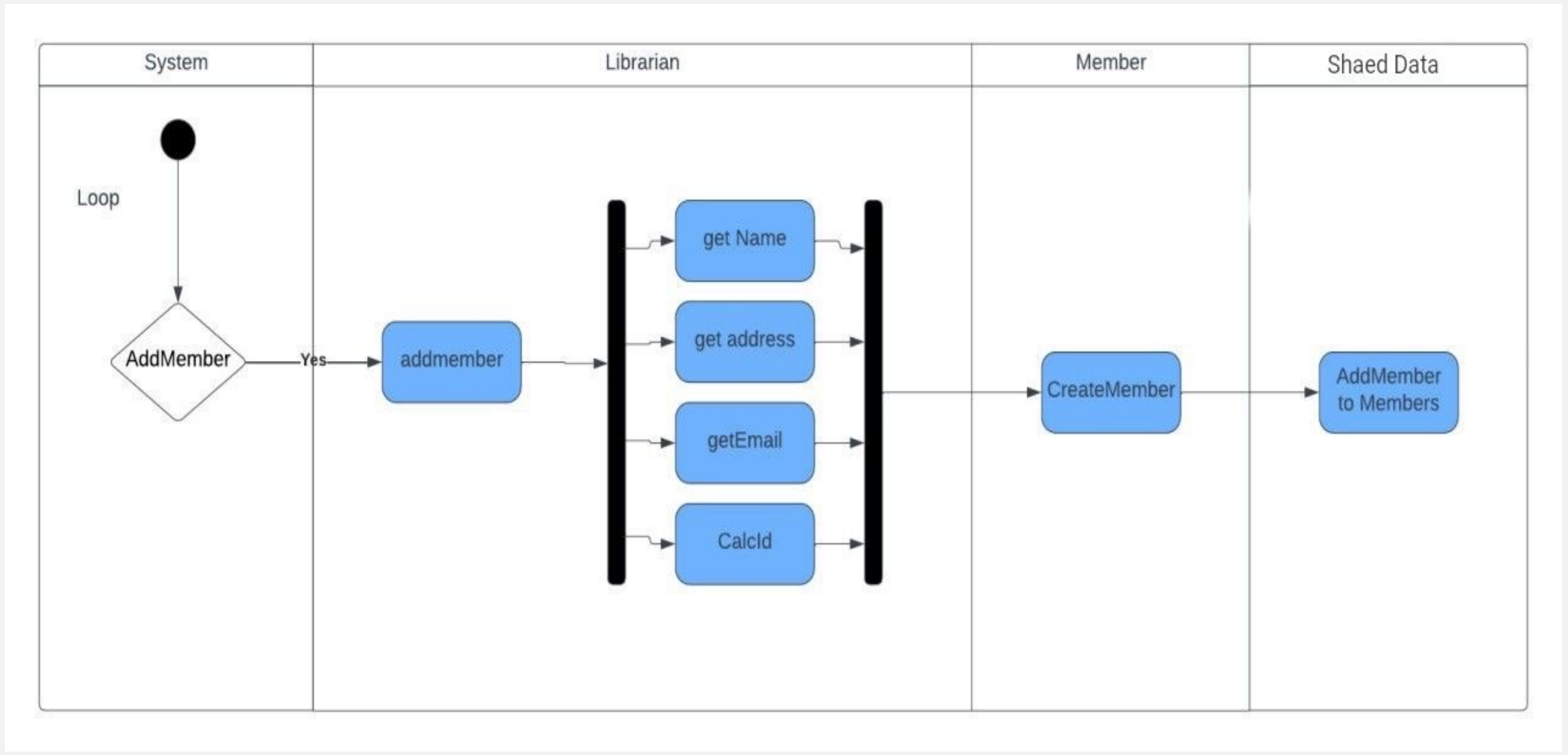




M00872834

UML

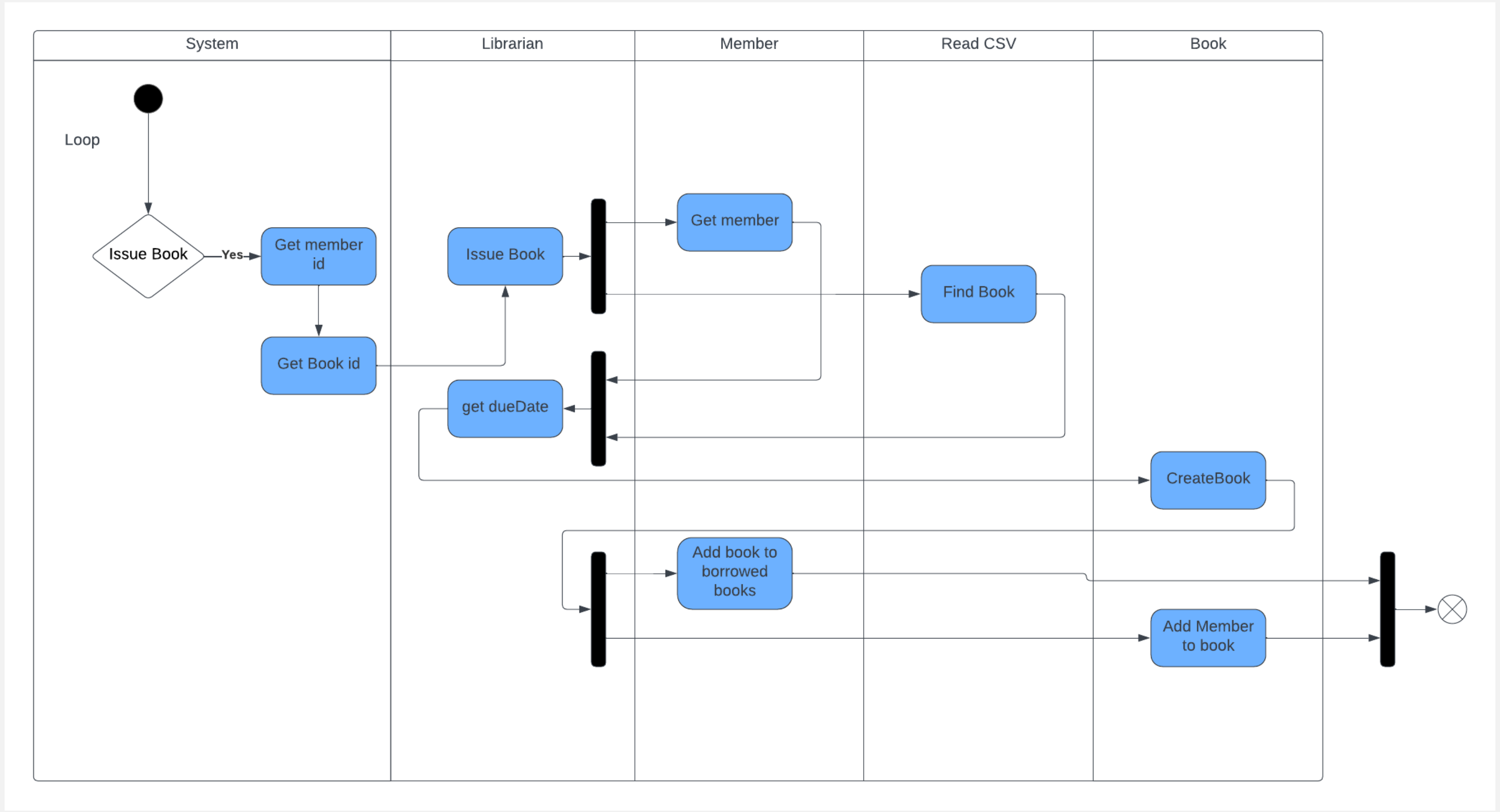
ACTIVITY DIAGRAM (ADD MEMBER)



M00872834

UML

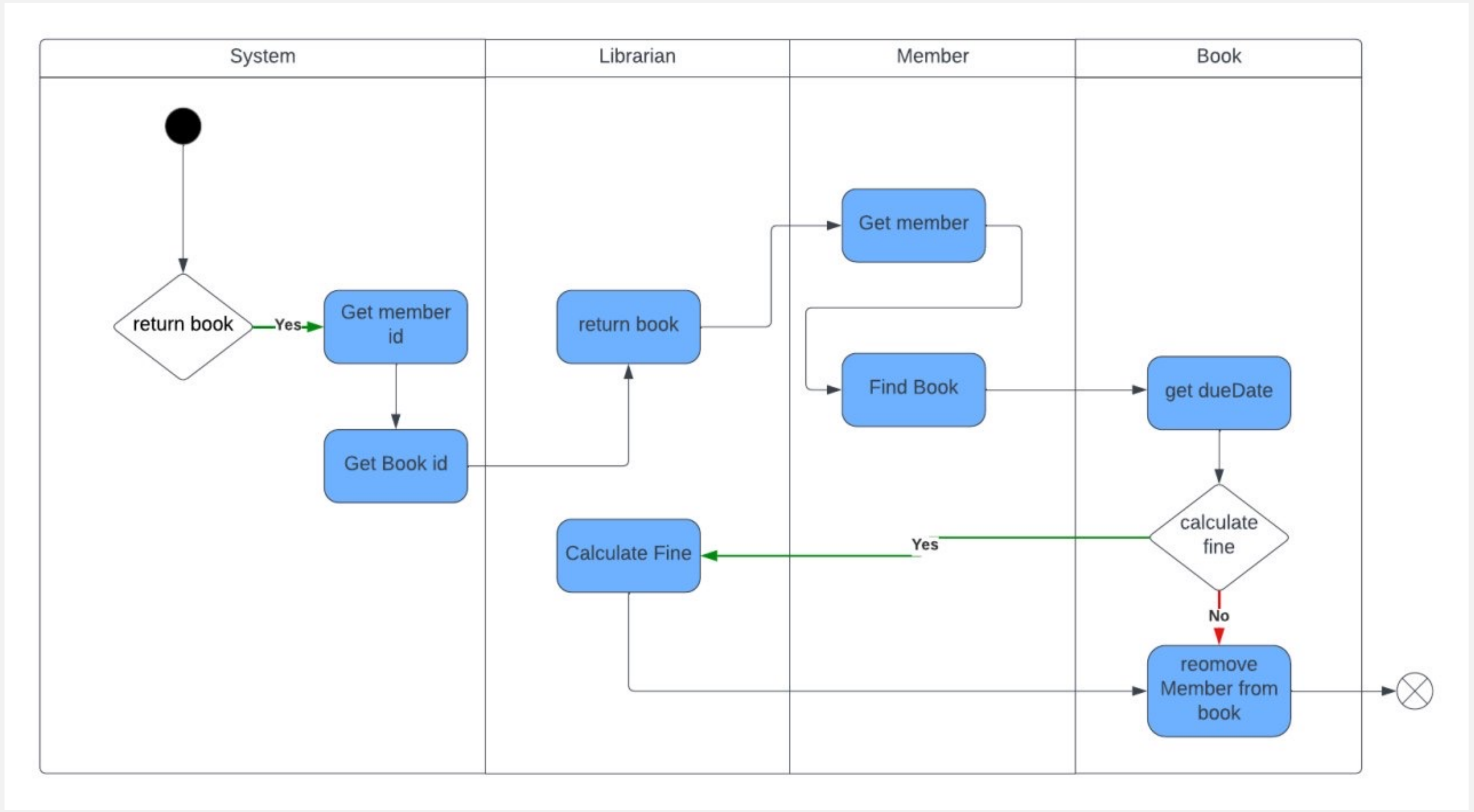
ACTIVITY DIAGRAM (ISSUE A BOOK)



M00872834

UML

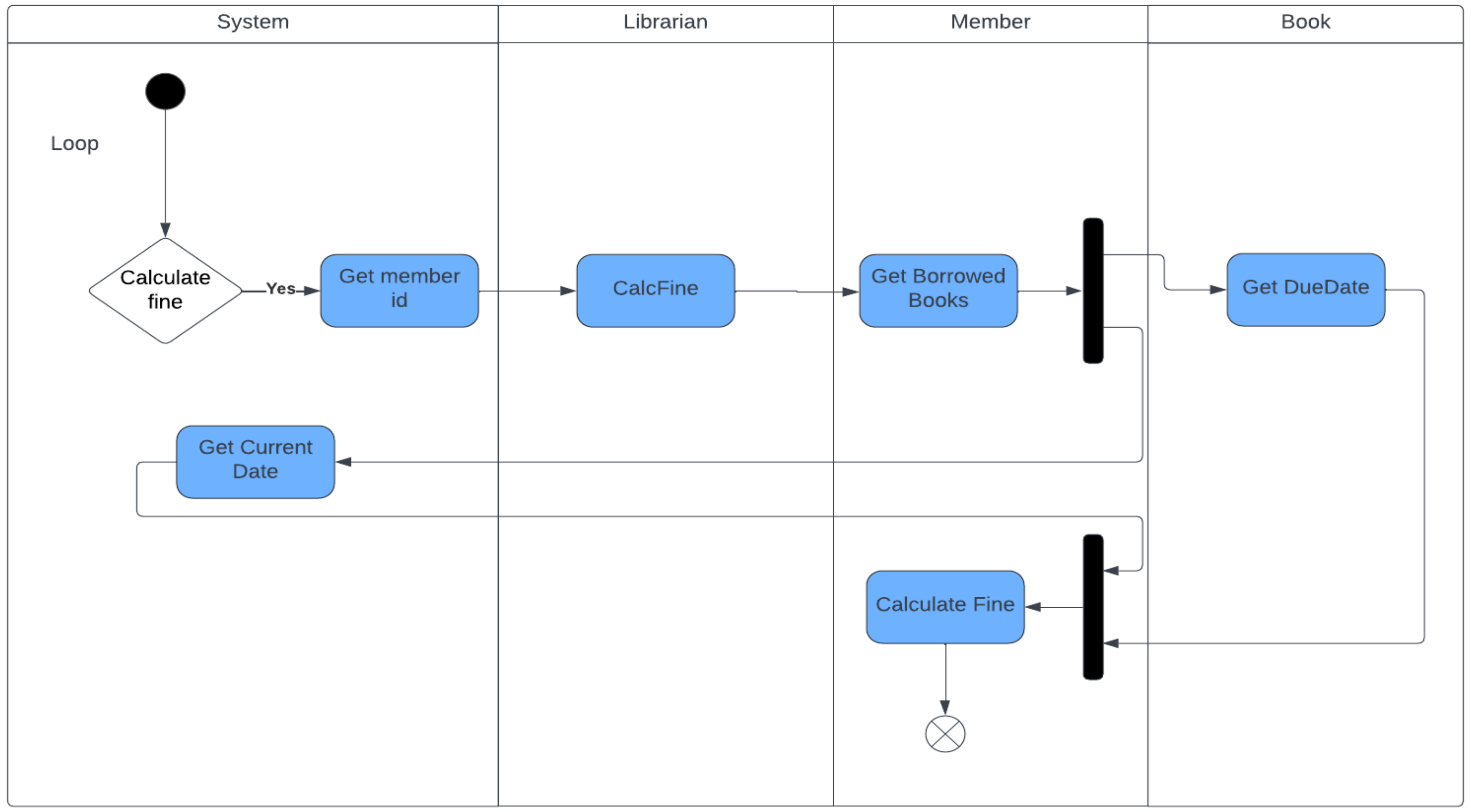
ACTIVITY DIAGRAM (RETURN A BOOK)



M00872834

UML

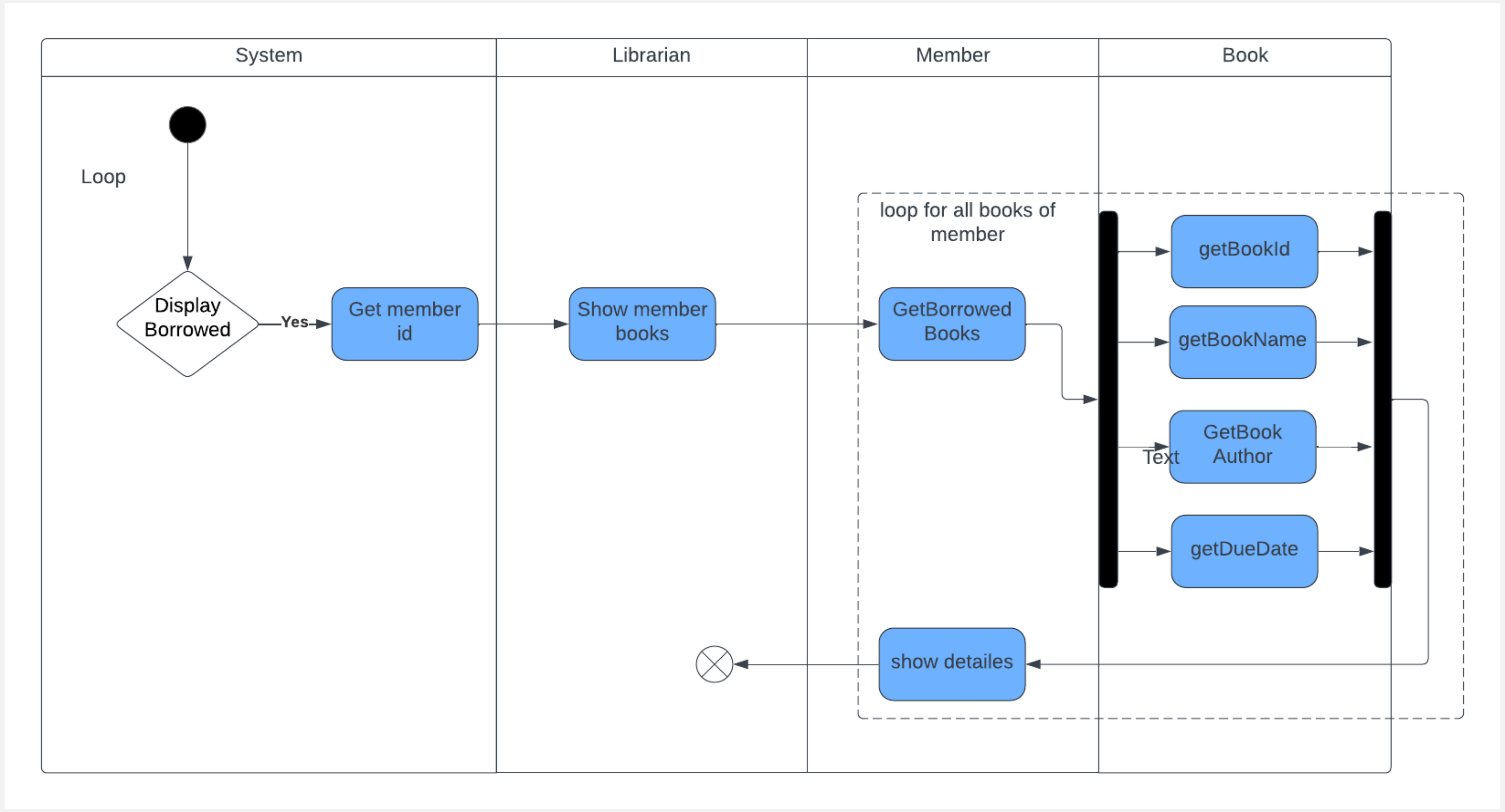
ACTIVITY DIAGRAM (CALCULATE FINE)



M00872834

UML

ACTIVITY DIAGRAM (DISPLAY BORROWED BOOKS)

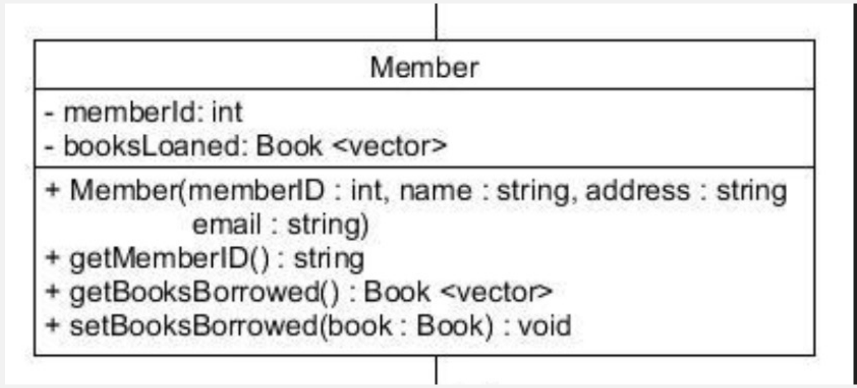


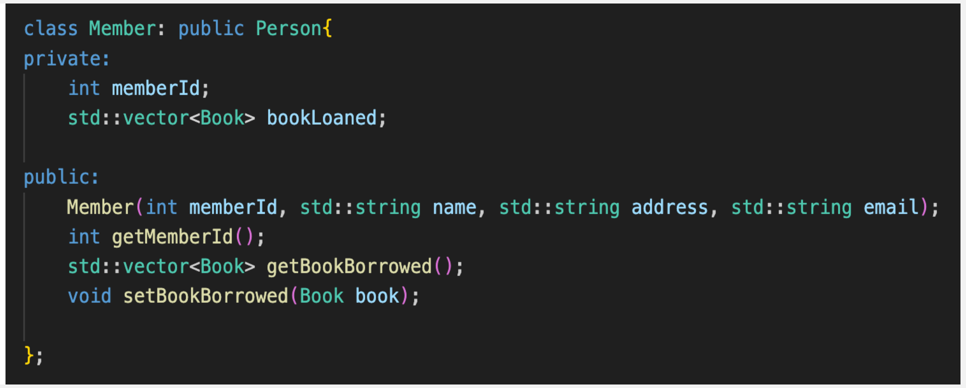
M00872834

IMPLEMENTATION

TRANSLATE THE DESIGN INTO A SOFTWARE

- **UML to Classes:**
- UML diagrams provided a blueprint for classes structures
- Developing classes like "Member" according to Class Diagram
- Developing member variables and functions from Class Diagram


Member
- memberId: int
- booksLoaned: Book <vector>
+ Member(memberID : int, name : string, address : string email : string)
+ getMemberID() : string
+ getBooksBorrowed() : Book <vector>
+ setBooksBorrowed(book : Book) : void



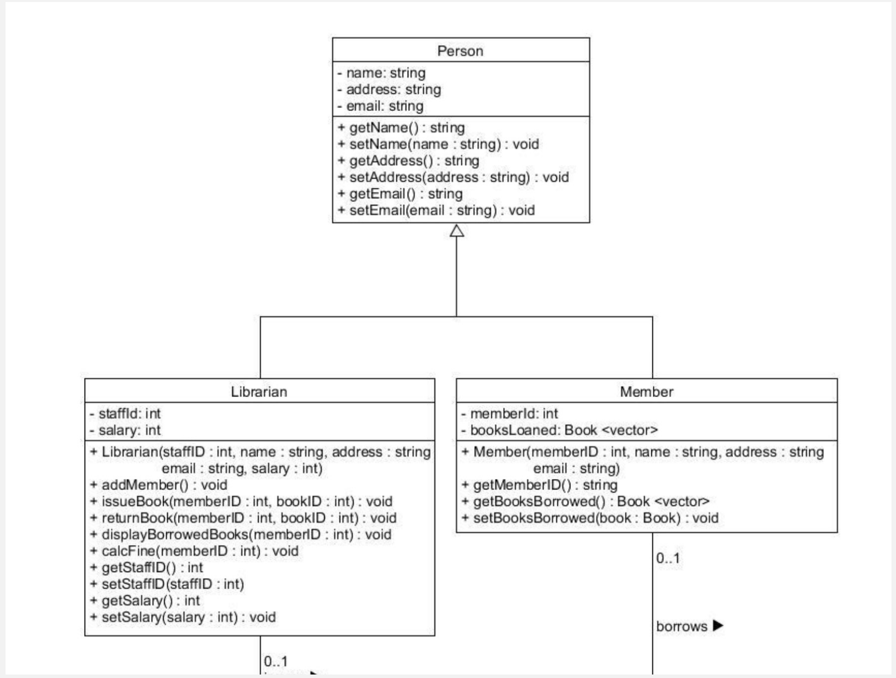
```
class Member: public Person{
private:
    int memberId;
    std::vector<Book> bookLoaned;

public:
    Member(int memberId, std::string name, std::string address, std::string email);
    int getMemberId();
    std::vector<Book> getBookBorrowed();
    void setBookBorrowed(Book book);

};
```

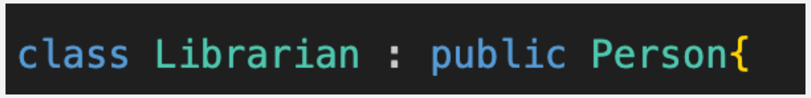
M00872834

IMPLEMENTATION

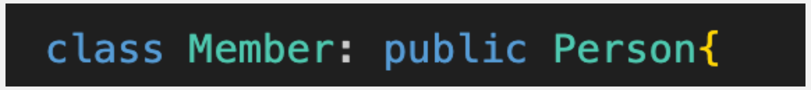
TRANSLATE THE DESIGN INTO A SOFTWARE

- **Define Relationships:**
- Establishing relationships (inheritance, association) between classes from 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
borrows ▶
0..1



```
class Librarian : public Person{
```



```
class Member: public Person{
```

M00872834

IMPLEMENTATION

WHAT THE MAKEFILE WAS USED FOR

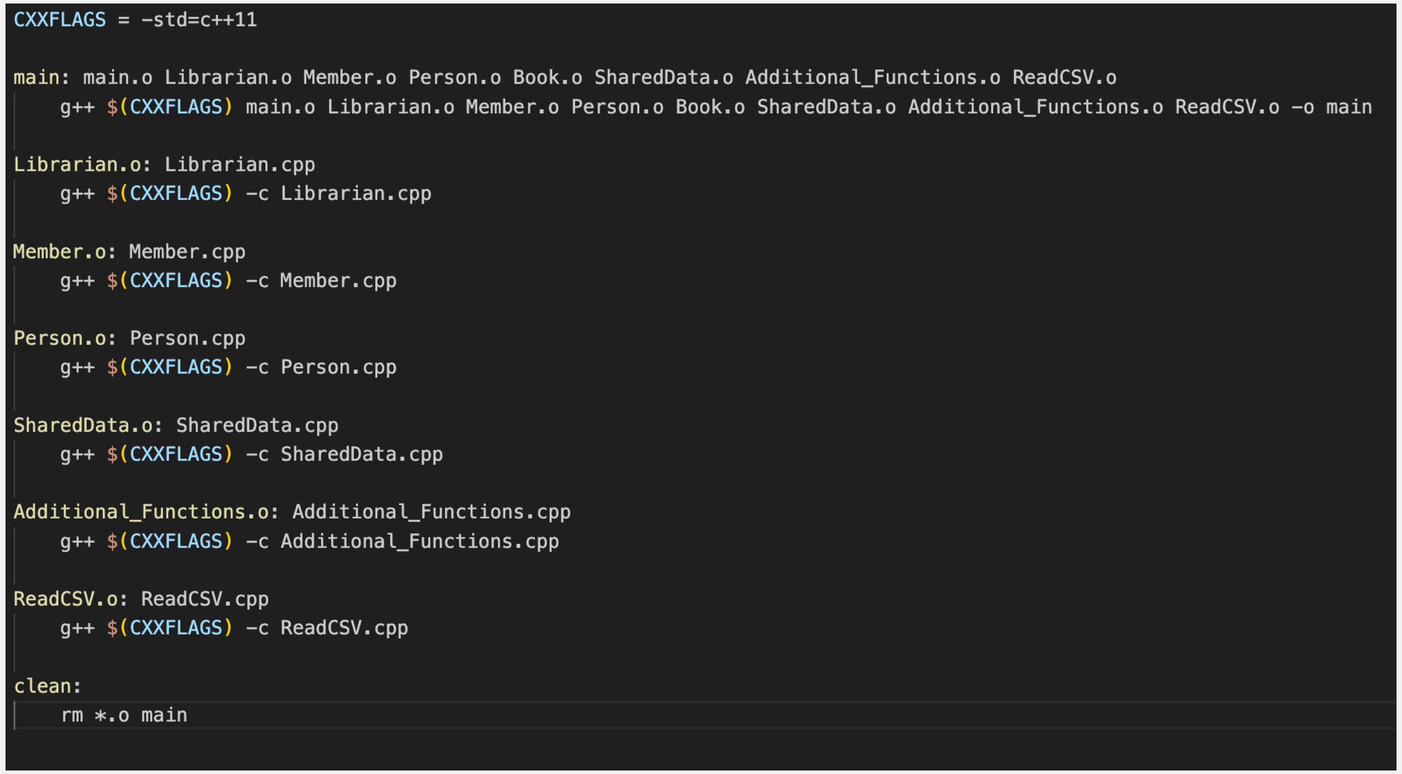
- **Automated Compilation:**
- Simplified the process of compiling multiple source files into an executable program

- **Efficient Build Rules:**
- Only recompile files that have been changed since the last build, saving time.

- **Consistency:**
- Ensures that every team member or user compiles the project in the same way.

IMPLEMENTATION

WHAT THE MAKEFILE WAS USED FOR



```
CXXFLAGS = -std=c++11

main: main.o Librarian.o Member.o Person.o Book.o SharedData.o Additional_Functions.o ReadCSV.o
	g++ $(CXXFLAGS) main.o Librarian.o Member.o Person.o Book.o SharedData.o Additional_Functions.o ReadCSV.o -o main

Librarian.o: Librarian.cpp
	g++ $(CXXFLAGS) -c Librarian.cpp

Member.o: Member.cpp
	g++ $(CXXFLAGS) -c Member.cpp

Person.o: Person.cpp
	g++ $(CXXFLAGS) -c Person.cpp

SharedData.o: SharedData.cpp
	g++ $(CXXFLAGS) -c SharedData.cpp

Additional_Functions.o: Additional_Functions.cpp
	g++ $(CXXFLAGS) -c Additional_Functions.cpp

ReadCSV.o: ReadCSV.cpp
	g++ $(CXXFLAGS) -c ReadCSV.cpp

clean:
	rm *.o main
```

Makefile of Library Management System

IMPLEMENTATION

HOW AND WHY VERSION CONTROL USED FOR

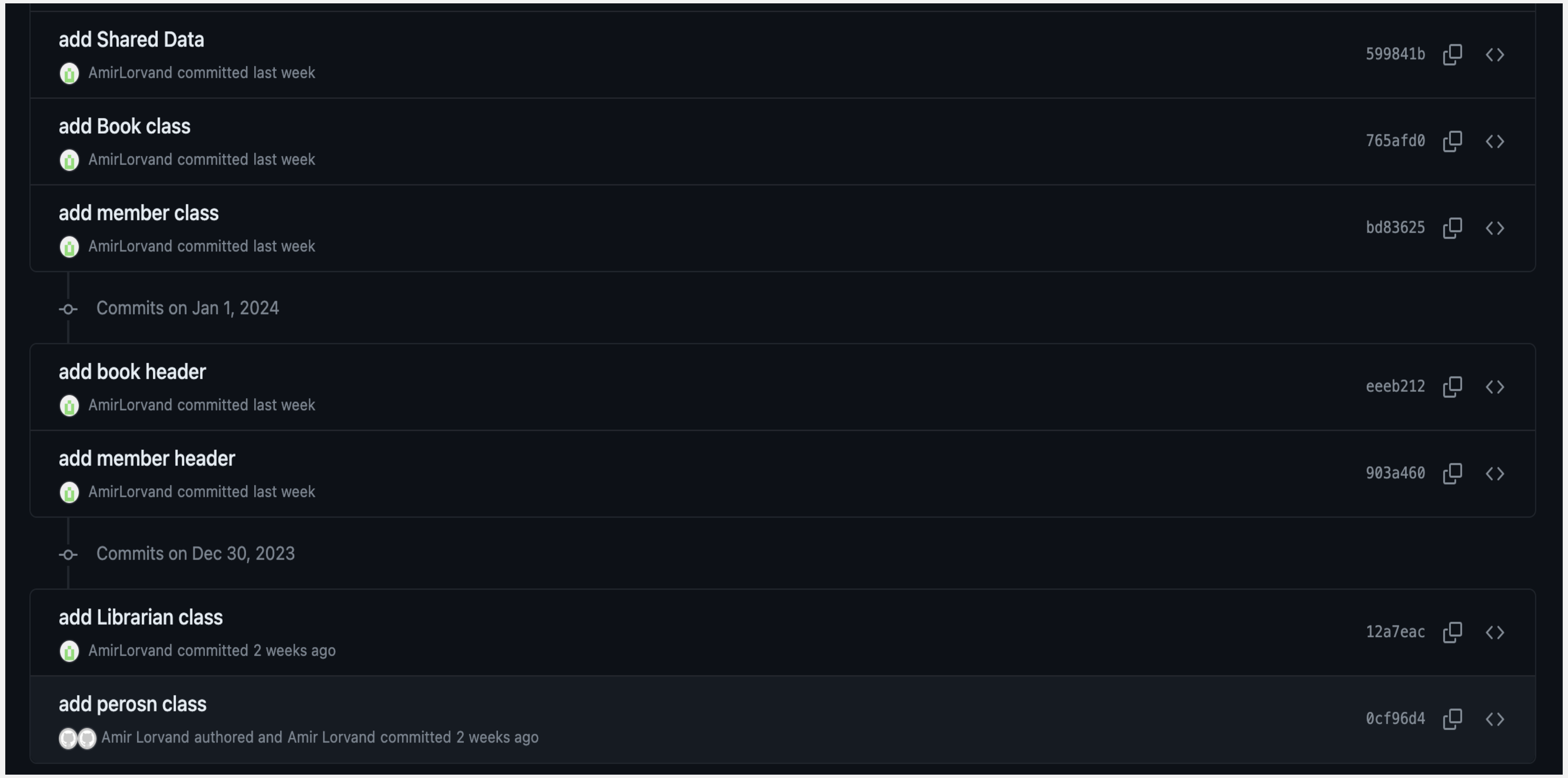
- **Tracking Changes:**
- Version control allows for keeping a detailed history of code changes, making it easier to understand how the project evolved.

- **Collaboration:**
- Enables multiple people to work on the same project.
- (not useful in this project!!!)

- **Backup and Recovery:**
- Previous versions of the code can be retrieved easily.

IMPLEMENTATION

SCREENSHOT OF GITHUB COMMITS



add Shared Data
AmirLorvand committed last week
599841b

add Book class
AmirLorvand committed last week
765afd0

add member class
AmirLorvand committed last week
bd83625

Commits on Jan 1, 2024

add book header
AmirLorvand committed last week
eeeb212

add member header
AmirLorvand committed last week
903a460

Commits on Dec 30, 2023

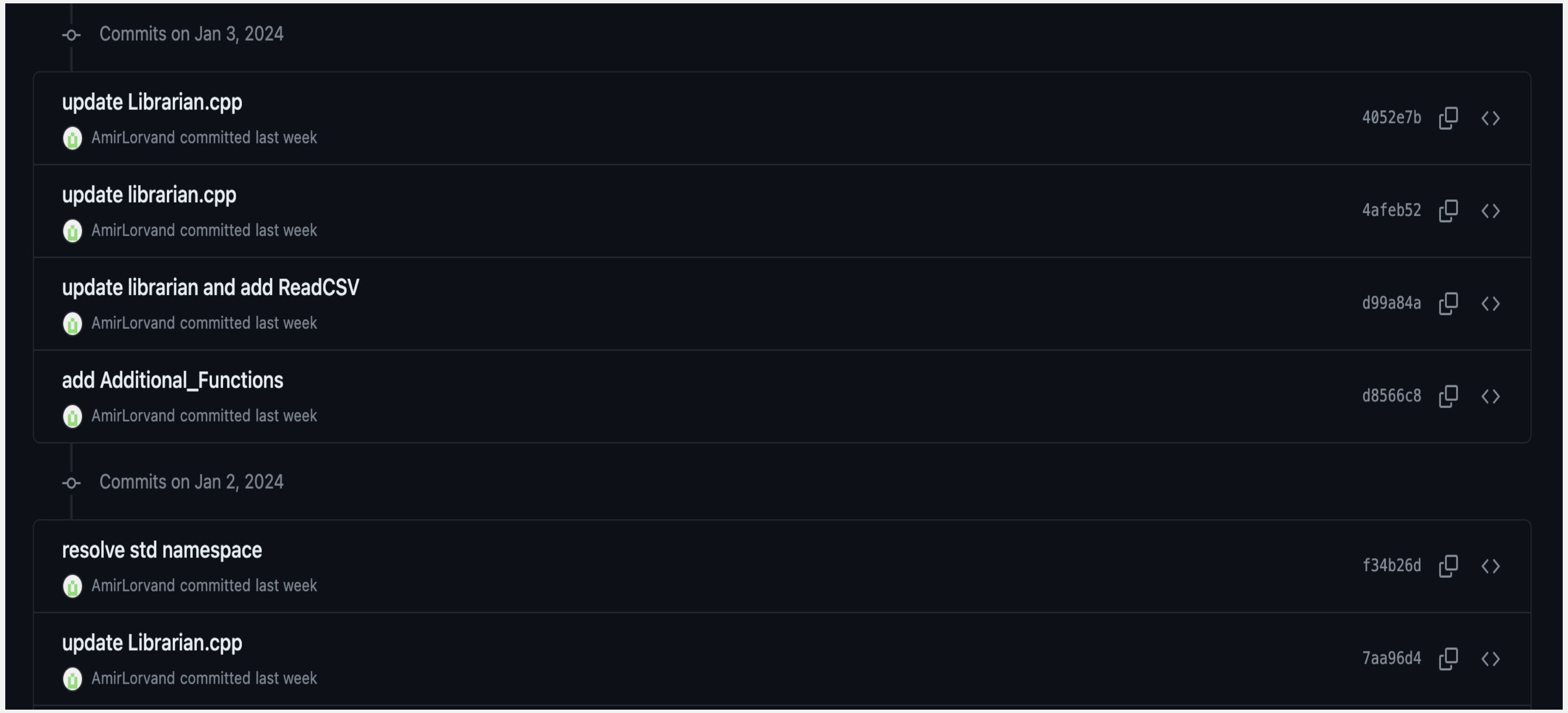
add Librarian class
AmirLorvand committed 2 weeks ago
12a7eac

add perosn class
Amir Lorvand authored and Amir Lorvand committed 2 weeks ago
0cf96d4

M00872834

IMPLEMENTATION

SCREENSHOT OF GITHUB COMMITS



Commits on Jan 3, 2024

update Librarian.cpp
AmirLorvand committed last week
4052e7b

update librarian.cpp
AmirLorvand committed last week
4afeb52

update librarian and add ReadCSV
AmirLorvand committed last week
d99a84a

add Additional_Functions
AmirLorvand committed last week
d8566c8

Commits on Jan 2, 2024

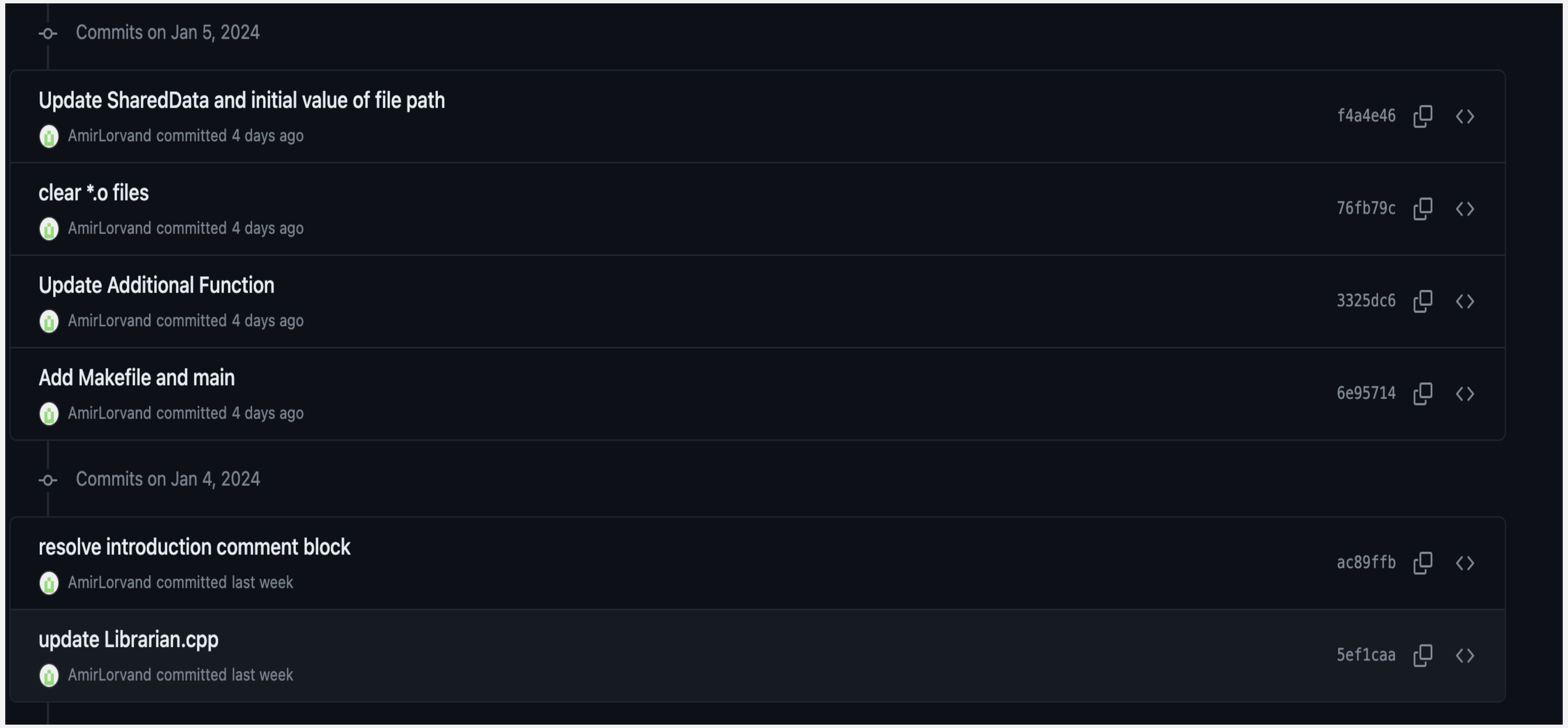
resolve std namespace
AmirLorvand committed last week
f34b26d

update Librarian.cpp
AmirLorvand committed last week
7aa96d4

M00872834

IMPLEMENTATION

SCREENSHOT OF GITHUB COMMITS



Commits on Jan 5, 2024

Update SharedData and initial value of file path
AmirLorvand committed 4 days ago
f4a4e46

clear *.o files
AmirLorvand committed 4 days ago
76fb79c

Update Additional Function
AmirLorvand committed 4 days ago
3325dc6

Add Makefile and main
AmirLorvand committed 4 days ago
6e95714

Commits on Jan 4, 2024

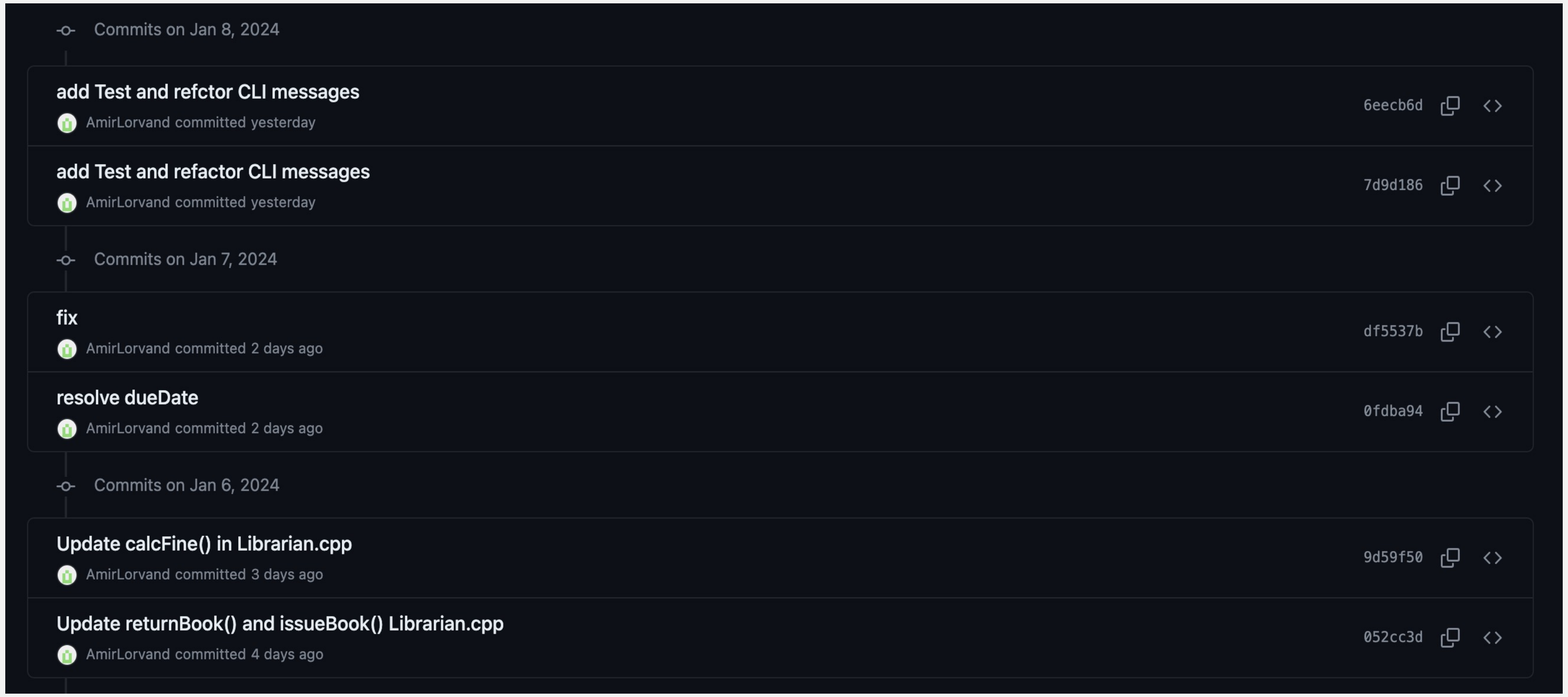
resolve introduction comment block
AmirLorvand committed last week
ac89ffb

update Librarian.cpp
AmirLorvand committed last week
5ef1caa

M00872834

IMPLEMENTATION

SCREENSHOT OF GITHUB COMMITS



Commits on Jan 8, 2024

add Test and refctor CLI messages
AmirLorvand committed yesterday
6eecb6d

add Test and refactor CLI messages
AmirLorvand committed yesterday
7d9d186

Commits on Jan 7, 2024

fix
AmirLorvand committed 2 days ago
df5537b

resolve dueDate
AmirLorvand committed 2 days ago
0fdba94

Commits on Jan 6, 2024

Update calcFine() in Librarian.cpp
AmirLorvand committed 3 days ago
9d59f50

Update returnBook() and issueBook() Librarian.cpp
AmirLorvand committed 4 days ago
052cc3d

M00872834

TESTING APPROACH
STRATEGY FOR TESTING

- **Void functions:**
- Using Test & Run
- For example, create a member and check that if the member has been created or not using a main function.

- **Functions have return value:**
- Using catch2

TESTING APPROACH

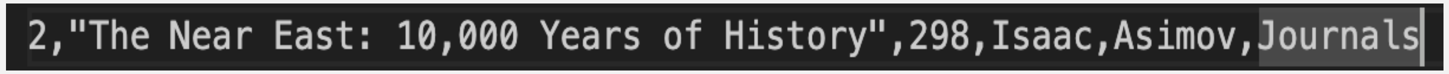
HOW TO APPLY THIS STRATEGY

- **Void functions:**
- It was checked at the same time as writing the code

- **Functions have return value:**
- Writing test case using catch2
- Check that the expected output match the output
- For example, test the input validation

TESTING APPROACH
WHAT WAS BEING TESTED

- **Comma problem in book names:**
- A serious problem was found in arrangement of columns in CSV files by a test case in catch2
- When a book has a comma in it

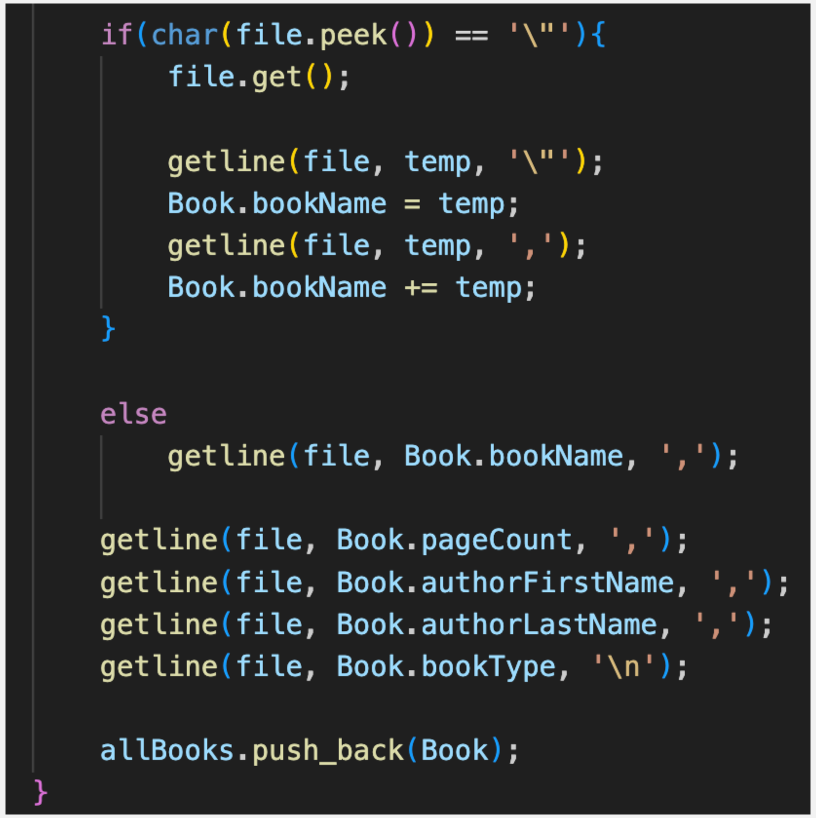


```
2,"The Near East: 10,000 Years of History",298,Isaac,Asimov,Journals
```

M00872834

TESTING APPROACH
WHAT WAS BEING TESTED

- **Solution:**



```
    if(char(file.peek()) == '\"'){
        file.get();

        getline(file, temp, '\"');
        Book.bookName = temp;
        getline(file, temp, ',');
        Book.bookName += temp;
    }

    else
        getline(file, Book.bookName, ',');

    getline(file, Book.pageCount, ',');
    getline(file, Book.authorFirstName, ',');
    getline(file, Book.authorLastName, ',');
    getline(file, Book.bookType, '\n');

    allBooks.push_back(Book);
}
```

M00872834

CONCLUSION

LIMITATION OF THE YOUR WORK

- **Limited Functionality:**
- Include advanced search options for books, but couldn’t due to time limitation.
- (it is implemented but not being used)

- **User Interface:**
- This system uses CLI, more user-friendly if it could be GUI.

- **Scalability:**
- Do not know the system can handle a significant increase in data volume or not?

CONCLUSION

HOW TO HANDLE A SIMILAR PROJECT IN FUTURE

- **Early Planning:**
- Consider requirements and potential challenges

- **Enhanced Testing:**
- Plan for more comprehensive testing.

- **User-Centric Design:**
- Focus more on the user experience by involving potential users in the design process and collecting feedback for UI/UX improvements.