Coursework-1 (CST 2550)

Student id: M00881174

Student name: Divyani Rana

Title: Library Management system

Objective: Our goal is to create a small library system that keeps the records of the books and its members efficiently in a small library, the system is allowing its members to return and borrow books within a three day time period. We have been provided with the csv file that contains information of the available books in the system , the system is flexible to work with this file and other files too.

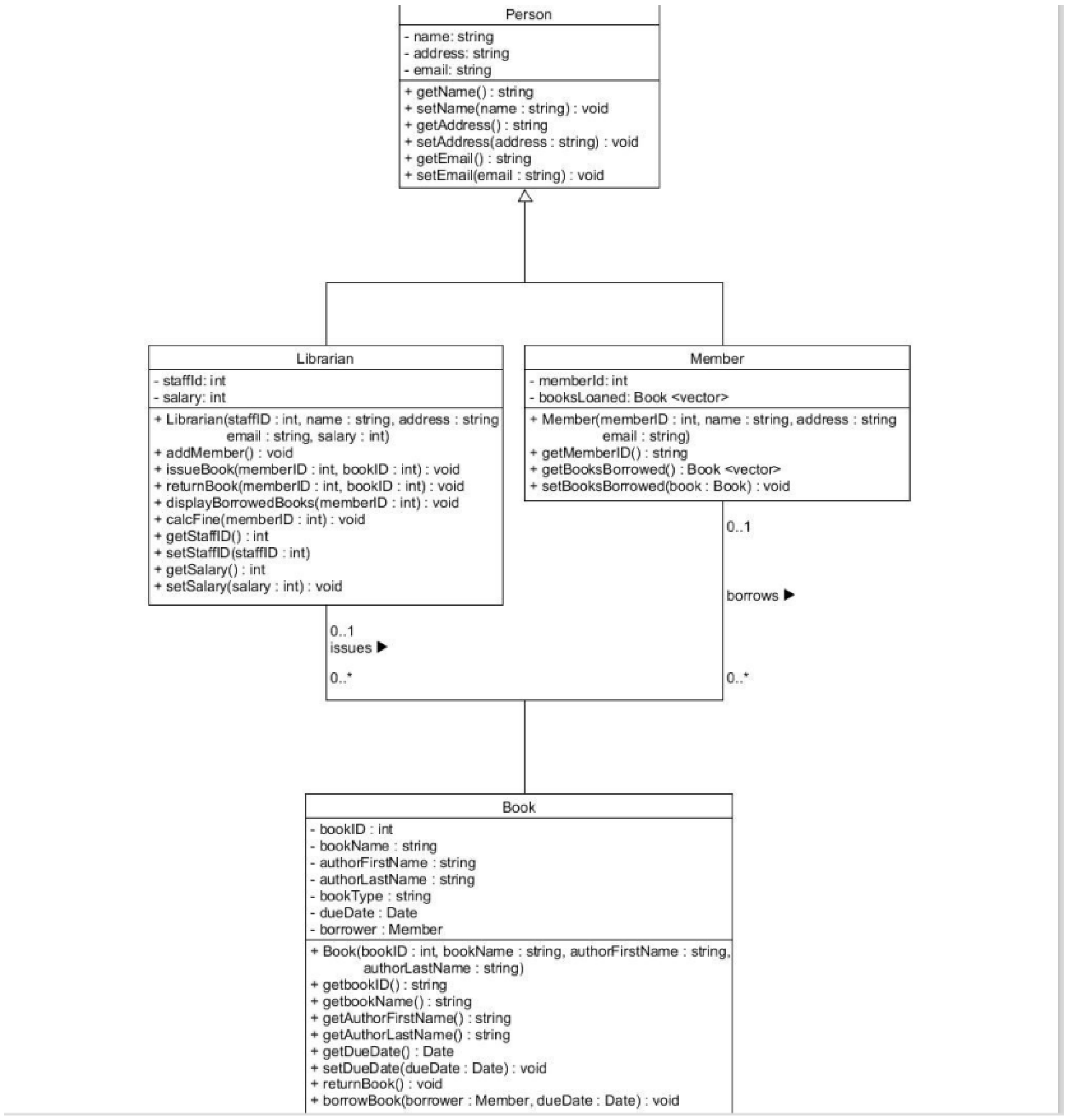
Overview:

I will be talking about the components, testing , design overview, code structure, in my presentation.

Design: this is the UML diagram for the library management system.

In the UML diagram we have four classes:

1.Person: representing common attributes shared by member, librarian, and book, this is our abstract class.
2. Book: represents a book with details such as id , author, name, due date and borrower.
3. Member: represents a library member derived from the person class with a unique member id and the list of borrowed books.
4. Librarian: represents a librarian derived from the person class with a staff id , salary, and a list of members.
5. show interface: represents the user interface class handling the interactions between the users and the system, and it doesn’t inherit from the person class.
In the librarian class, the librarian will be interacting with the functions like adding members , borrowing books to them and , returning books from them and charging fines for all overdue books. Additionally I have the student class as well in which the student can view the booklist etc.

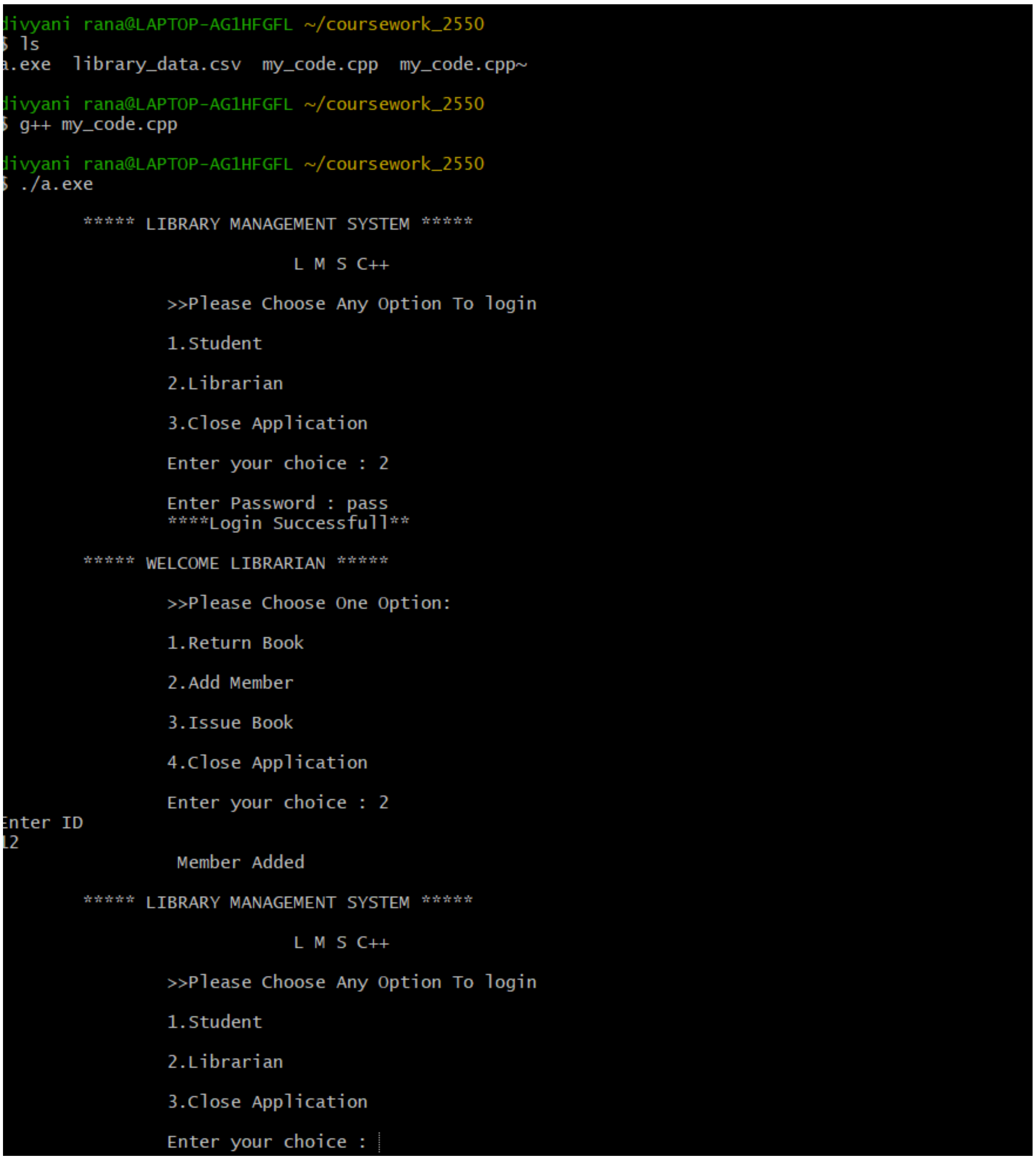

Person
- name: string
- address: string
- email: string
+ getName() : string
+ setName(name : string) : void
+ getAddress() : string
+ setAddress(address : string) : void
+ getEmail() : string
+ setEmail(email : string) : void
Librarian
- staffId: int
- salary: int
+ Librarian(staffID : int, name : string,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

Implementation: for converting the design into working software I followed the following steps:

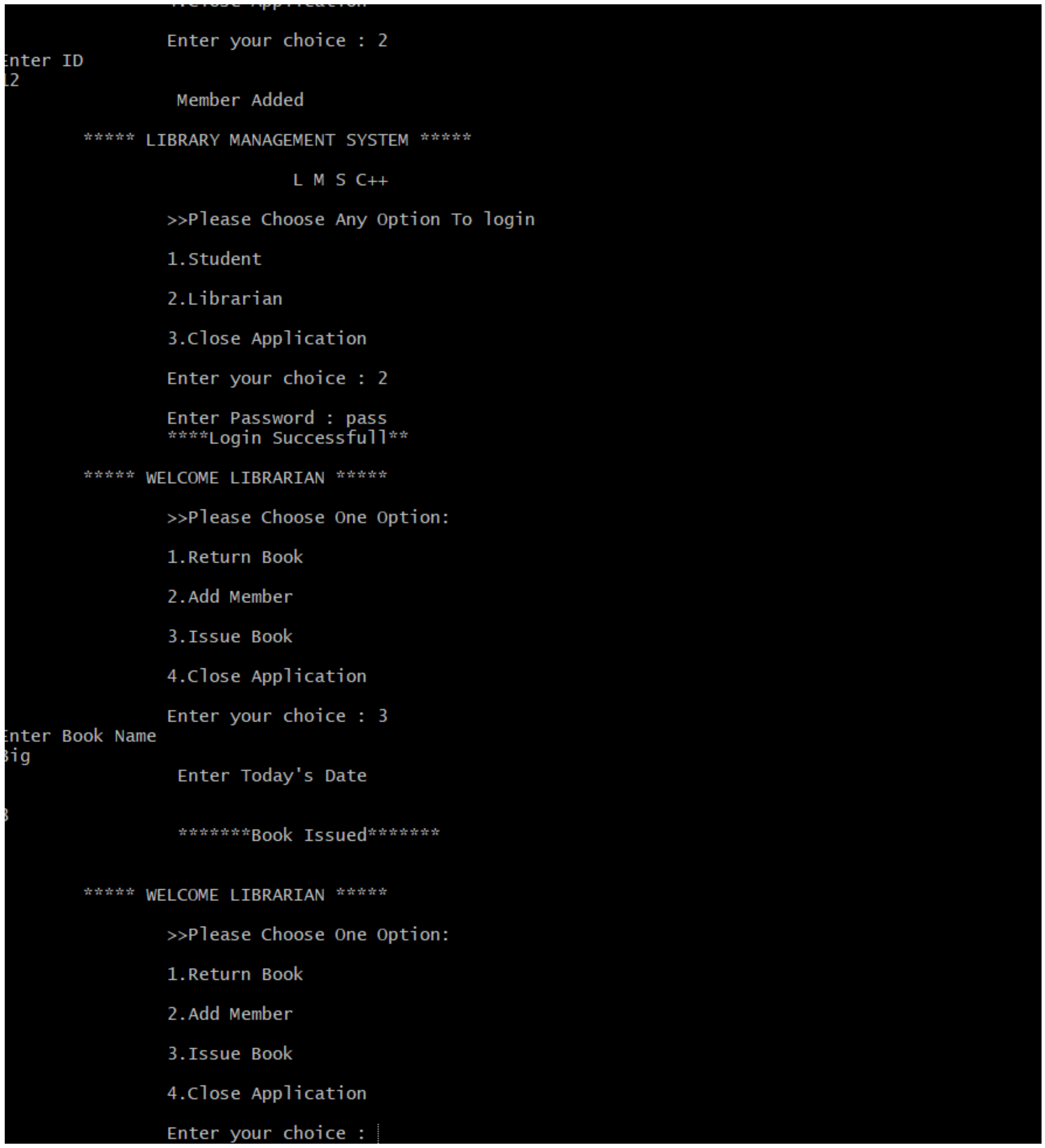
1. I understood the requirements of the design which includes its features like students , librarians, books .

2. For the code organization I just prepared a single file as not to be confused among different files as it limits my organization for the files , I kept it simple.
3. I implemented the classes according to the requirements relating methods and attributes to them and defining the relationship as well based on the UML diagram.
4. I implemented file handling for reading data from the csv file following the format as I organised my main code file and the csv file in the same folder.
5. Did some error handling to manage unexpected situations by providing user friendly messages.
6. Added comments to explain code functionality .

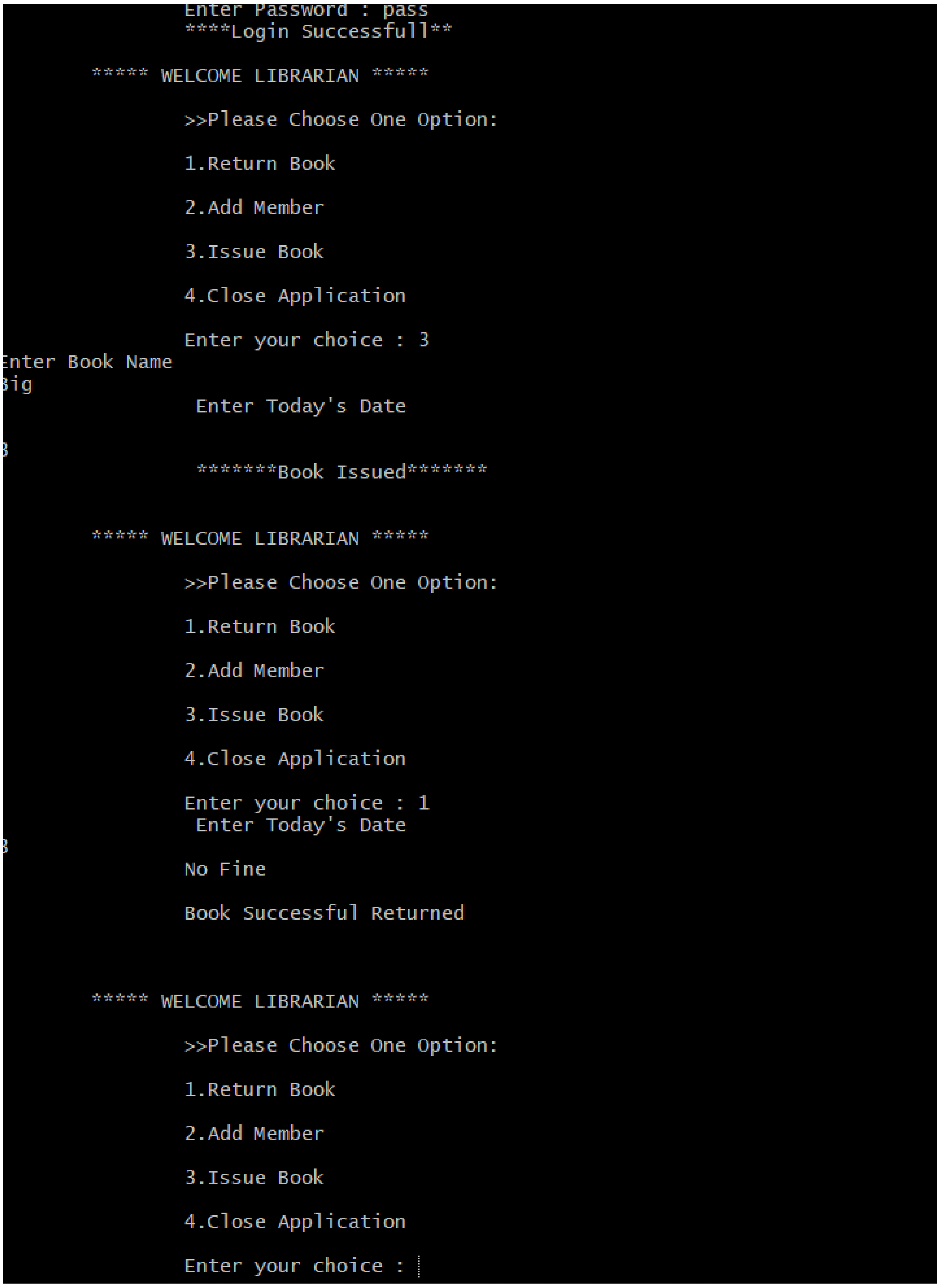
```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <ctime>
#include <set>

// define a set to store member ids
std::set<int> mySet;
//vectors to store book names and tokens from the csv file
std::vector<std::string> Bookname;
std::vector<std::string> tokens;
// variable to store the data of book issue
int dateOfIssue;
class Person;

class Book;
// Creating Date Class to represent date info
class Date
{
public:
    int day, month, year;
};
// Creating Person Class
class Person
{
private:
    std::string name;
    std::string address;
    std::string email;

public:
  //constructors and getter setters for name , address and email
    Person() : name(""), address(""), email("") {}

    Person(const std::string &name, const std::string &address, const std::string &email)
        : name(name), address(address), email(email) {}
```



```
divyani rana@LAPTOP-AG1HFGFL ~/coursework_2550
$ ls
a.exe   library_data.csv   my_code.cpp   my_code.cpp~

divyani rana@LAPTOP-AG1HFGFL ~/coursework_2550
$ g++ my_code.cpp

divyani rana@LAPTOP-AG1HFGFL ~/coursework_2550
$ ./a.exe

        ***** LIBRARY MANAGEMENT SYSTEM *****

                            L M S C++

                >>Please Choose Any Option To login

                1.Student

                2.Librarian

                3.Close Application

                Enter your choice : 2

                Enter Password : pass
                ****Login Successfull**

        ***** WELCOME LIBRARIAN *****

                >>Please Choose One Option:

                1.Return Book

                2.Add Member

                3.Issue Book

                4.Close Application

                Enter your choice : 2
Enter ID
12

                 Member Added

        ***** LIBRARY MANAGEMENT SYSTEM *****

                            L M S C++

                >>Please Choose Any Option To login

                1.Student

                2.Librarian

                3.Close Application

                Enter your choice :
```



```
                Enter your choice : 2
nter ID
12
                 Member Added

        ***** LIBRARY MANAGEMENT SYSTEM *****

                            L M S C++

                >>Please Choose Any Option To login

                1.Student

                2.Librarian

                3.Close Application

                Enter your choice : 2

                Enter Password : pass
                ****Login Successfull**

        ***** WELCOME LIBRARIAN *****

                >>Please Choose One Option:

                1.Return Book

                2.Add Member

                3.Issue Book

                4.Close Application

                Enter your choice : 3
nter Book Name
ig
                 Enter Today's Date


                 *******Book Issued*******


        ***** WELCOME LIBRARIAN *****

                >>Please Choose One Option:

                1.Return Book

                2.Add Member

                3.Issue Book

                4.Close Application

                Enter your choice :
```

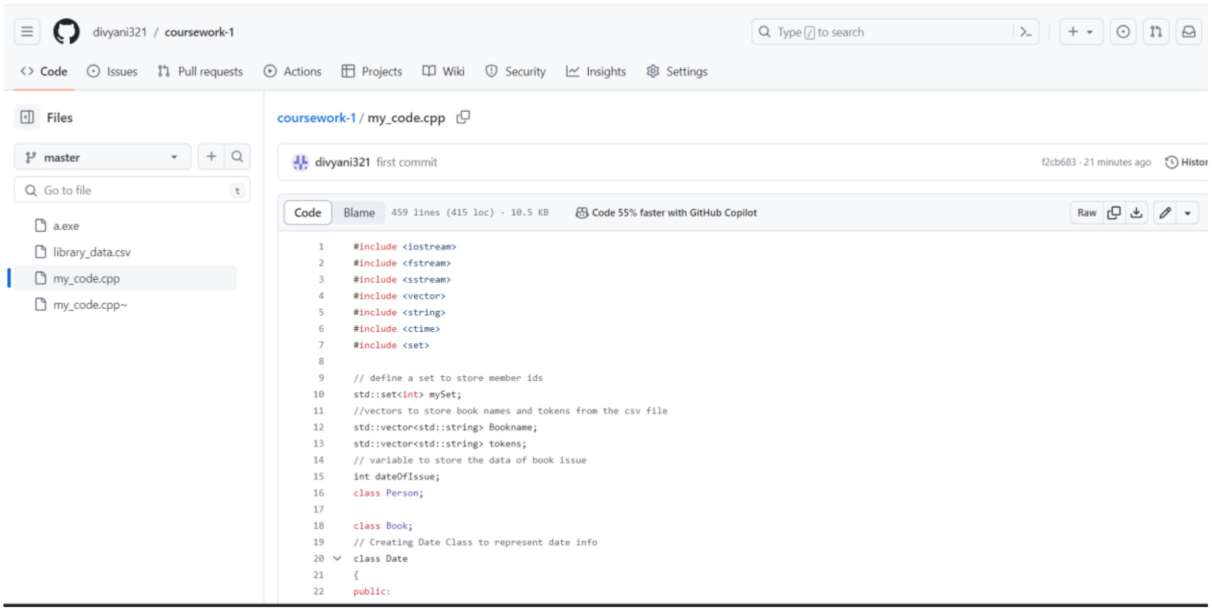


```
               Enter Password : pass
               ****Login Successfull**

       ***** WELCOME LIBRARIAN *****

               >>Please Choose One Option:

               1.Return Book

               2.Add Member

               3.Issue Book

               4.Close Application

               Enter your choice : 3
nter Book Name
ig
                Enter Today's Date



                *******Book Issued*******


       ***** WELCOME LIBRARIAN *****

               >>Please Choose One Option:

               1.Return Book

               2.Add Member

               3.Issue Book

               4.Close Application

               Enter your choice : 1
                Enter Today's Date


               No Fine

               Book Successful Returned


       ***** WELCOME LIBRARIAN *****

               >>Please Choose One Option:

               1.Return Book

               2.Add Member

               3.Issue Book

               4.Close Application

               Enter your choice :
```

Make file: we are using makefile in the system for the efficiency . readability , to integrate with version control systems and for the customization allowing additional project specific tasks in the build process

Also it helps in deciding which parts of the program need to be recompiled.

Version control: since dealing with the large codes for making this library management system in c++ and adding it to the git was required so to edit the code again and again then we can commit it easily to the git ensures that correct edited code has been transferred to the git also it ensures that for the safety we have the previous version of the code as well , hence for these reasons we used version control.



```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <ctime>
#include <set>

// define a set to store member ids
std::set<int> mySet;
//vectors to store book names and tokens from the csv file
std::vector<std::string> Bookname;
std::vector<std::string> tokens;
// variable to store the data of book issue
int dateOfIssue;
class Person;

class Book;
// Creating Date Class to represent date info
class Date
{
public:
```

Test approach: there are many tests approach that I have been doing in the system:

1.testing methods and functions within classes.

2. testing interactions between the classes eg , librarian adding a member.

3. ensuring file handling operations , like reading from the csv file.

4. verifying error handling when invalid inputs are provided.

5. testing the addition of members, issuance and returning of books to ensure correct data handling.

6. testing date calculations and fines .

s.no	Actions	Inputs	Expected output	Actual output	Test result
1.	Enter password	****	Login success	Login successful	pass
2.	Return book	1	Book returned	Book returned	pass
3.	Issue book	3	Book issued	Book issued	pass
4.	Borrow book	2	Book already	Book already borrowed	Pass

			borrowe d		

Implementation of the program:

1. Firstly the librarian is prompted to enter the password to access the system.
2. After successfully login , he is presented with the options of return book, adding member, issuing book, and closing the application.
3. If the librarian chooses to add the member he has to provide the id of the member to be added to the system.
4. If the librarian selects to issue books to members , he enters the name of the book to be issued like Big, Nature etc.
5. If the book is found , he is asked to enter todays date.
6. Then the book is marked as issued with the due date and the success message is displayed.
7. If the librarian selects to return a book from a member , the system first checks the due date of the borrowed books and if any book is overdue , a fine is calculated and displayed as 1 pound for each overdue day.
8. The librarian can choose to close the application.

9. The student can also access the system by looking into the booklist , search for a book from the csv file and closing the application at the end.

Conclusion:

1. Class hierarchy: the code defines the classes like person, book , members, and librarian.
2. Functionality: the book class handles operations such as borrowing and returning books.

member class defines adding members which further keeps track of the borrowed books.

The librarian class which manages members issuing, borrowing books and adding members and charging fines.

3. File handling: the system reads information from the csv file about the books.
4. Password protection: the code implements a password protection mechanism for the librarian access to the system.
5. Data handling: the code manages the data handling for the books overdue and the fine calculations from the users.

Limitations of my work: had some issues in committing my work to the git for the version control which resulted in code mismatches and deleted files , recreating them with some edits, minimal error handling which was not giving me the correct results for the different scenarios.

Approach for future projects:

I will ensure that ill manage the version control at the beginning of the project committing the initial code so later I can push any changes to avoid permanent removal of files from the Cygwin and emacs.