**Student ID #M00774667**

Coursework 1:

Final submission for CST2550

Contents

[Description 3](#_Toc156173692)

[Content 3](#_Toc156173693)

[Diagrams 3](#_Toc156173694)

[Header files and classes 3](#_Toc156173695)

[Person.h and Person.cpp 4](#_Toc156173696)

[Member.h and Member.cpp 4](#_Toc156173697)

[Book.h and Book.cpp 4](#_Toc156173698)

[Librarian.h and Librarian.cpp 4](#_Toc156173699)

[Difficulties and Conclusions 5](#_Toc156173700)

[Screenshots 5](#_Toc156173701)

# Description

This submission is aimed to describe the code implementation for a book and member management system for a small library which allows the librarian to provide books to a member, return the book from the member, display all the books borrowed by the member, and calculate a fine if the book is returned after three days. This program was developed using C++ with an object-oriented approach, implementing four classes (Person, Member, Librarian and Book), the use of a makefile for the compiling order, a tester to check for the right functionality of each class, and the use of diagrams to visualise the approach to develop the program.

The program starts with a welcome message and an input request for the csv that will be used for the library, followed by a check for the root details of the program. Once the root librarian is confirmed, a menu of choices will appear. The choices won’t be available except for the choice to add a member. Afterwards, all the other options will be fully available to either issue a book, return a book or display all the books borrowed by the user. There is an extra option, which is to exit the program.

# Content

## Diagrams

The UML diagram (Fig.1) as the base of the project, specifies the four classes used, the private attributes and the public operations for each class, some of these classes will contain a constructor, the inheritance between classes and the cardinality.

The Use Case diagram (Fig.2) describes the high-level functions and scope of the system. The main actor responsible of what the system does in this case scenario, will be the Librarian.

The Activity diagram (Fig.3) demonstrates the steps that the system will be taking to accomplish all the desired functionality of the system.

## Header files and classes

A header file (.h) and a source code file (.cpp) was created for each class. The header file includes the declared private attributes and the public functions for the specific class. However, all the implementation of the functions will occur in the source code file.

### Person.h and Person.cpp

The Person.h file includes the iostream, string and regex library. It has also declared the variables name , email and address, and functions to set and get these values. The Person.cpp file includes the Person.h file and implements the set or get functions on the variables declared in this class.(Fig.4)

### Member.h and Member.cpp

The Member.h file includes the Person.h file to include all the libraries in the Person.h file and to be able to specify that the Member class is a child of the Person class. Member.h also includes the vector library and a forward declaration of the Book class to avoid compiling errors. This file declares the variable memberID, a vector called “booksLoaned” for the pointer of book objects, a constructor to add all the values necessary for each member object and one function to add Book pointers into the vector. Outside the class, there is an external function to store all the members that will be created.

In the Member.cpp implementation, the private attributes of the member are the values required for the constructor, set and get functions are available for the variables and the vector of Book pointers. (Fig.5)

### Book.h and Book.cpp

The Book.h file also includes another header file to include all the previous libraries too, in this case, it is the Member.h as it is required for one of the operations in the Book.h file. Book.h will also include the ctime and fstream library. The Book.h file will declare seven private attributes, one constructor and eight public operations, an extra operation was added to get the borrower pointer of the book, and an external function out of the class to store all the books created from the csv file.

The Book.cpp file will implement the private attributes as the values required for the constructor, set and get functions for the attributes and the date that each book is due, functions to add pointers of members into a vector to specify if the book is being borrowed or not, and a function to return the book. (Fig.6)

### Librarian.h and Librarian.cpp

The Librarian.h file to include all the previous headers and libraries, it includes the Book.h, it also specifies that the Librarian class is a child from the Person class. This header declares the variables staff ID and salary, one constructor, a function to add the member, one to issue and one to return the book, one to display all the borrowed books, one to calculate a fine, and four to set or get the variables of the Librarian class.

In the implementation carried in the Librarian.cpp, the constructor will use the private attributes as the variables for the constructor. The add member function will request for information that will validate with a regex check, to after use it to create a member object. To issue a book, it will add the book in the member information, and it will make the book unavailable for other users. The return book function will check in the member information for the specific book being returned, once the book is returned, it will allow the user to use the calculate the fine function, check whether there is any fees to pay and set the book as returned. And the display borrowed books functions will display all the books currently borrowed by the user. (Fig.7)

# Difficulties and Conclusions

The difficulties through the process of creating the classes and manipulating the information of each member and each book, occurred when the book requested had to be returned and remove from the vector of pointers. The use of pointers could be beneficial if there is complete access to all the pointers, for them to be manipulated.

To conclude, the creation of this library management system was crucial for the improvement of skills in C++, OOP(Object-Oriented Programming), and the use of pointers. The library system will be considered as a base for future improvements.

# Screenshots

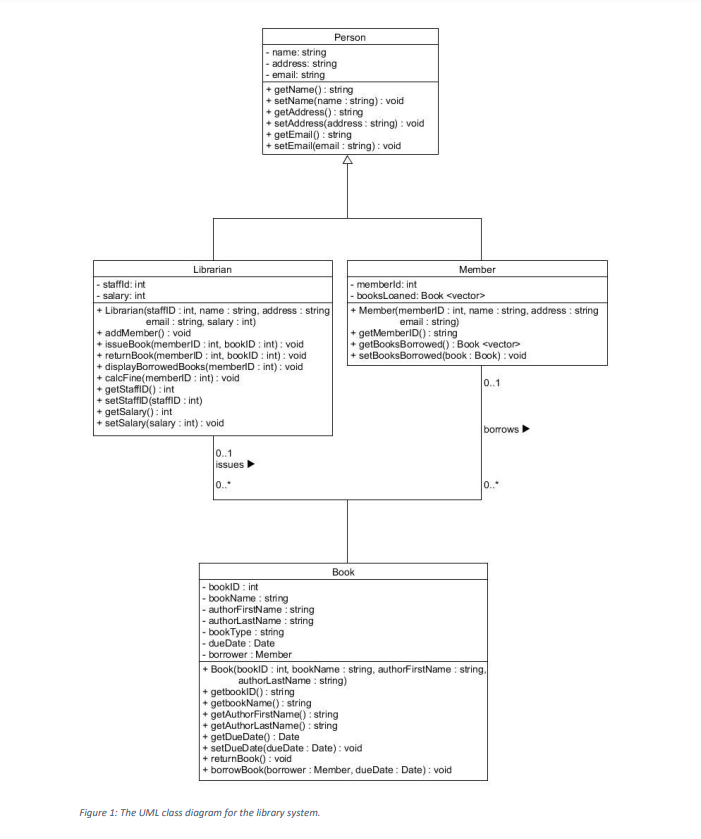


Fig. The UML Diagram

A diagram of a library management system

Description automatically generated

Fig. Use Case Diagram

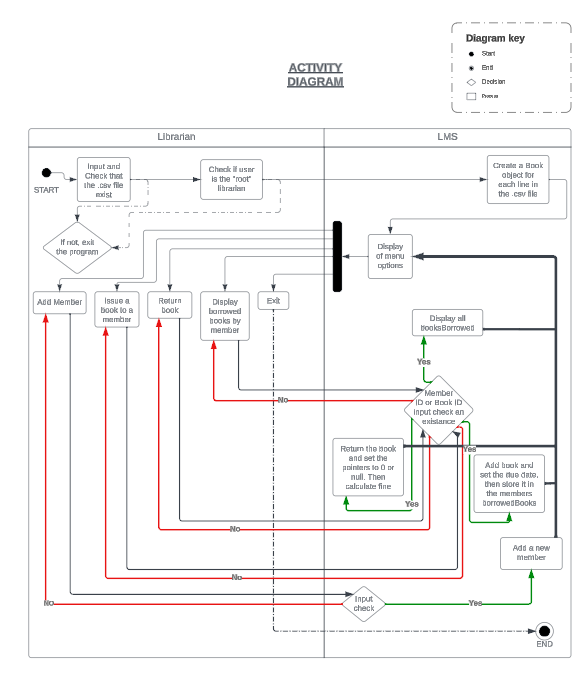


Fig. Activity Diagram

A screenshot of a computer screen

Description automatically generated

Fig. Person.h Person.cpp

A screenshot of a computer program

Description automatically generated

Fig. Member.h Member.cpp

A screenshot of a computer screen

Description automatically generated

Fig. Book.h Book.cpp

A screenshot of a computer program

Description automatically generated

Fig. Librarian.h Librarian.cpp

A screenshot of a computer

Description automatically generated

Fig. GitHub commits PART 1

A screenshot of a computer

Description automatically generated

Fig. GitHub commits PART 2

A screenshot of a computer

Description automatically generated

Fig. GitHub commits PART 3