Project Description

Artem Halis. M00951627

How to run program:

After running make command, navigate to bin directory where all .exe files remain. Run any of those. Catch2Tests.exe - Catch2Tests. ManagementSystem.exe - executable program.

General Information:

This project outlines the implementation for a basic library system which allows librarian use many options, such as: *adding a member to a system*,issuing a book to member, returning book by member and calculating fine in case of late return. The whole program was developed using C++ with OOP standards. It contains four classes - Librarian, Member, Book, Person. As well it contains a makefile for compiling purposes, Test file in order to be able to check and imitate functionality of the classes and their functions.

Project overview:

Program starts with a welcome message and checks if the librarian wants to proceed with running a program. Then a book data file request comes. Once the book data file is entered and checked, all menu options will be shown. All choices are available, however a librarian will face errors if no members were added beforehands. After adding a member librarian will be able to issue a book for existing member, return book from member and display all books borrowed by a member.

Project content:

– Diagrams:

UML diagram reveals four classes implemented, private fields, public operations for each class. Some of the classes will have a constructor and inheritance between classes.

Use Case diagram will illustrate the interactions between main actor(Librarian) and other actors. As well it will provide a blueprint for system development.

Activity diagram will illustrate the sequence of activities which will be taken to fulfill the program's functionality.

–Classes and Headers:

For each class(.cpp) a header file(.h) was created. Inside header files we included private fields and public methods(functions). All implementations are defined in .cpp files.

–Person.cpp + Person.h:

Person.h header file has such private fields as: name,address,email.Moreover getters and setters are defined there. Person.cpp includes Person.h file and implements all functions and variables declared in header file.

–Member.cpp + Member.h:

Member.h header file includes Person.h header file in order to have access to Person.h file and define that Member class instantiates from Person class. Member.h also includes vector library. Member.h declares such variables as: member_id and vector of Book pointers called books_loaned. Header file also has constructor and public functions to get member id, function that returns a vector of Book's pointers. called get_books_borrowed. Setter set_books_borrowed, static function called get_list_of_members that returns a reference to a vector of Member objects. Lastly it has function called remove_borrowed_book that takes a pointer to a Book object and removes collection of borrowed books, but the functionality is implemented in Member.cpp file.

–Book.cpp + Book.h:

The `Book.h` header file encapsulates the declaration of the Book class, which represents books in a library management system. It includes libraries such as <ctime>, <string>, and <vector>. Additionally, it depends on the Member.h header file for some operations. The class defines private attributes, including book details and borrowing information. It features a constructor and public operations for accessing and modifying book information, borrowing and returning books, printing all books, and obtaining a list of books. The class also includes a static vector to store all created books and a static function for retrieving this book list. The implementation of these functionalities is provided in the corresponding Book.cpp file. Book.h also includes a forward declaration of Member class to avoid compiling errors.

–Librarian.cpp + Librarian.h:
The Librarian.h header file defines the Librarian class, inheriting from the Person class and including necessary headers such as person.h and member.h. The class represents a librarian in a library management system and declares private variables for staff ID and salary. It provides a constructor, functions to add a member, issue and return books, display borrowed books, calculate fines, and setters/getters for the librarian's variables. Additionally, it includes functions to print member details, manage books, and locate specific members or books. The class is instantiated as a global variable librarian.
–Enums.h:
The Enums.h header file declares an enumeration MenuOption. The enumeration specifies menu options for a library management system, including adding a member, managing books, displaying borrowed books, and ending the current session.
–Makefile:
 Makefile compiles and links whole program consisting of multiple source files.
-CXX and CXXFLAGS:
  - CXX is a variable representing the C++ compiler
  - CXXFLAGS are compiler flags that include warnings (-Wall -Wextra -Wpedantic), dependencies (-MMD -MP for generating dependencies), and include directories (-Iinclude).
- Directories:
  - SRCDIR: Source code directory containing .cpp files.
  - BUILDDIR: Directory which stores object files.
  - BINDIR: Directory which stores the final executables.
- Targets:
  - all: The default target that builds both the ManagementSystem and Catch2tests.
  - $(BINDIR)/Catch2tests: Target for building the Catch2 tests executable.
  - $(BINDIR)/ManagementSystem: Target for building the main executable.
- Object Files:
  - Object files are generated for each source file in the BUILDDIR.

  - For example: $(BUILDDIR)/main.o, $(BUILDDIR)/test.o

- Dependencies:
  - Dependency rules are provided for each object file, specifying the corresponding .cpp file and associated header files.
- Cleaning:
  - clean: A target to remove all generated object files and executables.
- Dependency Generation:
  - The -MMD -MP flags in CXXFLAGS enable the generation of dependency files (.d files) for each source file. These files contain information about header dependencies.

–Difficulties:

I personally had many difficulties in the development process of this project, such as linking together logically all classes and creating necessary functions. However, the one that stands out is removing book from the borrowed books vector. Because I had to deal with pointers while managing Member and Book objects. As such operations often lead to null pointer dereferences. Checking for null pointers before dereferencing helped a lot in my case.
Overall, this project improved my knowledge and skills in C++ and OOP as well as usage of pointers and references.

–Screenshots:

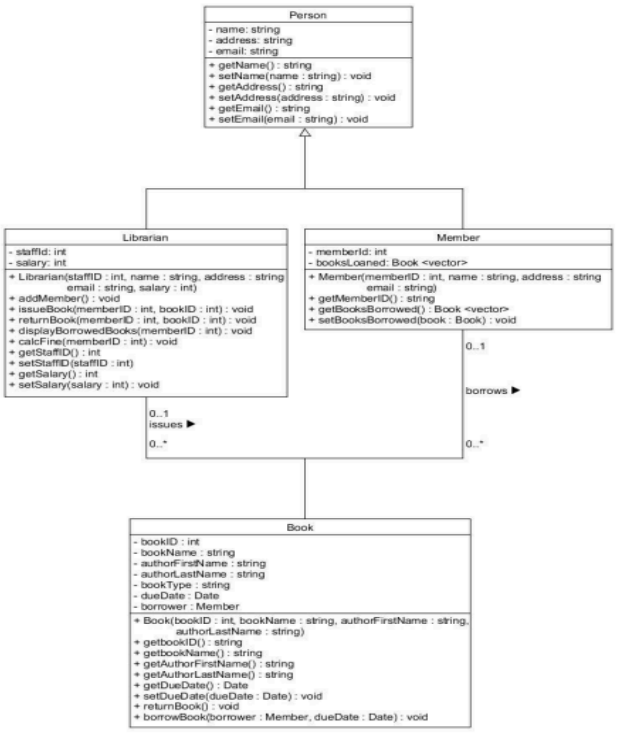
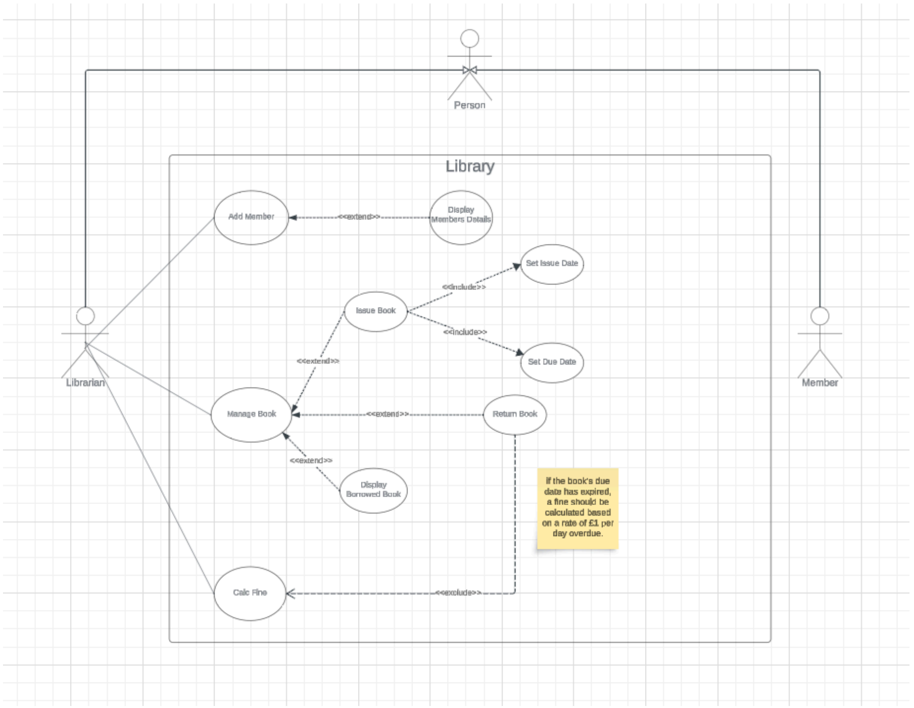
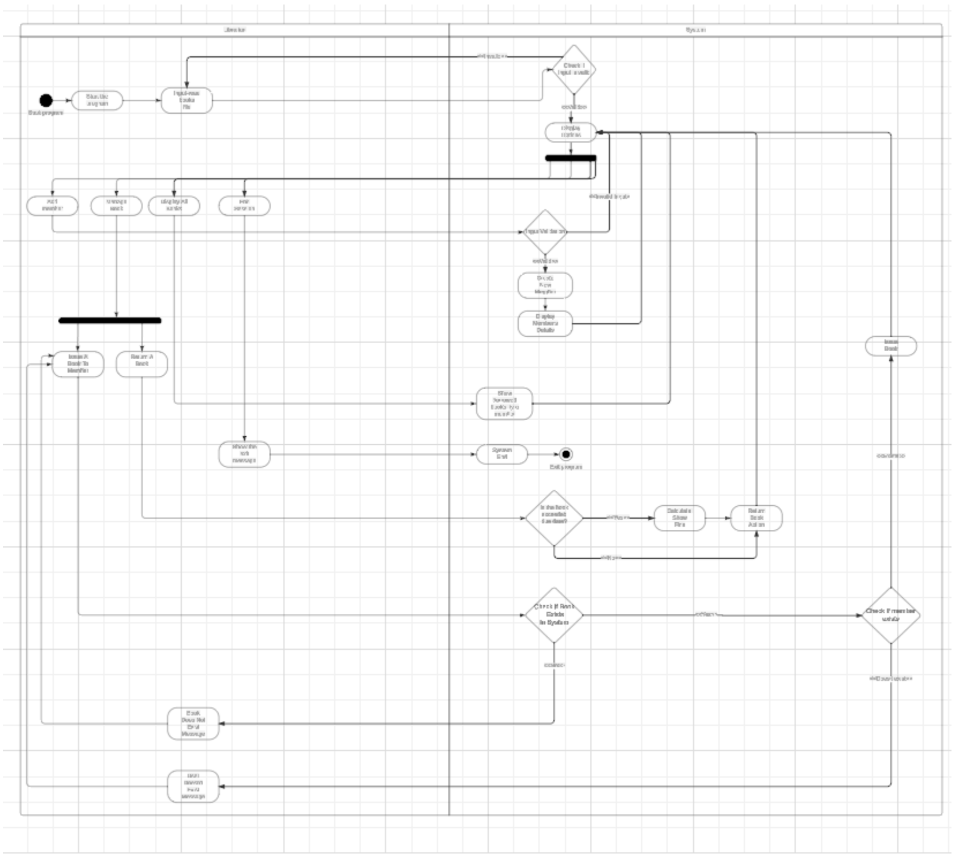

Person
- name: string
- address: string
- email: string
+ getName() : string
+ setName(name : string) : void
+ getAddress() : string
+ setAddress(address : string) : void
+ getEmail() : string
+ setEmail(email : string) : void
Librarian
- staffId: int
- salary: int
+ Librarian(staffID : int, name : string,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

Figure 1: The UML class diagram for the library system.

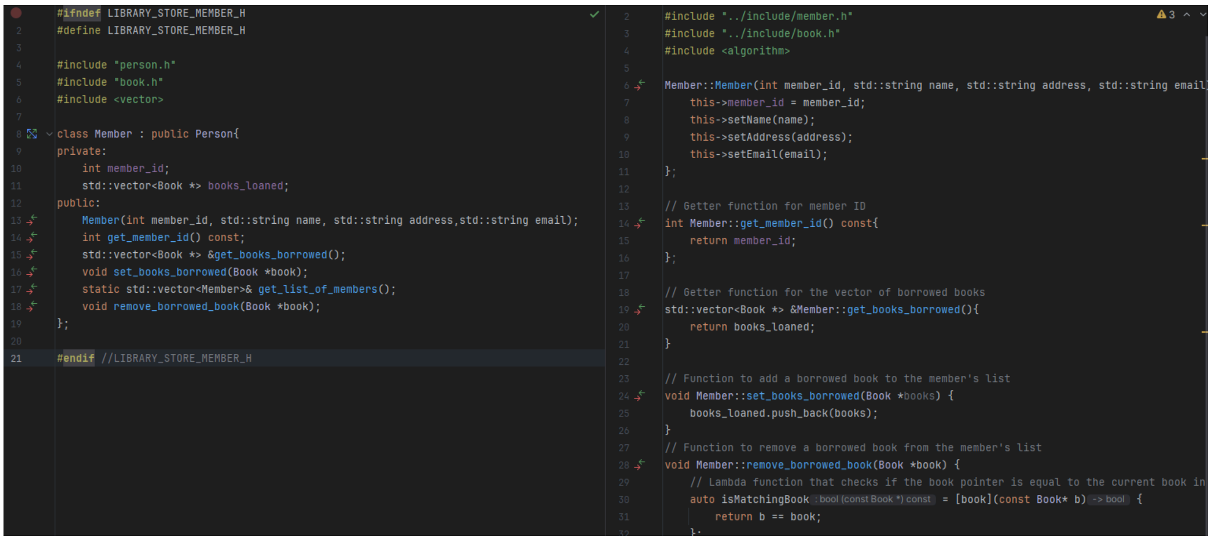
Use Case diagram:



Activity diagram:



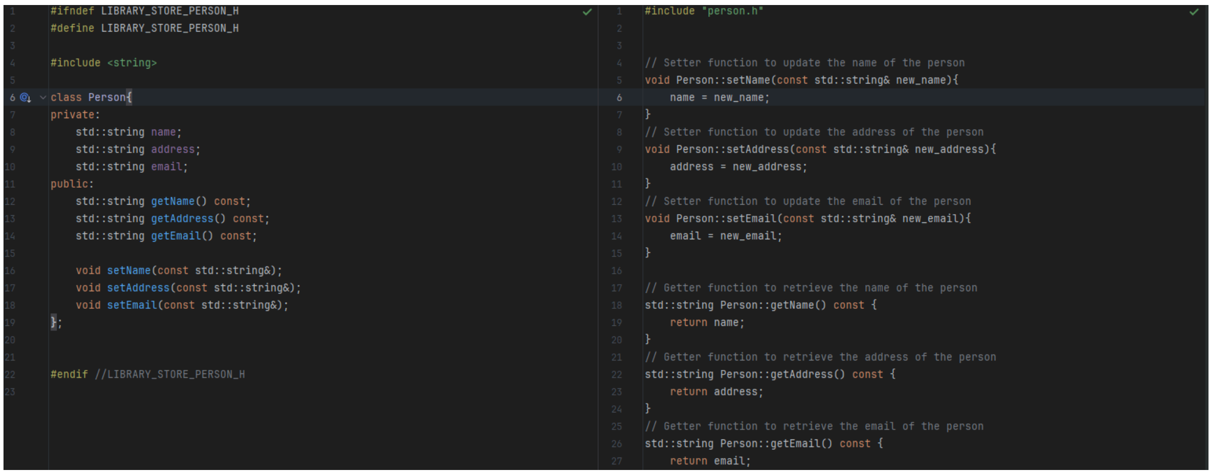
Member.h and Member.cpp



```cpp
#ifndef LIBRARY_STORE_MEMBER_H
#define LIBRARY_STORE_MEMBER_H

#include "person.h"
#include "book.h"
#include <vector>

class Member : public Person{
private:
    int member_id;
    std::vector<Book *> books_loaned;
public:
    Member(int member_id, std::string name, std::string address,std::string email);
    int get_member_id() const;
    std::vector<Book *> &get_books_borrowed();
    void set_books_borrowed(Book *book);
    static std::vector<Member>& get_list_of_members();
    void remove_borrowed_book(Book *book);
};

#endif //LIBRARY_STORE_MEMBER_H
```

```cpp
#include "../include/member.h"
#include "../include/book.h"
#include <algorithm>

Member::Member(int member_id, std::string name, std::string address, std::string email
    this->member_id = member_id;
    this->setName(name);
    this->setAddress(address);
    this->setEmail(email);
};

// Getter function for member ID
int Member::get_member_id() const{
    return member_id;
};

// Getter function for the vector of borrowed books
std::vector<Book *> &Member::get_books_borrowed(){
    return books_loaned;
}

// Function to add a borrowed book to the member's list
void Member::set_books_borrowed(Book *books) {
    books_loaned.push_back(books);
}
// Function to remove a borrowed book from the member's list
void Member::remove_borrowed_book(Book *book) {
    // Lambda function that checks if the book pointer is equal to the current book in
    auto isMatchingBook : bool (const Book *) const = [book](const Book* b) -> bool {
        return b == book;
```

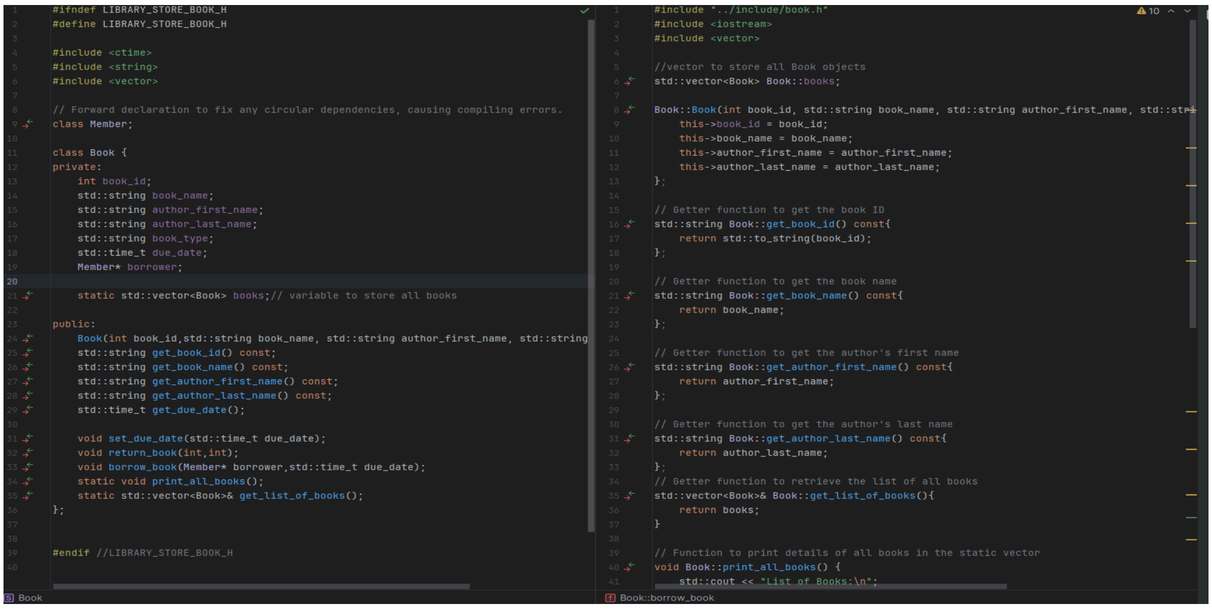
Person.h and Person.cpp



```cpp
#ifndef LIBRARY_STORE_PERSON_H
#define LIBRARY_STORE_PERSON_H

#include <string>

class Person{
private:
    std::string name;
    std::string address;
    std::string email;
public:
    std::string getName() const;
    std::string getAddress() const;
    std::string getEmail() const;

    void setName(const std::string&);
    void setAddress(const std::string&);
    void setEmail(const std::string&);
};


#endif //LIBRARY_STORE_PERSON_H
```

```cpp
#include "person.h"


// Setter function to update the name of the person
void Person::setName(const std::string& new_name){
    name = new_name;
}
// Setter function to update the address of the person
void Person::setAddress(const std::string& new_address){
    address = new_address;
}
// Setter function to update the email of the person
void Person::setEmail(const std::string& new_email){
    email = new_email;
}

// Getter function to retrieve the name of the person
std::string Person::getName() const {
    return name;
}
// Getter function to retrieve the address of the person
std::string Person::getAddress() const {
    return address;
}
// Getter function to retrieve the email of the person
std::string Person::getEmail() const {
    return email;
```

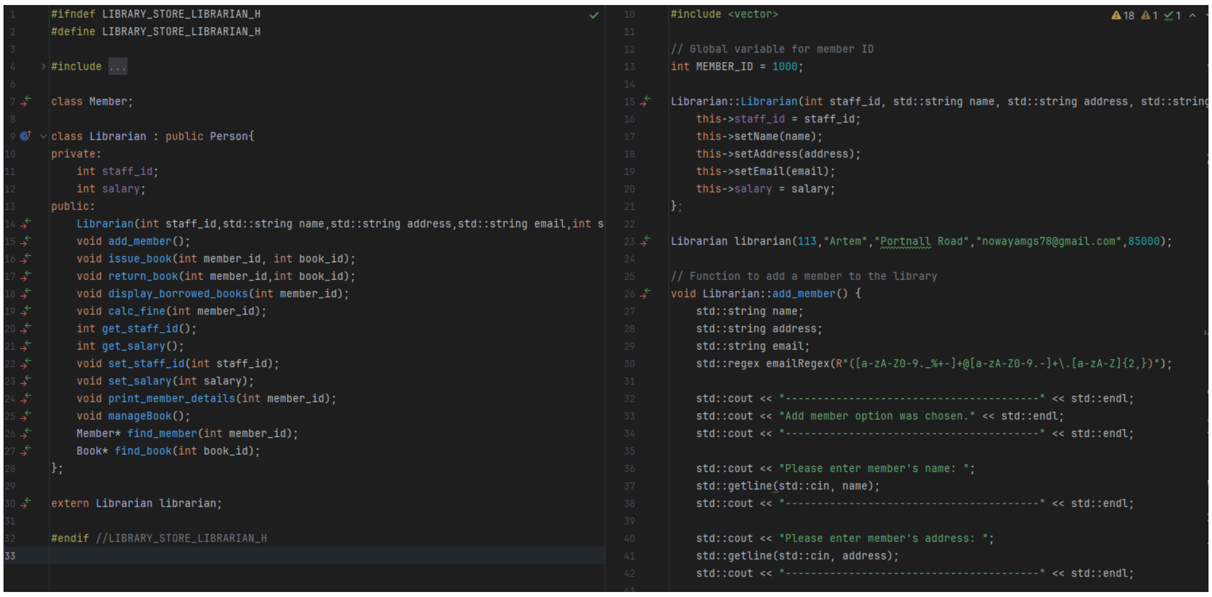
Book.h and Book.cpp



```cpp
#ifndef LIBRARY_STORE_BOOK_H
#define LIBRARY_STORE_BOOK_H

#include <ctime>
#include <string>
#include <vector>

// Forward declaration to fix any circular dependencies, causing compiling errors.
class Member;

class Book {
private:
    int book_id;
    std::string book_name;
    std::string author_first_name;
    std::string author_last_name;
    std::string book_type;
    std::time_t due_date;
    Member* borrower;

    static std::vector<Book> books;// variable to store all books

public:
    Book(int book_id,std::string book_name, std::string author_first_name, std::string
    std::string get_book_id() const;
    std::string get_book_name() const;
    std::string get_author_first_name() const;
    std::string get_author_last_name() const;
    std::time_t get_due_date();

    void set_due_date(std::time_t due_date);
    void return_book(int,int);
    void borrow_book(Member* borrower,std::time_t due_date);
    static void print_all_books();
    static std::vector<Book>& get_list_of_books();
};


#endif //LIBRARY_STORE_BOOK_H

Book
```

```cpp
#include "../include/book.h"
#include <iostream>
#include <vector>

//vector to store all Book objects
std::vector<Book> Book::books;

Book::Book(int book_id, std::string book_name, std::string author_first_name, std::stri
    this->book_id = book_id;
    this->book_name = book_name;
    this->author_first_name = author_first_name;
    this->author_last_name = author_last_name;
};

// Getter function to get the book ID
std::string Book::get_book_id() const{
    return std::to_string(book_id);
};

// Getter function to get the book name
std::string Book::get_book_name() const{
    return book_name;
};

// Getter function to get the author's first name
std::string Book::get_author_first_name() const{
    return author_first_name;
};

// Getter function to get the author's last name
std::string Book::get_author_last_name() const{
    return author_last_name;
};
// Getter function to retrieve the list of all books
std::vector<Book>& Book::get_list_of_books(){
    return books;
}

// Function to print details of all books in the static vector
void Book::print_all_books() {
    std::cout << "List of Books:\n";

Book::borrow_book
```

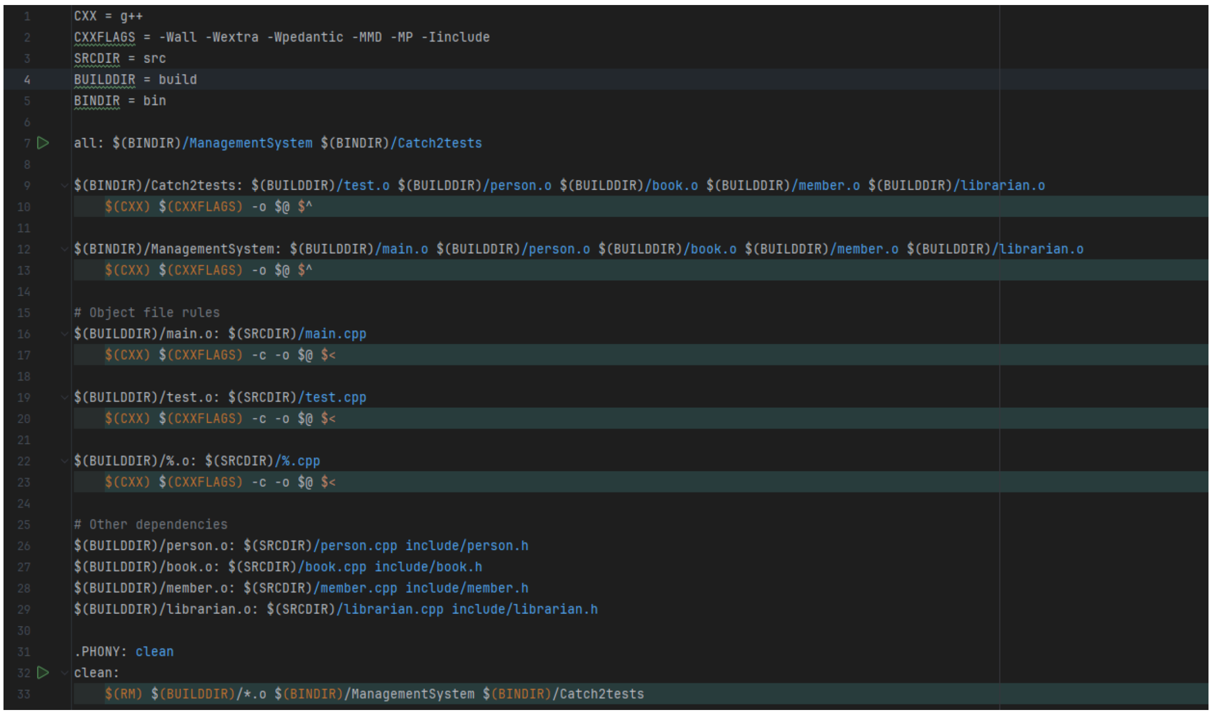
Librarian.h and Librarian.cpp



```cpp
#ifndef LIBRARY_STORE_LIBRARIAN_H
#define LIBRARY_STORE_LIBRARIAN_H

#include ...

class Member;

class Librarian : public Person{
private:
    int staff_id;
    int salary;
public:
    Librarian(int staff_id,std::string name,std::string address,std::string email,int s
    void add_member();
    void issue_book(int member_id, int book_id);
    void return_book(int member_id,int book_id);
    void display_borrowed_books(int member_id);
    void calc_fine(int member_id);
    int get_staff_id();
    int get_salary();
    void set_staff_id(int staff_id);
    void set_salary(int salary);
    void print_member_details(int member_id);
    void manageBook();
    Member* find_member(int member_id);
    Book* find_book(int book_id);
};

extern Librarian librarian;

#endif //LIBRARY_STORE_LIBRARIAN_H
```

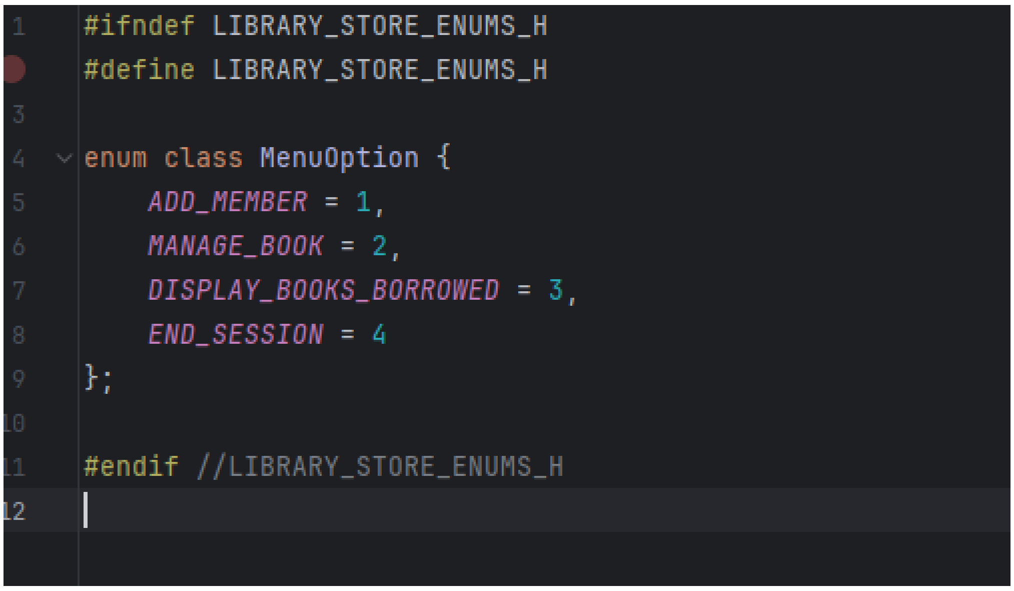
```cpp
#include <vector>

// Global variable for member ID
int MEMBER_ID = 1000;

Librarian::Librarian(int staff_id, std::string name, std::string address, std::string
    this->staff_id = staff_id;
    this->setName(name);
    this->setAddress(address);
    this->setEmail(email);
    this->salary = salary;
};

Librarian librarian(113,"Artem","Portnall Road","nowayamgs78@gmail.com",85000);

// Function to add a member to the library
void Librarian::add_member() {
    std::string name;
    std::string address;
    std::string email;
    std::regex emailRegex(R"([a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,})");

    std::cout << "-----------------------------------------" << std::endl;
    std::cout << "Add member option was chosen." << std::endl;
    std::cout << "-----------------------------------------" << std::endl;

    std::cout << "Please enter member's name: ";
    std::getline(std::cin, name);
    std::cout << "-----------------------------------------" << std::endl;

    std::cout << "Please enter member's address: ";
    std::getline(std::cin, address);
    std::cout << "-----------------------------------------" << std::endl;
```

Makefile



```makefile
CXX = g++
CXXFLAGS = -Wall -Wextra -Wpedantic -MMD -MP -Iinclude
SRCDIR = src
BUILDDIR = build
BINDIR = bin

all: $(BINDIR)/ManagementSystem $(BINDIR)/Catch2tests

$(BINDIR)/Catch2tests: $(BUILDDIR)/test.o $(BUILDDIR)/person.o $(BUILDDIR)/book.o $(BUILDDIR)/member.o $(BUILDDIR)/librarian.o
	$(CXX) $(CXXFLAGS) -o $@ $^

$(BINDIR)/ManagementSystem: $(BUILDDIR)/main.o $(BUILDDIR)/person.o $(BUILDDIR)/book.o $(BUILDDIR)/member.o $(BUILDDIR)/librarian.o
	$(CXX) $(CXXFLAGS) -o $@ $^

# Object file rules
$(BUILDDIR)/main.o: $(SRCDIR)/main.cpp
	$(CXX) $(CXXFLAGS) -c -o $@ $<

$(BUILDDIR)/test.o: $(SRCDIR)/test.cpp
	$(CXX) $(CXXFLAGS) -c -o $@ $<

$(BUILDDIR)/%.o: $(SRCDIR)/%.cpp
	$(CXX) $(CXXFLAGS) -c -o $@ $<

# Other dependencies
$(BUILDDIR)/person.o: $(SRCDIR)/person.cpp include/person.h
$(BUILDDIR)/book.o: $(SRCDIR)/book.cpp include/book.h
$(BUILDDIR)/member.o: $(SRCDIR)/member.cpp include/member.h
$(BUILDDIR)/librarian.o: $(SRCDIR)/librarian.cpp include/librarian.h

.PHONY: clean
clean:
	$(RM) $(BUILDDIR)/*.o $(BINDIR)/ManagementSystem $(BINDIR)/Catch2tests
```

Enums.h



```cpp
#ifndef LIBRARY_STORE_ENUMS_H
#define LIBRARY_STORE_ENUMS_H

enum class MenuOption {
    ADD_MEMBER = 1,
    MANAGE_BOOK = 2,
    DISPLAY_BOOKS_BORROWED = 3,
    END_SESSION = 4
};

#endif //LIBRARY_STORE_ENUMS_H

```