

Introduction

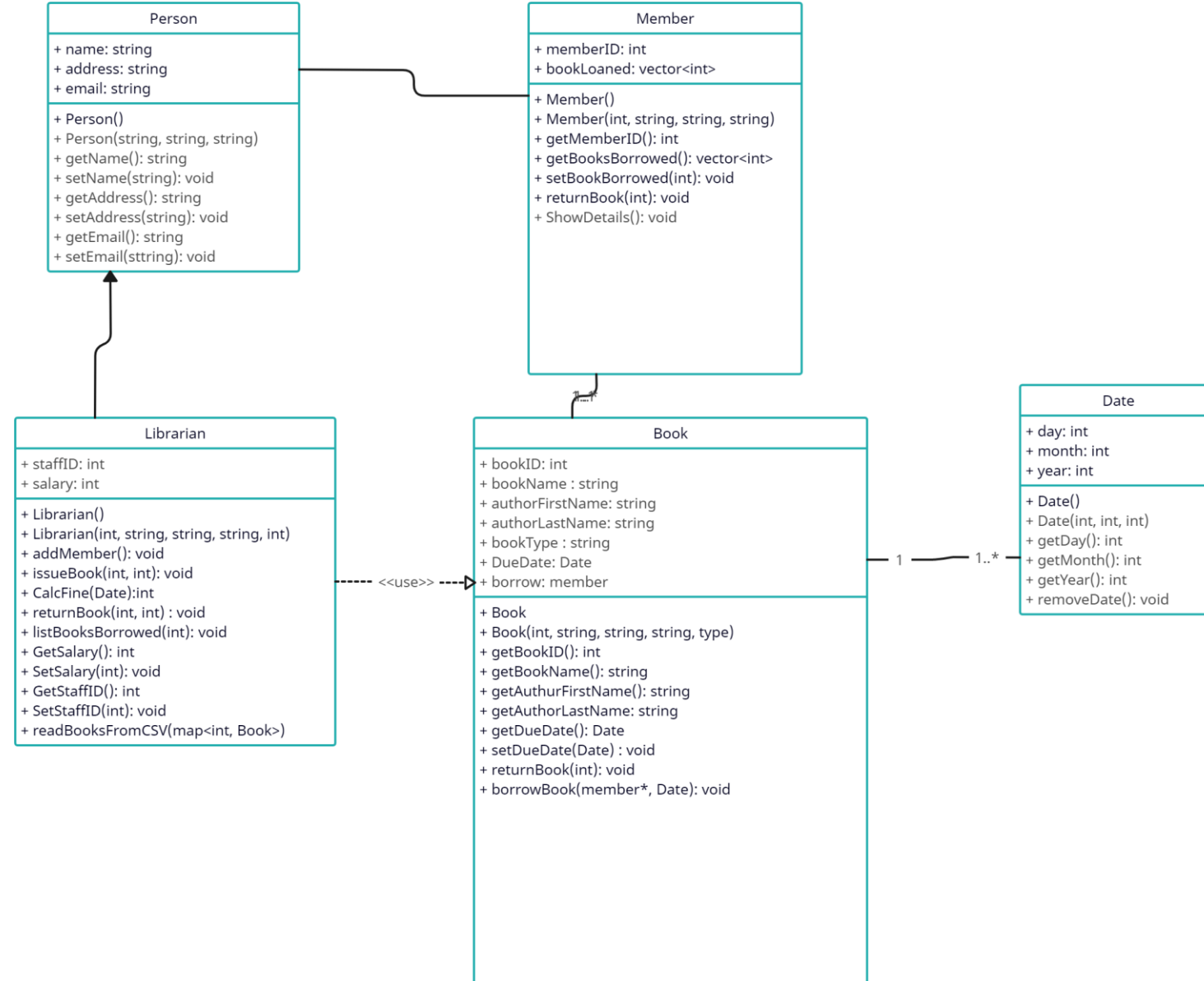
- **Student Number:** M00912138

- **Project Description:**

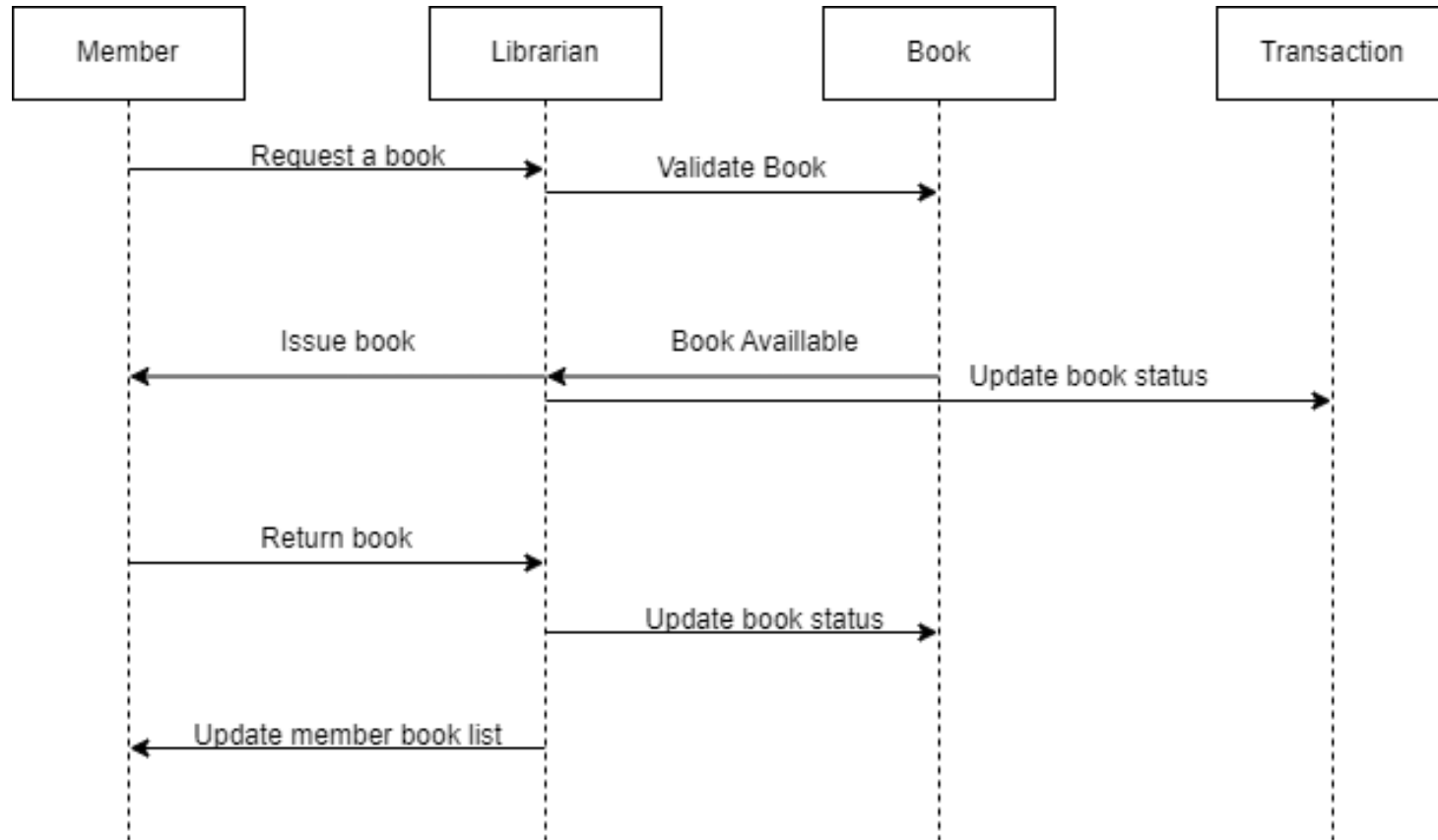
- The program is a library management system where librarians can add members, issue and return books, calculate fines for overdue books, and display borrowed books.
- It employs object-oriented principles with classes such as Person, Librarian, Member, and Book.

Design

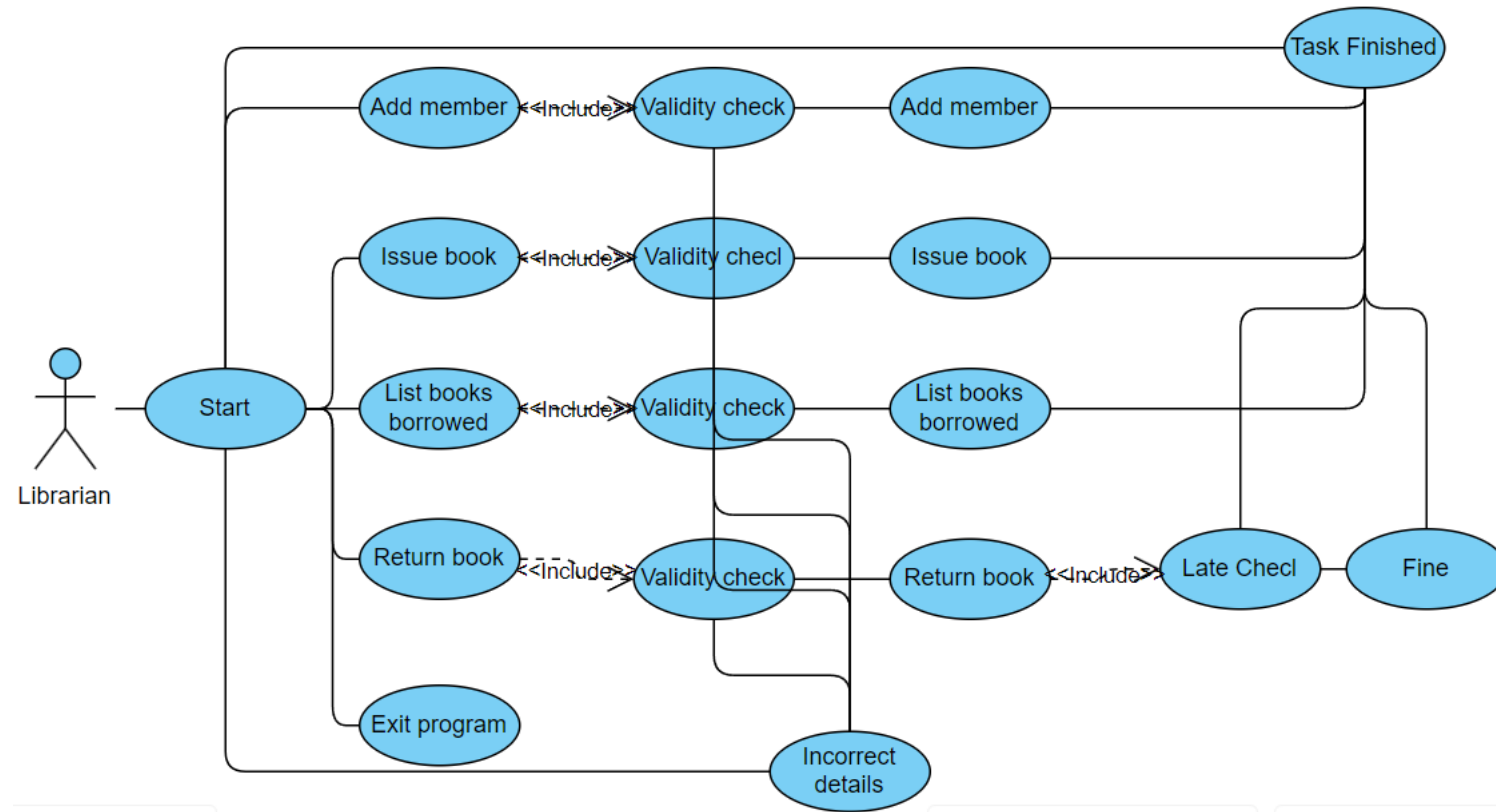
- Class Diagram



- Sequence Diagram



- Use Case Diagram



Implementation

Approach

- Create Class Files:
- Create separate header (.hpp) and implementation (.cpp) files for each class (**Person**, **Librarian**, **Member**, **Book**, **Date**).
- Define Class Attributes:
- In each class, define the attributes specified in the UML diagrams (**name**, **address**, **email**, **staffId**, **salary**, **memberId**, **booksLoaned**, etc.).
- Implement Constructors:
- Implement constructors in each class to initialize the attributes.
- Implement Setter and Getter Methods:
- Implement setter and getter methods for each attribute in the respective classes.
- Implement Member Functions:
- Implement member functions/methods as described in the UML diagrams for operations like adding members, issuing/returning books, calculating fines, etc.
- Handle Book Borrowing and Returning:
- Implement the borrowBook and returnBook methods in the Book class to handle the borrowing and returning of books by members.

Makefile

- **CXX** is the C++ compiler, and **CXXFLAGS** are the compiler flags, including C++ standard and warning options.
- **SRC_DIR** is the directory containing the source code, and **BUILD_DIR** is the directory where object files will be stored.
- **SRCS** is a list of all source files, and **OBJS** is the corresponding list of object files.
- **EXECUTABLE (library)** is the name of the final executable.
- The all target depends on the $\$(EXECUTABLE)$ target, which depends on the object files. It compiles the source code into the executable.
- The rule to build each object file specifies the compilation command.
- The clean target removes all object files and the executable.

Version Control

- Github is used for version control.
- All the required the **cpp** and **hpp** files are pushed to repository.
- **Makefile** is also pushed to repo. **Presentation** will also be published after creation.

Repository

github.com/M-Islam2004/MahinurIslam_M00912138_CW

main

1 Branch

0 Tags

Go to file

Add file

<> Code

About

mahi1232004

Ability to input date for book return

ecacaf2 · 13 hours ago

6 Commits

Book.hpp	Classes Implementation	2 days ago
Date.hpp	Ability to input date for book return	13 hours ago
Librarian.hpp	Ability to input date for book return	13 hours ago
Member.hpp	Classes Implementation	2 days ago
Person.hpp	Classes Implementation	2 days ago
README.md	first commit	2 days ago
catch.hpp	Classes Implementation	2 days ago
library_books.csv	Classes Implementation	2 days ago
main.cpp	Add functions and move readBooksFromCSV to librarian cl...	2 days ago
makefile	Classes Implementation	2 days ago
test.cpp	Addition of test cases and test.cpp general improvements	yesterday

About

No description, website, or topics provided.

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

C++ 99.9%

Makefile 0.1%

Commits

github.com/M-Islam2004/Mahinurlslam_M00912138_CW/commits/main/

M-Islam2004 / Mahinurlslam_M00912138_CW

Type to search

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

main

All users

All time

Commits on Jan 12, 2024

PDF of presentation slides

mahi1232004 committed 12 minutes ago

bea7ec2

<>

Commits on Jan 11, 2024

Ability to input date for book return

mahi1232004 committed 17 hours ago

ecacaf2

<>

Addition of test cases and test.cpp general improvements

mahi1232004 committed yesterday

ff3bdc4

<>

Commits on Jan 10, 2024

Ability to input date for book return

mahi1232004 committed 17 hours ago

ecacaf2

<>

Addition of test cases and test.cpp general improvements

mahi1232004 committed yesterday

ff3bdc4

<>

Commits on Jan 10, 2024

Add functions and move readBooksFromCSV to librarian class

mahi1232004 committed 2 days ago

082aba9

<>

Add tests.cpp

mahi1232004 committed 2 days ago

722d77f

<>

Commits on Jan 9, 2024

Classes Implementation

mahi1232004 committed 2 days ago

17f38c1

<>

first commit

mahi1232004 committed 2 days ago

846f2b8

<>

Testing Approach

- Catch2 has been used for test cases.
- Multiple test cases has been defined in the test.cpp

Application of Approach

- **Test Case 1: Reading CSV File:** Verify the functionality of reading book details from a CSV file.
- **Test Case 2: Creating New Member :** Test the creation of new members with distinct details.
- **Test Case 3: Issuing Book:** Confirm the correct issuance of books to a member.
- **Test Case 4: List Books :** Validate the librarian's ability to list books borrowed by a member.
- **Test Case 5: Salary :** Assess the functionality of setting and getting the librarian's salary.
- **Test Case 6: StaffID :** Ensure the librarian's staff ID can be correctly set and retrieved.

Conclusion

- The presented project is a comprehensive library management system implemented in C++.
- Key features include the ability to read book details from a CSV file, create and display member information, issue and return books, and manage librarian details.
- The **Makefile** streamlines the build process, and version control is handled solely using Git/GitHub, however, could be improved with more commits more early on in development.
- The testing approach employs **Catch2** test cases to systematically validate functionalities such as **CSV** file reading, member creation, book issuance, and librarian information management. There's room for improvement by adding extra test cases for input validation as well as implementing test which focus more on the change of data for example the tracking of the new members rather than final output
- The project showcases a well-organized and modular design, ensuring effective management and accessibility of library resources.