Coursework 1

**Introduction**

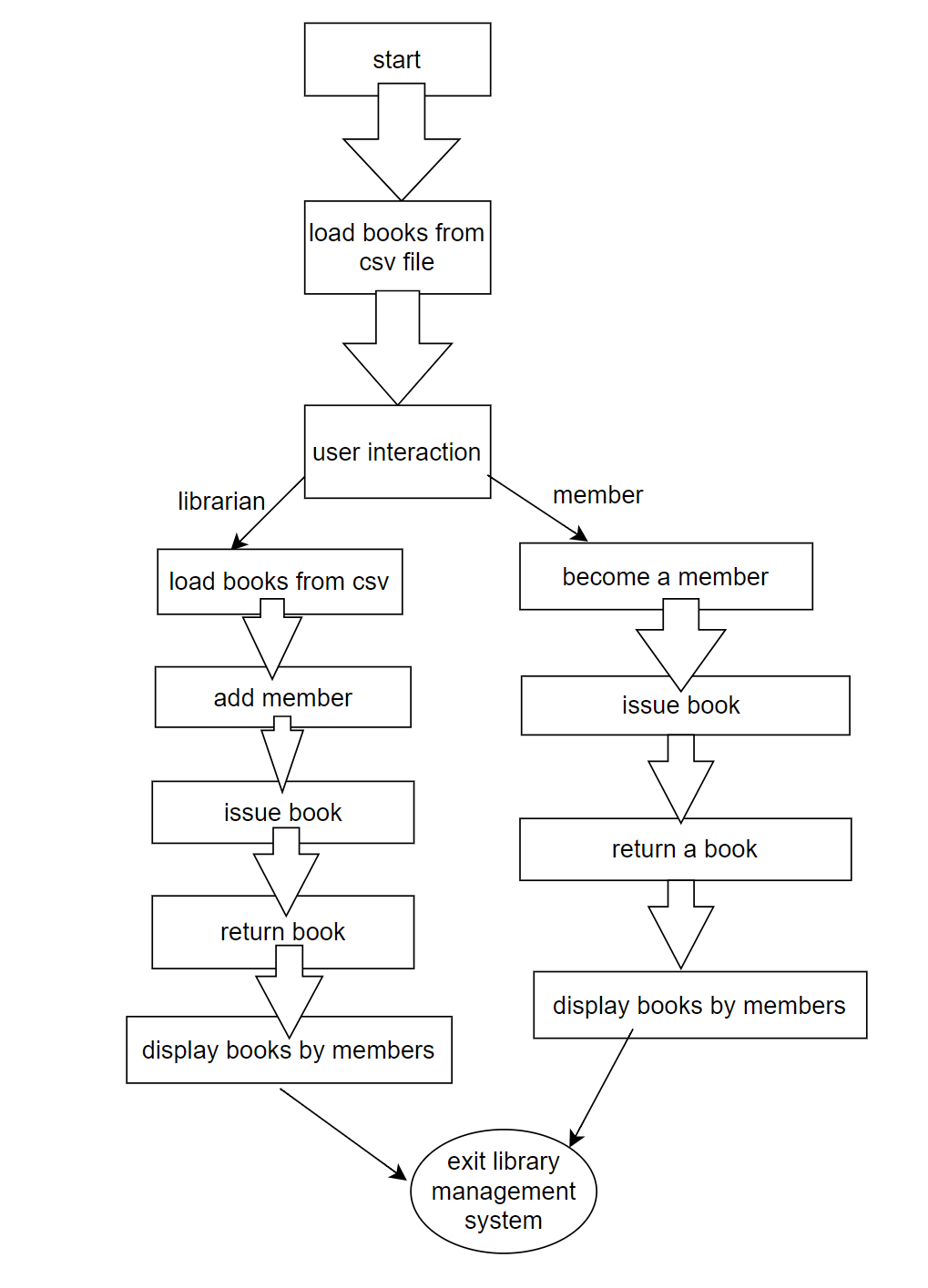
* The provided code represents a simple library management system implemented in C++.
* The system allows users to add members, issue books, return books, and display books borrowed by a particular member.
* It uses a structured approach with defined Book and Member structures and a Library class that provides functionalities for managing books and members.

**Design**

A diagram of a library system

Description automatically generated

* This is my use case diagram for my library management system.
* The focus of this use case diagram is on the library management system’s functionality and static structure.
* With the main advantage being it represent actors, use cases, and relationships.
* This use case diagrams focuses on representing the functionalities or features provided by the library management system from an external user's perspective.
* It’s a simple diagram showing how the user and librarian interact within the program.
* The user is given 5 options, create a id(add member),borrow a book(issue a book),return a book, display books the user is currently borrowing ,exit library.
* The Librarian return 5 different options including, issues an member ID, lets the user borrow book with a return date within 3 days, returns the users book and lastly shows the list of books the user is currently borrowing.



* This is my activity diagram for the library management system.
* The main advantage of the activity diagram it represent activities, actions, and control flows within the system.
* It shows how the roles of the librarian and user differ.
* The activity diagrams focus on representing the workflow or dynamic aspects of a system. Which illustrates the sequence of activities and actions within the library management system.

A diagram of a computer program

Description automatically generated with medium confidence

* This is the UMl diagram for the library management system.
* The UML (Unified Modelling Language) diagram is a visual representation of the library management system.
* Attributes are properties or data members of a class. They represent the characteristics or features of the class in this library management system
* Methods are functions or behaviours associated with a class. They represent actions that objects of the class can perform within the library management system
* For the member Class the attributes are member Id, name, email and address. The Methods I have used are borrowBook(), returnBook().
* Book Class: Attributes: book id
* For the Library Class the attributes are books (list of Book), members (list of Member). The Methods I have used are addMember(), issueBook(), returnBook(), displayBookByMember()

**Software Testing**

* I used Integration testing. After individual components were verified, the interactions between components.
* Prior to integration, each method—`addMember()`, `issueBook()`, `returnBook()`, and `displayBooksByMember()` underwent thorough individual testing to ensure both functionality and the absence of any bugs or errors. Only after confirming the reliability of each method were, they then were combined with the structures, classes, and the main function.
* However, the primary challenge with integration testing arises when combining methods with the main function, classes and structures often resulted in numerous bugs and errors in the code. Consequently, extensive debugging of my code and running numerous tests became necessary.

**Implementation of code**

* The code follows an object-oriented approach with structures (Book and Member) and a class (Library) to encapsulate related functionalities.
* The code begins by including necessary C++ libraries, such as <iostream>, <vector>, <string>, <ctime>, <algorithm>, <fstream>, and <sstream>.
* Two structures are defined: Book to represent book details, and Member to represent member details.
* The library class is created to manage books and members.
* Private Members :std::vector<Book> books: A vector to store information about books. std::vector<Member> members: A vector to store information about library members.
* Public Methods:
  + findBookById(int id): Finds a book by its ID and returns a pointer to it.
  + addMember(const std::string& name, const std::string& email, const std::string& address): Adds a new member to the library.
  + loadBooksFromCSV(const std::string& filename): Loads books from a CSV file into the library system.
  + issueBook(int bookId, int memberId): Issues a book to a member.
  + returnBook(int bookId, int memberId): Returns a borrowed book and calculates fines if applicable.
  + displayBooksByMember(int memberId): Displays books borrowed by a specific member.

The main function:

* Creates an instance of the library class.
* Opens and reads the content of the "library.csv" file.
* Displays the content of the CSV file to the console.
* Enters a loop to display a menu to the user until the user chooses to exit (shouldExit is set to true).
* The menu options include:
* Add Member: Takes member details and adds a new member to the library.
* Issue Book: Takes book and member IDs and issues a book to a member.
* Return Book: Takes book and member IDs, returns a book, and calculates fines if applicable.
* Display Books Borrowed by Member: Takes a member ID and displays the books borrowed by that member.
* Exit Library: Exits the program.
* The code checks for invalid input when reading integers from the user using std::cin. If the input is invalid, it clears the input buffer and prompts the user to enter a valid input.
* The code loads book information from a CSV file ("library.csv") using loadBooksFromCSV method.
* The code includes error messages for cases such as invalid book ID, invalid member ID, book already borrowed, and book not found in the member's list of borrowed books.
* Success messages are displayed for actions like adding a member, loading books from CSV, issuing a book, and returning a book.
* The program uses a do-while loop to continuously display the menu until the user chooses to exit.

**Implementation of Make File**

* The Makefile simplifies the build process by automating compilation and linking steps. I used variables like CXXFLAGS, to ensures consistent compilation settings across multiple source files.
* The make clean command is beneficial during development and debugging phases. It allows you to easily remove all generated files, ensuring that the next build is from scratch without any residual files
* I used the make command to build the project and used the make clean command to clean the project and remove all generated files.
* Overall the Makefile made it easier to manage my C++ project. Each section contributes to specifying the rules and dependencies required for a successful build.

**Version control and Implementation of Git**

* By utilizing Git commands, I successfully saved my source folder,catch2 and Makefile from Cygwin terminal to my GitHub repository, ensuring its secure storage.
* These are the following Git commands I used:
* git add .
* git commit .
* git push -u origin main
* below are screenshots off my GIT repository and commit messages

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Summary of work done and limitations**

* While the project aims to create a seamless library management system enabling users to effortlessly manage books and members by incorporating features like member addition, book issuance, and returns, the current code implementation requires further refinement to achieve this desired functionality.
* For example, the program is running well, the program is reading the excel csv file and printing it to the Cygwin terminal.
* However, when selecting option 2 or 3 to issue a book or return a book the program is not fetching the book id from the csv file. As a result when entering a book Id it prints an error message stating “invalid book id”.
* However, the rest of the program is working such as adding a member id and the books borrowed function.

**What I would do next time**

* In the future, I'll be diligent about committing my C++ code to GitHub regularly. The main advantage of utilizing Git is its capability to seamlessly track changes, ensuring that if errors emerge, I can revert to any prior version with ease.
* Initially, I stored previous versions of my code in separate files. Yet, upon discovering that GitHub offers a way to view and save past versions of my code, I understood that this extra step was redundant.
* I would also approach the way I start my code differently with more focus on functionality rather than design.