2024-2025 Spring Semester

CMP2005 / SEN2005 / CP2005 / MIS2005 / MIS3005

Object-Oriented Programming

Library Management System

Objective:

Design and implement a simple **Library Management System** using Object-Oriented Programming principles in C++. The system allows users to register, borrow, and return books from a catalog of books and magazines. The catalog and user data will be managed in memory, and transaction logs will be written to a text file.

Key Concepts Covered:

- Classes and Inheritance
- **Polymorphism** with virtual functions
- Exception Handling
- Operator Overloading
- Using maps and vectors
- File operations for logging
- Project structure with multiple files and a Makefile

Learning Outcomes:

	Covered?	
Apply pointers and dynamic memory allocation to manage memory efficiently in C++ programs.		
Utilize object-oriented programming (OOP) principles, including inheritance, polymorphism,		
and operator overloading, to design modular and reusable software.		
Implement structured data types, templates, and exceptions to create robust and scalable	YES	
applications.		
Perform advanced file operations for data storage, retrieval, and manipulation.		
Use the Standard Template Library (STL), including vectors and maps, to streamline data		
management.		
Develop a comprehensive C++ project , integrating learned concepts and presenting it effectively.		

Development Guidance

Step 1: Design Base and Derived Classes

- **LibraryItem** (Abstract class): Common attributes (title, author, itemID) for all items.
- Book and Magazine will inherit from LibraryItem.
 - Book supports borrowing and returning.
 - o Magazine does **not** support borrowing.

Step 2: Implement the User Class

- Stores a userID, name, and list of borrowed item IDs (as std::vector).
- Should allow borrowing and returning items.
- Include proper checks and exceptions.

Step 3: Implement LibrarySystem Class

- Manages:
 - All library items: std::map<std::string, LibraryItem*>
 - o Registered users: std::map<std::string, User>
- Allows:
 - Adding books and magazines
 - Adding users
 - o Borrowing and returning books
 - o Displaying all items/users
 - o Logging transactions to a file

Step 4: Add Custom Exceptions

- Define LibraryException class for handling application-specific errors.
- Throw exceptions in cases like:
 - o Borrowing an already borrowed book
 - o Returning an unborrowed book
 - o Accessing nonexistent users or items

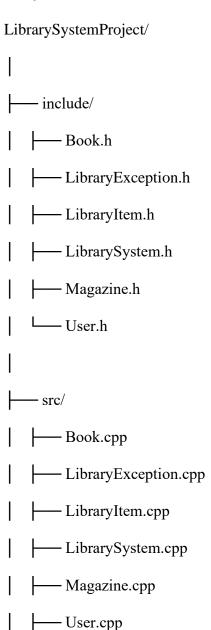
Step 5: Build a CLI Menu in main.cpp

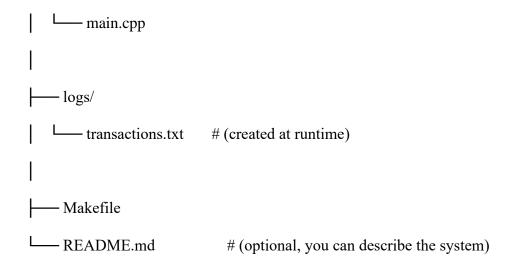
Create an interactive menu:

- 1. Add Book
- 2. Add Magazine
- 3. Add User
- 4. Borrow Book

- 5. Return Book
- 6. View All Items
- 7. View All Users
- 8. Exit

Project Folder Structure & Makefile





Make commands:

Command What it does

make Compiles the project

make run Compiles and runs the program make clean Deletes compiled files and binary

Submission Guideline

• Students must submit a **compressed folder** (**ZIP**) named:

YourFullName_StudentID_LibraryProject.zip

Folder Contents:

Requirements:

- Use **OOP concepts** correctly: inheritance, polymorphism, operator overloading.
- Project **must compile and run** using the provided Makefile with make run.
- Program should handle **exceptional cases** gracefully.
- Maintain clean and consistent formatting.
- Use meaningful class, variable, and function names.
- Code must be written by the student. Plagiarism results in a zero.

Marking Breakdown (Total: 100 points)

Section	Points	ints Criteria	
1. Code Organization	10	Uses proper file structure and Makefile. Includes all necessary files.	
2. Class Design	15	Proper use of classes, inheritance, and virtual functions. Clean design.	
3. Polymorphism	10	Virtual methods used properly, base class pointers used for derived types.	
4. Operator Overloading	5	Stream operator (<<) or others used correctly and meaningfully.	
5. Exception Handling	10	Custom exceptions created and used effectively to handle invalid cases.	
6. File Operations (Logs)	10	Borrow/return actions are logged to a file in correct format.	
7. Map/Vector Usage	10	Manages data structures effectively for items and users.	
8. User Menu (main.cpp)	10	Functional and user-friendly CLI menu system implemented.	
9. Code Quality and Style	10	Proper indentation, comments, naming conventions, and modularization.	
10. Program Functionality	10	Core features (add, borrow, return, display) work as expected.	
11. Bonus Features (optional)) +5	E.g., search function, input validation improvements, categories, etc.	

Notes:

- Late submissions may incur penalties.
- Projects that fail to compile and run will get at most 30 points unless errors are minor.
- Your implementation will be manually tested by the instructor.

transactions.txt

```
[2025-04-19 14:03:21] User: U1001 (Alice Smith) borrowed Book: B001 - "The Great Gatsby"
[2025-04-19 14:05:12] User: U1002 (Bob Lee) borrowed Book: B002 - "1984"
[2025-04-19 14:07:03] User: U1001 (Alice Smith) returned Book: B001 - "The Great Gatsby"
[2025-04-19 14:10:45] User: U1003 (Charlie Kim) attempted to borrow Book: B002 - "1984" - FAILED (Already borrowed)
```

[2025-04-19 14:12:09] User: U1004 (Dana Liu) viewed all available items

[2025-04-19 14:15:40] User: U1002 (Bob Lee) returned Book: B002 - "1984"

[2025-04-19 14:18:01] User: U1003 (Charlie Kim) borrowed Book: B002 - "1984"

Log Generation Notes

- The timestamp format is: [YYYY-MM-DD HH:MM:SS]
- Each entry clearly identifies:
 - o **Action type** (borrow, return, failed attempt, etc.)
 - o User ID and name
 - o Book ID and title
- Optional actions like **viewing catalog** can also be logged.

You would implement this inside your LibrarySystem::logTransaction(...) method, using std::ofstream in append mode and std::chrono for time.