

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

## 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.
  - 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\*>
  - Registered users: std::map<std::string, User>
- Allows:
  - Adding books and magazines
  - Adding users
  - Borrowing and returning books
  - Displaying all items/users
  - Logging transactions to a file

### Step 4: Add Custom Exceptions

- Define LibraryException class for handling application-specific errors.
- Throw exceptions in cases like:
  - Borrowing an already borrowed book
  - Returning an unborrowed book
  - 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**

LibrarySystemProject/

```
|
|
|— include/
|   |— Book.h
|   |— LibraryException.h
|   |— LibraryItem.h
|   |— LibrarySystem.h
|   |— Magazine.h
|   |— User.h
|
|— src/
|   |— Book.cpp
|   |— LibraryException.cpp
|   |— LibraryItem.cpp
|   |— LibrarySystem.cpp
|   |— Magazine.cpp
|   |— User.cpp
```

```
|   └── main.cpp
|
|── logs/
|   └── transactions.txt    # (created at runtime)
|
|── Makefile
└── README.md              # (optional, you can describe the system)
```

### 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:

YourFullName\_StudentID\_LibraryProject/

```
|── include/          # All header files (.h)
|── src/              # All implementation files (.cpp)
|── logs/             # (Empty or containing transaction logs if available)
|── Makefile          # Compiles and runs the project
|── README.md         # (Optional) Description or notes
└── main.cpp          # Menu and CLI
```

## 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	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 ( <i>optional</i> )	+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:
  - **Action type** (borrow, return, failed attempt, etc.)
  - **User ID and name**
  - **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.