The Enchanted Library

An Intelligent and Secure Book Management System

Problem Statement:

For centuries, the Grand Library of Eldoria has housed the world's most ancient manuscripts, rare books, and mystical scrolls. It was maintained by an intelligent cataloging and security system, which managed book lending, visitor authentication, section access, and archival preservation.
However, after the Chief Librarian mysteriously disappeared, the system began failing:
Books vanished from records, even though they were still on shelves.
Visitors borrowed books but were never logged, leading to loss of valuable texts.
Forbidden sections became accessible to unauthorized users, risking damage to priceless knowledge.
Restoration logs became corrupt, preventing tracking of book conditions.
The Council of Scholars has now tasked your team of OOP software architects to restore and upgrade the library's management system using Object-Oriented Design and Design Patterns, ensuring a secure, scalable, and efficient book cataloging and lending system.

Project Requirements

You will design an OOP-based Library Management System in C++/Java/Python
→ integrating security → user access control → inventory tracking (while ensuring maintainability and efficiency)

1. Core OOP Concepts

Encapsulation: Secure book records and visitor details, restricting direct modifications.
Inheritance: Design different types of books (**AncientScript**, **RareBook**, **GeneralBook**) from a base Book class.
Polymorphism: Implement different **lending policies** (Short-Term, Long-Term, Restricted Access).
Abstraction: Provide simplified interfaces for librarians and visitors to interact with the system.

2. Implementation of Creational Design Patterns

Factory Pattern:
o Dynamically create **book objects** and **user roles** (Librarian, Scholar, Guest).
Singleton Pattern:
o Ensure a single central catalog system to manage book records.

Builder Pattern:
o Allow customized book entries, including metadata like preservation requirements, digital access permissions, and lending restrictions.

3. Implementation of Structural Design Patterns

Facade Pattern:
o Provide a single library management dashboard for book tracking and user management.
Adapter Pattern:
o Integrate legacy handwritten records with the new digital system.
Decorator Pattern:
o Add features dynamically, like automatic overdue reminders or restricted section permissions.

4. Implementation of Behavioral Design Patterns

Observer Pattern:
o Notify librarians when books are overdue or need restoration.
Strategy Pattern:
o Implement different book lending rules (Academic Borrowing, Public Lending, Restricted Reading Room).
Command Pattern:
o Allow undoable actions, like returning an incorrectly borrowed book.
State Pattern:
o Track book status (Available, Borrowed, Reserved, Restoration Needed).

5. Additional Features & Challenges

Role-Based Access Control
o Implement permission levels (e.g., Scholars can access restricted books, General Visitors cannot).
Automated Late Fee Calculation
o Dynamically calculate late fees based on book type and borrowing period.
Archival & Preservation Module
o Automatically flag books for restoration based on condition reports.
Smart Book Recommendations
o Suggest books based on visitor reading history and current research topics.

6. Optional Graphical Dashboard (Bonus Feature)

Develop a GUI using Tkinter (Python), JavaFX (Java), or Qt (C++).
Display book availability, borrowing status, and overdue records.
Allow librarians to manage lending, book restoration, and visitor permissions.