Project Case: Library Management System

Background

A modern library is not just a collection of books but a system of organized information management. As the library expands, manually keeping track of books, borrowers, and transactions becomes inefficient and error-prone. To solve this, you are tasked with designing and implementing a **Library Management System (LMS)** that handles all key aspects of book and member management.

Project Objectives

- Develop a modular Python application that automates book handling, member registration, borrowing/returning, and reporting.
- Implement business rules and constraints that reflect real-world library operations.
- Ensure scalability and maintainability by using structured modules and persistent data storage.

Functional Requirements

1. **Book Management**
    - Add new books with attributes (ID/ISBN, title, author, genre, publication year, age restriction, and available copies).
    - Update details when new copies are received.
    - Handle duplicate book entries by incrementing stock rather than creating a duplicate.
    - Search books by title, author, or genre.
    - Mark books as unavailable if all copies are borrowed.
2. **Member Management**
    - Register members with attributes (member ID, name, age).

    - Prevent underaged users from accessing 18+ rated materials.
    - Enforce borrowing limits (e.g., a member cannot borrow another book until the current one is returned).

3. **Borrowing & Returning**
    - Borrow a book if:
        - The book exists and is available.
        - The member is eligible (meets age restrictions, does not have overdue books).
    - Return a book and update availability.
    - Handle overdue returns by applying fines.
    - Handle invalid returns (book not borrowed or user not recognized).
4. **Constraints and Rules**
    - A member cannot borrow a book if they already have one borrowed.
    - A member cannot borrow more than `N` books at once (configurable).
    - Age restrictions must be strictly enforced (e.g., under 18 cannot borrow 18+ books).
5. **Reporting & Logging**
    - Generate reports of borrowed, returned, overdue, and unavailable books.
    - Log all transactions (borrowing, returning, fines).

Technical Requirements

- **Storage**: JSON or CSV files for persistence (e.g., `books.json`, `members.json`).

Deliverables

- **Codebase**: Structured into modules as outlined (Try as much as possible to factor in all of the Python concepts that how have learned so far, but only when and where necessary.
- **Documentation**: Explaining system design, functions, and how to run the project.
- **Demonstration**: A test scenario showcasing:
    - Adding books and members.
    - Borrowing and returning books (including error cases like unavailable books or age restrictions).
    - Generating a report of transactions.

Evaluation Criteria

- **Correctness**: Does the system handle all specified requirements and edge cases?
- **Modularity**: Is the code organized into logical, reusable components?
- **Scalability**: Can new features (e.g., digital media, reservations) be added easily?
- **Robustness**: Does the system gracefully handle errors (invalid input, missing books, etc.)?
- **Documentation**: Is the system well-documented and easy to use?