Mini Library Management System – Design Rationale

1. Introduction

The Mini Library Management System was designed to simulate essential library operations in a simple, modular Python program. It provides CRUD (Create, Read, Update, Delete) functionality for both books and members, as well as borrowing and returning book operations.

2. Design Objectives

• Keep the code modular and easy to maintain.
• Use Python dictionaries and lists for lightweight in-memory data storage.
• Provide clear error handling and validations for user operations.
• Ensure scalability for future features such as data persistence.

3. System Architecture

The system follows a modular architecture with three main Python modules:
• operations.py – Core logic and CRUD operations.
• Demo.py – Demonstration of system behavior.
• tests.py – Automated validation tests.

4. Data Structures

Books are stored in a dictionary with ISBN keys and info dictionaries as values. Members are stored in a list of dictionaries. Genres are predefined in a tuple for validation.

5. Functional Design

Each operation is implemented as a function that updates shared data. Borrowing and returning enforce rules: a member can borrow up to 3 books, and deletions are restricted when items are borrowed.

6. Error Handling

User actions return readable messages instead of exceptions for a simpler user experience.

7. Testing and Validation

tests.py uses assert statements to confirm correctness of all operations. A successful test run prints: '■ All tests passed successfully!'

8. Future Improvements

• Add persistent storage (JSON or database).
• Add GUI or web interface.
• Implement fine/penalty system for overdue books.
• Add admin roles.

9. Conclusion

This project demonstrates a clean, modular, and extendable Python design for small-scale management systems. It’s ideal for educational use and further development into full applications.