Library Management System: A SOLID Approach

Introduction

This report presents the design and implementation of a **Library Management System**, developed in **C** while adhering to **SOLID principles**. The system is structured to efficiently manage core library functions, including adding books, searching for books, lending and returning books, and generating reports. The design emphasizes **maintainability, scalability, and future extensibility**, ensuring that the system remains adaptable to evolving needs.

System Design

UML Class Diagram

The UML class diagram provides a visual representation of the system's structure, depicting the relationships between various components. The system is organized into three primary modules:

- **Book Management**
- **Reader Management**
- **Report Generation Service**

Each module is carefully designed to follow **SOLID principles**, promoting a **clean, maintainable, and modular architecture**.

Class and Interface Definitions

1. Models

- **IBook**: Defines the core properties and methods for a book.
- **Book**: Implements `IBook`, representing a physical book in the library.
- **Reader**: Represents a library member, uniquely identified with a list of borrowed books.

2. Repositories

- **IBookRepository**: Interface for managing book storage operations.
- **IReaderRepository**: Interface for handling reader data storage.
- **InMemoryBookRepository**: An in-memory implementation of `IBookRepository`.
- **InMemoryReaderRepository**: An in-memory implementation of `IReaderRepository`.

3. Services

- **ILibraryService**: Defines core library operations such as adding, lending, and returning books.
- **LibraryService**: Implements `ILibraryService`, facilitating essential library functions.
- **IReportService**: Defines methods for generating reports.
- **ReportService**: Implements `IReportService`, providing detailed reports on borrowed books.

Implementation

Core Functionalities

1. **Adding New Books**
   - Administrators can add books with details such as **Title, Author, Category, and Quantity**.
   - Each book is assigned a **unique identifier** for efficient tracking.
1. **Searching for Books**
   - Users can search for books using **Title or Category**.
   - The search is **case-insensitive**, returning all relevant matches.
1. **Lending Books**
   - Readers can borrow books if they are available.
   - Each reader can borrow a maximum of **three books** at a time.
   - The system updates the **book quantity** upon lending.
1. **Returning Books**
   - When a reader returns a book, the system **updates inventory records**, making the book available for others.
1. **Generating Reports**
   - The system produces **comprehensive reports**, listing readers and the books they have borrowed.
   - These reports provide valuable insights into **library activity and resource utilization**.

Sample Data

To facilitate testing and demonstration, the system includes a **preloaded dataset** with a variety of books and registered readers, enabling thorough validation of its functionalities.

Application of SOLID Principles

1. **Single Responsibility Principle (SRP)**
   - Each class is **dedicated to a single responsibility**, making the system **modular and easy to maintain**.
   - Example: The `Book` class handles **book properties**, while `LibraryService` manages **operations**.
1. **Open/Closed Principle (OCP)**

- The system is **open for extension but closed for modification**.
- New book types can be added **without altering existing code**, simply by implementing the `IBook` interface.

1. **Liskov Substitution Principle (LSP)**

- Any class implementing `IBook` can be substituted in place of `IBook` **without affecting functionality**.
- This ensures that different book types remain **interchangeable and compatible**.

1. **Interface Segregation Principle (ISP)**

- Interfaces are designed to be **specific to individual functionalities**, preventing unnecessary dependencies.

1. **Dependency Inversion Principle (DIP)**

- High-level modules depend on **abstractions rather than concrete implementations**, improving **scalability and flexibility**.

Extensibility and Future Enhancements

The system is designed to be **easily extendable**, allowing for future improvements such as:

1. **Managing eBooks**

- Introducing an `EBook` class that implements `IBook`, enabling the system to manage **digital books** alongside physical ones.

1. **Book Reservations**

- Implementing an `IReservationService` interface to allow users to **reserve books in advance**, ensuring availability.

1. **Integration with External Systems**

- Enhancing functionality by integrating with **external databases or library management platforms** for a more comprehensive solution.

Conclusion

The **Library Management System** is a well-structured, **maintainable**, and **scalable** solution that adheres to **SOLID principles**. By carefully implementing a modular design, the system **efficiently manages core library operations** while providing a strong foundation for future expansion.

This report highlights how **SOLID principles** can be successfully applied to build **flexible, efficient, and reliable software**, ensuring that the system **evolves seamlessly with changing requirements**.