**Design Document for Library Management System**

**1. Introduction**

This document describes the design of the Library Management System. The system allows members to borrow and return books, and librarians to manage the book catalog and member accounts. It incorporates several design patterns to achieve a modular, flexible, and maintainable design.

**2. Component Diagram**

graph LR

UI["User Interface"] --> LibraryFacade["Library Facade"]

LibraryFacade --> BookService["Book Service"]

LibraryFacade --> MemberService["Member Service"]

LibraryFacade --> LoanService["Loan Service"]

LibraryFacade --> BookManagementSystem["Book Management System"]

BookManagementSystem --> BookService

LoanService --> NotificationService["Notification Service"]

NotificationService --> SMSAdapter["SMS Adapter"]

SMSAdapter --> ExternalSMSService["External SMS Service"]

BookService --> Book["Book"]

MemberService --> Member["Member"]

LoanService --> Loan["Loan"]

DailyTaskScheduler["Daily Task Scheduler"] --> LoanService

**3. Class Diagram (Simplified Description)**

* **User (Interface):** Defines common methods for Member and Librarian.
* **Member (Class):** Implements User. Represents a library member. Attributes: name, ID, contact details. Methods: getters, setters.
* **Librarian (Class):** Implements User. Represents a librarian. Attributes: name, ID, contact details. Methods: getters, setters.
* **Book (Class):** Represents a book. Attributes: ISBN, title, availability, reservations. Methods: getters, setters, reserve(), setAvailable(), isReserved(). Implements Reservable.
* **BookService (Class):** Manages the book catalog. Uses BookRegistry for storage. Methods: addBook(), updateBook(), removeBook(), findBookByISBN(), searchBooksByTitle(), getAllBooks().
* **MemberService (Class):** Manages members. Uses UserRegistry for storage. Methods: registerUser(), updateUser(), deleteUser(), findUserById(), getAllUsers().
* **Loan (Class):** Represents a loan. Attributes: member, book, dueDate, returned. Methods: getters, markAsReturned().
* **LoanService (Class):** Manages loans. Implements DueDateObservable. Attributes: notificationService (injected). Methods: borrowBook(), returnBook(), checkForDueDates(), getActiveLoans(), addDueDateObserver(), removeDueDateObserver(), notifyDueSoonObservers(), notifyOverdueObservers().
* **LibraryFacade (Class):** Provides a simplified interface to the library system. Methods: addBook(), updateBook(), removeBook(), searchBooksByTitle(), getAllBooks(), registerMember(), findMemberById(), getAllMembers(), borrowBook(), returnBook(), getActiveLoans(), checkOverdueBooks(), reserveBook(), undoLastBookOperation().
* **NotificationService (Interface):** Defines the interface for sending notifications. Method: sendNotification(User user, String message).
* **SMSAdapter (Class):** Implements NotificationService. Adapts to the ExternalSMSService.
* **ExternalSMSService (Interface):** Defines the interface for an external SMS service. Method: sendSMS(String phoneNumber, String message).
* **BookIterator (Interface):** Defines the interface for iterating over books.
* **BookIteratorImpl (Class):** Implements BookIterator.
* **BookManagementSystem (Class):** Manages book-related commands.
* **Command (Interface):** Defines the interface for commands.
* **AddBookCommand, UpdateBookCommand, DeleteBookCommand (Classes):** Implement the Command interface.
* **UI (Class):** Handles user interface interactions. Methods: getUserDetails(), getBookDetailsFromUser(), getUpdatedBookDetailsFromUser(), getISBNFromUser(), getTitleFromUser(), viewLibrarianOptions(), viewMemberOptions().
* **DailyTaskScheduler (Class):** Responsible for scheduling the daily execution of the LoanService's checkForDueDates() method. Uses ScheduledExecutorService (or Quartz, etc.).

**4. Design Patterns**

* **Factory Method:** UserFactory creates Member or Librarian objects based on the provided role.
* **Singleton:** UserRegistry and BookRegistry are likely implemented as Singletons to ensure only one instance exists.
* **Adapter:** SMSAdapter adapts the NotificationService interface to the external SMS service's interface.
* **Facade:** LibraryFacade simplifies the interface to the library system, hiding the complexity of the underlying components.
* **Command:** The Command pattern is used for book management operations (add, update, delete). This allows for undo/redo functionality. BookManagementSystem acts as the invoker.
* **Iterator:** BookIterator and BookIteratorImpl provide a way to iterate over the books in the catalog.

**6. Key Interactions**

* **Borrowing a Book:** The LibraryFacade receives the borrow request. It interacts with the MemberService to verify the member, the BookService to check book availability, and the LoanService to create a loan record.
* **Returning a Book:** The LibraryFacade interacts with the LoanService to mark the book as returned and with the BookService to update the book's availability. The Book object, when its availability is set to true, notifies any registered Member observers (who have reserved it).
* **Reserving a Book:** The LibraryFacade interacts with the Book class to reserve a book.
* **Adding a Book:** The LibraryFacade uses the BookManagementSystem to execute an AddBookCommand, which interacts with the BookService to add the book. The other book commands (UpdateBookCommand, DeleteBookCommand) work similarly.
* **Availability Notification:** When a book's availability changes to true, the Book class's BookAvailability instance notifies all registered Member observers. Each Member then uses the injected NotificationService (via the SMSAdapter) to send an SMS message.
* **Due Date and Overdue Notifications:** The LoanService periodically checks for due and overdue loans. It then notifies the relevant Member observers, who in turn send SMS messages.
* **Reserving a Book:** The LibraryFacade receives the reserve request. It interacts with the Book class's reserve() to reserve the book for requested member.
* **Daily Due Date Check:** The DailyTaskScheduler (using ScheduledExecutorService or Quartz) schedules and triggers the LoanService's checkForDueDates() method to run daily. The checkForDueDates() method then checks for due and overdue books and notifies the relevant Member observers.