Project: Library Management System

1. Introduction

The purpose of this document is to provide a detailed Low-Level Design (LLD) for the Library Management System, which allows users to manage book collections, member registrations, borrowing and returning of books, and overdue tracking. The system supports both an Angular or React frontend and a REST API-based backend. The design is compatible with Java (Spring Boot) and .NET (ASP.NET Core) frameworks.

2. Module Overview

The system is composed of the following modules:

- **Book Management**: Manages the book collection, including adding, updating, and removing books.
- **Member Management**: Handles member registration, profile updates, and membership status.
- **Book Borrowing and Return**: Manages the borrowing and return processes for members.
- **Overdue and Fines**: Tracks overdue books and applies fines for late returns.
- **Notifications and Alerts**: Sends notifications and alerts to members regarding due dates, overdue books, and fines.

3. Architecture Overview

3.1 Architectural Style

- **Frontend**: Developed using Angular or React
- **Backend**: RESTful API architecture with Spring Boot/ASP.NET Core
- **Database**: Relational Database (MySQL/PostgreSQL/SQL Server)

3.2 Component Interaction

- Frontend communicates with the backend via HTTP/HTTPS protocols.
- The backend processes requests and manages CRUD operations with the database.
- The frontend updates dynamically based on the data received from the backend.

4. Module-Wise Design

4.1 Book Management Module

- **Features**:
  - Add, update, and remove books from the system.
  - Search and view books by title, author, genre, etc.
- **Data Flow**:
  - Admin adds, updates, or deletes book information via the frontend.

        - Requests are processed by the backend, and the database is updated.
        - The frontend receives the updated data and displays it accordingly.
    - **Entities**:
        - **Book**: BookID, Title, Author, Genre, ISBN, YearPublished, AvailableCopies.

4.2 Member Management Module

- **Features**:
    - Register new members, update profiles, and manage membership status.
- **Data Flow**:
    - User (or admin) registers and updates member details through the frontend.
    - The backend processes the data and interacts with the database.
    - The frontend renders member-related data.
- **Entities**:
    - **Member**: MemberID, Name, Email, Phone, Address, MembershipStatus.

4.3 Book Borrowing and Return Module

- **Features**:
    - Borrow and return books.
    - Track book availability and borrowing limits.
- **Data Flow**:
    - Member requests to borrow or return a book via the frontend.
    - Backend validates the request (e.g., book availability, borrowing limits) and updates the database.
    - The frontend reflects the updated book status.
- **Entities**:
    - **BorrowingTransaction**: TransactionID, BookID, MemberID, BorrowDate, ReturnDate, Status (Borrowed, Returned).

4.4 Overdue and Fines Module

- **Features**:
    - Track overdue books and calculate fines based on overdue days.
    - Allow members to view and pay fines.
- **Data Flow**:
    - The system automatically checks for overdue books and calculates fines.
    - Notifications about overdue books and fines are sent to members.
    - Members can view fines and pay via the frontend.
- **Entities**:
    - **Fine**: FineID, MemberID, Amount, Status (Paid, Pending), TransactionDate.

4.5 Notifications and Alerts Module

- **Features**:
    - Send notifications to members for overdue books, approaching due dates, and fines.
- **Data Flow**:
    - The backend triggers notifications based on system events (e.g., due date nearing, overdue book).

        - Notifications are sent via email/SMS or displayed in the frontend.
    - **Entities**:
        - **Notification**: NotificationID, MemberID, Message, DateSent.

5. Deployment Strategy

5.1 Local Deployment

- The system will be deployed in a local development environment to facilitate testing and development.

5.1.1 Frontend Deployment

- Local development servers like ng serve (for Angular) or npm start (for React) will be used.

5.1.2 Backend Deployment

- The backend will be deployed using Spring Boot or ASP.NET Core on a local server for testing.

5.1.3 Database Setup

- A local database instance (e.g., MySQL/PostgreSQL/SQL Server) will be set up for development and testing.

6. Database Design

6.1 Tables and Relationships

6.1.1 Book

- **Primary Key**: BookID
- **Foreign Keys**: None

6.1.2 Member

- **Primary Key**: MemberID
- **Foreign Key**: None

6.1.3 BorrowingTransaction

- **Primary Key**: TransactionID
- **Foreign Keys**: BookID, MemberID (Links borrowed books to members)

6.1.4 Fine

- **Primary Key**: FineID
- **Foreign Key**: MemberID (Links fine to member)

6.1.5 Notification

- **Primary Key**: NotificationID
- **Foreign Key**: MemberID (Links notifications to members)

7. User Interface Design

7.1 Wireframes

- **Book Management Console**: Admin interface for managing the library book collection.
- **Member Registration and Profile Interface**: Page for new members to register and existing members to update their profiles.
- **Borrowing and Return Interface**: Allows members to borrow and return books, and track borrowing history.
- **Overdue and Fines Page**: Displays overdue books and associated fines for members.
- **Notification Panel**: Displays alerts related to due dates and fines.

8. Non-Functional Requirements

8.1 Performance

- The system should handle up to 100 concurrent users for basic testing in the local environment.

8.2 Scalability

- Designed to scale and accommodate more users, especially if deployed in a production environment.

8.3 Security

- User data must be encrypted, and access controls should be implemented to ensure privacy and security.

8.4 Usability

- The system should be intuitive and user-friendly, with a responsive design for various screen sizes.

9. Assumptions and Constraints

9.1 Assumptions

- The system will run only in the local environment during the development phase.
- Developers will use standard IDEs for development and debugging.

9.2 Constraints

- No cloud deployment is planned at this stage.
- The system will initially support basic features and scale gradually based on further requirements.

This LLD document for a **Library Management System** provides a structured overview of the components, data flow, database design, and deployment strategy for the system, ensuring compatibility with both Java (Spring Boot) and .NET (ASP.NET Core) frameworks.