



Case Study: Library Management System (LMS)

1. Overview

The **Library Management System (LMS)** automates library operations such as **catalog management, member registration, book lending/returns, fine calculation, and reporting**. It provides role-based access for **Admin, Librarian, and Members**, ensuring secure access with **JWT authentication** and offering a **responsive React UI** integrated with **Spring Boot REST APIs**.

2. Scope

- **Admin:**
 - Add/Remove/Update books
 - Manage members (add/update/remove)
 - Track issued/returned books
 - Generate reports (most borrowed books, overdue reports)
- **Librarian:**
 - Issue/Return books
 - Manage fines & payments
 - Assist members with catalog
- **Member:**
 - Search catalog (filters: title, author, category)
 - Borrow & return books
 - Reserve unavailable books
 - View transaction history

3. Tech Stack

- **Backend:** Spring Boot, Spring Security + JWT, Hibernate/JPA, Swagger
- **Frontend:** React (Hooks, Axios), Bootstrap for UI styling
- **Database:** MySQL

- **Middleware/Integration:** Node.js (for notification & cron jobs like due-date reminders)
- **API Documentation:** Swagger UI

4. System Architecture

- **React Frontend** → Axios → **Spring Boot REST API** → MySQL DB
- **JWT Security Layer:**
 - Role-based authentication (Admin, Librarian, Member)

5. Key Features with API Integration

◆ Authentication & Security

- **Login API:** POST /auth/login
- **Register API:** POST /auth/register
- **JWT Security:** Token-based authorization for all requests.

◆ Book Management

- **Add Book (Admin):** POST /api/books
- **Get All Books:** GET /api/books
- **Search Book by Title/Author:** GET /api/books/search?title=Java
- **Remove Book:** DELETE /api/books/{id}

◆ Member Management

- **Add Member:** POST /api/members
- **Get All Members:** GET /api/members
- **Update Member:** PUT /api/members/{id}
- **Delete Member:** DELETE /api/members/{id}

◆ Lending & Returns

- **Issue Book:** POST /api/lending/issue
- **Return Book:** POST /api/lending/return
- **Track Due Date:** GET /api/lending/due/{memberId}

◆ Fine Management

- **Calculate Fine:** GET /api/fines/calculate/{memberId}
- **Pay Fine:** POST /api/fines/pay

◆ Reports

- **Most Borrowed Books:** GET /api/reports/most-borrowed
- **Overdue Books:** GET /api/reports/overdue

6. Swagger Integration

- Spring Boot integrates **Swagger** (springdoc-openapi-ui)
- Access APIs at: `http://localhost:8080/swagger-ui.html`
- Provides documentation for all LMS endpoints with request/response samples

7. Frontend (React + Bootstrap)

- **Login & Register Page:** JWT authentication stored in localStorage
- **Dashboard:** Role-based UI (Admin, Librarian, Member)
- **Book Catalog:** Search & filter books
- **Issue/Return Page:** Librarian module
- **Reports Page:** Charts (using Recharts library)

8. Sample Folder Structure

Backend (Spring Boot)

```

lms-backend/
├── src/main/java/com/lms
│   ├── controller    # REST Controllers
│   ├── service        # Business logic
│   ├── repository    # JPA Repositories
│   ├── model          # Entities (Book, Member, Loan, Fine)
│   ├── security       # JWT Config
│   └── exception      # Custom Exceptions
└── src/main/resources
    ├── application.properties
    └── data.sql

```



Frontend (React + Bootstrap)

```

lms-frontend/
├── src/
│   ├── components/
│   │   ├── Login.js
│   │   ├── Dashboard.js
│   │   ├── BookCatalog.js
│   │   ├── IssueReturn.js
│   │   └── Reports.js
│   ├── services/ (Axios API calls)
│   └── App.js

```

9. Security Implementation (Spring Boot + JWT)

- **Login generates JWT** → stored in React localStorage.
- **Axios Interceptor** → attaches JWT token in headers for API calls.
- **Role-based access:**
 - `/api/admin/**` → ADMIN only
 - `/api/librarian/**` → LIBRARIAN only
 - `/api/member/**` → MEMBER

10. Benefits

- ✓ Centralized and automated library operations
- ✓ Secure authentication and role-based authorization

- ✓ Easy API documentation with Swagger
- ✓ Responsive UI with React + Bootstrap
- ✓ Scalable architecture with modular services