### Projects

### Library Management System Project Documentation

### 1. ****Project Overview****

The Library Management System is a full-stack web application designed to manage library resources efficiently. It allows users to perform CRUD (Create, Read, Update, Delete) operations on books, authors, and members. The system provides features for administrators to manage the library catalog and for members to view and borrow books.

### 2. ****Project Objectives****

* To develop a user-friendly web application for managing library operations.
* To implement CRUD operations using Java, MySQL, HTML, CSS, JavaScript, and React.js.
* To enhance skills in full-stack development, particularly in integrating front-end and back-end technologies.
* To practice database management and application security.

### 3. ****Technology Stack****

* **Front-End**: HTML, CSS, JavaScript, React.js
* **Back-End**: Java (Spring Boot)
* **Database**: MySQL
* **Build Tool**: Maven/Gradle
* **Version Control**: Git
* **IDE**: IntelliJ IDEA/Eclipse/VS Code

### 4. ****System Architecture****

The system follows a multi-tier architecture:

* **Presentation Layer**: React.js for the user interface.
* **Business Logic Layer**: Java Spring Boot for handling the application logic.
* **Data Access Layer**: Java Persistence API (JPA) with Hibernate ORM for database interactions.
* **Database Layer**: MySQL database for storing library data.

### 5. ****Core Features****

#### 5.1. ****Admin Module****

* **Add/Edit/Delete Books**: Admin can manage the library catalog by adding, editing, or deleting books.
* **Manage Authors**: Admin can add, update, or delete author information.
* **Manage Members**: Admin can manage member accounts, including adding new members or updating existing ones.
* **Track Borrowed Books**: Admin can view which books are borrowed and by whom.

#### 5.2. ****Member Module****

* **View Books**: Members can browse the catalog and view book details.
* **Search Books**: Members can search for books by title, author, or category.
* **Borrow/Return Books**: Members can borrow available books and return borrowed ones.
* **View Borrowing History**: Members can view their borrowing history.

### 6. ****Database Design****

#### 6.1. ****Tables****

* **Books**:
  + book\_id (Primary Key)
  + title
  + author\_id (Foreign Key)
  + category
  + isbn
  + quantity
  + available
* **Authors**:
  + author\_id (Primary Key)
  + name
  + bio
* **Members**:
  + member\_id (Primary Key)
  + name
  + email
  + phone\_number
  + membership\_date
* **Borrowed\_Books**:
  + borrow\_id (Primary Key)
  + book\_id (Foreign Key)
  + member\_id (Foreign Key)
  + borrow\_date
  + return\_date

#### 6.2. ****Entity-Relationship Diagram (ERD)****

* **Books**: Linked to Authors (One-to-Many)
* **Members**: Linked to Borrowed Books (One-to-Many)
* **Borrowed Books**: Linked to both Books and Members

### 7. ****User Interface Design****

#### 7.1. ****Admin Dashboard****

* Overview of the system (e.g., total books, members, borrowed books).
* Navigation menu to manage books, authors, and members.
* Tables and forms for CRUD operations on library entities.

#### 7.2. ****Member Interface****

* Home page displaying book categories and featured books.
* Search bar for finding specific books.
* Book detail pages with borrowing options.
* Profile page showing borrowing history and account details.

### 8. ****Backend Development****

#### 8.1. ****Spring Boot Application Setup****

* Create a Spring Boot project with dependencies like Spring Data JPA, Spring Web, and MySQL Connector.
* Configure the application.properties file for database connection.
* Implement service classes for handling business logic.
* Create RESTful APIs for CRUD operations on books, authors, and members.

#### 8.2. ****Security****

* Implement user authentication and authorization.
* Admins have full access to the system.
* Members have restricted access, limited to viewing and borrowing books.

### 9. ****Frontend Development****

#### 9.1. ****React.js Setup****

* Create a React.js project and set up the required dependencies.
* Design components for the Admin and Member interfaces.
* Use React Router for navigation between pages.
* Integrate with the backend API using Axios or Fetch API.

#### 9.2. ****Responsive Design****

* Use CSS Grid and Flexbox for responsive layouts.
* Ensure the application is mobile-friendly.

### 10. ****Testing****

#### 10.1. ****Unit Testing****

* Write JUnit test cases for backend services.
* Test each API endpoint using Postman or similar tools.

#### 10.2. ****Integration Testing****

* Ensure the integration between front-end and back-end is seamless.
* Test the complete user flow, from logging in to borrowing a book.

### 11. ****Deployment****

#### 11.1. ****Environment Setup****

* Set up a server (e.g., AWS EC2) for deployment.
* Install necessary software like Java, MySQL, and a web server.

#### 11.2. ****Deploying the Application****

* Build the Spring Boot application and deploy it on the server.
* Deploy the React.js application using a web server like Nginx or Apache.

### 12. ****Future Enhancements****

* **Notifications**: Implement email or SMS notifications for due dates.
* **Book Reservations**: Allow members to reserve books that are currently borrowed.
* **Reports**: Generate reports for admin insights, such as most borrowed books.
* **Machine Learning**: Recommend books to members based on their borrowing history.

### 13. ****Conclusion****

This Library Management System project will not only help you strengthen your understanding of full-stack development but also give you hands-on experience with integrating various technologies. By completing this project, you will be well-equipped with the skills needed for building robust web applications.