System Design Document

on

Library Management System

Course Module: (VLC-SCFSD10-25-0650)

Year: March2025

Author: Karen Chan

# Architecture

This document outlines the architecture for a web application built using **React** for the frontend, **Spring Boot** for the backend, and **MySQL** as the database.

## Technology Stack

* Frontend: React (JavaScript/TypeScript)
* Backend: Spring Boot (Java)
* Database: MySQL
* API Communication: RESTful APIs (JSON format)
* Authentication: JWT (JSON Web Token)
* Deployment: Docker, Kubernetes (Optional), AWS/Azure/GCP

## System Architecture

The system follows a three-tier architecture:

* Presentation Layer (Frontend - React): Handles UI rendering and user interactions.
* Business Logic Layer (Backend - Spring Boot): Processes requests, enforces business rules, and interacts with the database.
* Data Layer (MySQL): Stores application data persistently.

## Frontend (React)

* **State Management:** Redux, Context API
* **Routing:** React Router
* **API Calls:** Axios or Fetch API to communicate with Spring Boot
* **Component-based UI:** Follows modular component design
* **Security:** JWT stored in HttpOnly cookies or local storage

## Backend (Spring Boot)

* **Spring Boot Modules:**
  + Spring Web (REST APIs)
  + Spring Security (Authentication & Authorization)
  + Spring Data JPA (Database Interaction)
  + Spring Boot Actuator (Monitoring & Metrics)
* **Controllers:** Handle HTTP requests and responses
* **Services:** Contain business logic
* **Repositories:** Interface with MySQL using JPA/Hibernate

# Database

Database Design includes below:

* Schema Design:
  + Tables for Users, Roles, Transactions, etc.
  + Relationships (One-to-Many, Many-to-Many)
* Indexes & Optimization: To enhance query performance
* ORM: Hibernate (JPA) for object-relational mapping

## Entities:

1. **tbl\_users**
   * Attributes:
     + user\_id (Primary Key)
     + name
     + pwd
     + identity
     + email
     + phone
     + role
     + created\_by
     + created\_dt
2. **tbl\_books**
   * Attributes:
     + book\_id (Primary Key)
     + title
     + author
     + ISBN
     + genre
     + year\_published
     + status
     + created\_by
     + created\_dt
     + updated\_by
     + updated\_dt
3. **tbl\_borrow\_hist**
   * Attributes:
     + hist\_id (Primary Key)
     + book\_id (Foreign Key to tbl\_books)
     + user\_id (Foreign Key to tbl\_users)
     + borrow\_dt
     + return\_dt
     + status
     + created\_by
     + created\_dt

## Relationships:

* **tbl\_borrow\_hist** references **tbl\_books** and **tbl\_users** through the foreign keys book\_id and user\_id.
  + **Relationship 1**: Between **tbl\_users** and **tbl\_borrow\_hist**: One user can have many borrow history records, but each borrow record is associated with one user.
    - Cardinality: **One-to-Many** (One user can borrow many books).
  + **Relationship 2**: Between **tbl\_books** and **tbl\_borrow\_hist**: A book can be borrowed many times, and each borrowing record is associated with one book.
    - Cardinality: **One-to-Many** (One book can appear in many borrow records).

## ER Diagram

 **tbl\_users** has a one-to-many relationship with **tbl\_borrow\_hist** via user\_id.

 **tbl\_books** has a one-to-many relationship with **tbl\_borrow\_hist** via book\_id.

A screenshot of a computer screen

AI-generated content may be incorrect.

* The **tbl\_users** and **tbl\_books** tables are linked by **tbl\_borrow\_hist** via their respective foreign keys (user\_id, book\_id).
* Each **borrow\_hist** record ties together a **book** and a **user**, capturing details such as borrow date, return date, and status.

# API Design

API design as below

* RESTful APIs with **CRUD** operations
* **Endpoints Example:**
  + POST /api/auth/login - User login
  + GET /api/users/{id} - Fetch user details
  + POST /api/orders - Create an order
* **Response Formats:** JSON
* **Error Handling:** Global exception handling with proper HTTP status codes

## User Registration & Authentication

* AuthController
  + GET - [http://localhost:9000/api/auth](http://localhost:9000/api/auth/)
* PasswordResetController
  + GET - <http://localhost:9000/api/passwordreset>

## User Management

* UserController
  + GET /POST- <http://localhost:9000/api/users>

## Book Management

* BookController
* GET /POST- <http://localhost:9000/api/books>

## Lending Management

* BorrowHistoryController
* GET /POST- <http://localhost:9000/api/borrowhistory>

# UI mock-ups

## 4.1 User Registration & Authentication

### Registration

A screenshot of a computer screen

AI-generated content may be incorrect.

### Login

A screenshot of a login screen

AI-generated content may be incorrect.

### Reset Password

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## 4.2 User Management

### Main User List

A screenshot of a computer

AI-generated content may be incorrect.

### Add User

A screenshot of a computer

AI-generated content may be incorrect.

### Search User

A screenshot of a computer

AI-generated content may be incorrect.

## 4.3 Book Management

### Main Book List

A screen shot of a computer

AI-generated content may be incorrect.

### Add Book

A screenshot of a computer

AI-generated content may be incorrect.

### Search Book

To display full book list, do not enter search criteria.

A screenshot of a computer

AI-generated content may be incorrect.

## 4.4 Lending Management

### Main Lending list

#### Borrow button

* + Lending status changes to < borrowed > if < reserved >.
  + Book status change to < borrowed >.

#### Return button

* + Lending status changes to < returned > if < borrowed >
  + Book status change to < available >.

#### Release button

* + Lending status changes to < released > if < reserved >
  + Book status change to < available >.

A screenshot of a computer

AI-generated content may be incorrect.