**Software Design Specification Document**

**Project Title:** Library Management System  
**Course:** DBMS & SDA  
**Submitted By:** Shoaib Akhtar Marwat  
**Date:** May 2025

**A. Requirements Specification**

**1. User Stories in Jira (Epics & Features)**

**Epic: User Authentication**

* **User Story:** As a user, I want to securely log in to the system so that I can access my dashboard.
  + **Non-Functional Requirements (NFR):** Session timeout after inactivity, login response < 2 seconds, encryption of credentials.

**Epic: Book Management**

* **User Story:** As a librarian, I want to add, update, and delete book records.
  + **NFR:** Changes reflected in < 2s, no duplicate ISBN entries.

**Epic: Member Management**

* **User Story:** As a librarian, I want to register and manage member details.
  + **NFR:** Data must be validated, securely stored, and retrievable in < 3s.

**Epic: Reservation System**

* **User Story:** As a member, I want to reserve available books.
  + **NFR:** Reservation should lock the book, send email confirmation.

**Epic: Fine Management**

* **User Story:** As staff, I want to calculate fines based on due dates.
  + **NFR:** Accurate calculation based on return date, must support holiday exclusions.

**Epic: Admin Controls**

* **User Story:** As an admin, I want to monitor users and usage statistics.
  + **NFR:** Secure access, data visualization, logs retained for 6 months.

**Note:** Jira link will be shared and admin access given to the specified emails.

**2. General Requirements & Assumptions**

* Only authenticated users can access system features.
* Users are assigned roles: Admin, Staff, Member.
* Books can be added/edited/deleted by Admin or Staff.
* Members can reserve and return books.
* System calculates fines for overdue returns.
* Notifications via email for reservations and fines.
* Search functionality for books.
* Admin dashboard for user management and analytics.

**B. Design Specification**

**1. System Architecture and Design Patterns**

**Architecture Style:** Model-View-Controller (MVC)

**Justification:**

* Separates concerns among UI, logic, and data layers.
* Facilitates modifiability and scalability.
* Complies with design constraints related to UI decoupling and data security.

**Design Patterns Applied:**

* **Strategy Pattern** – Fine calculation logic.
* **Observer Pattern** – Notifications on reservation, returns.
* **Facade Pattern** – Simplifying admin features.
* **Singleton Pattern** – Database connection handler.
* **Factory Pattern** – For dynamic object creation.

**2. Detailed Design**

**a. Use Case Diagram**

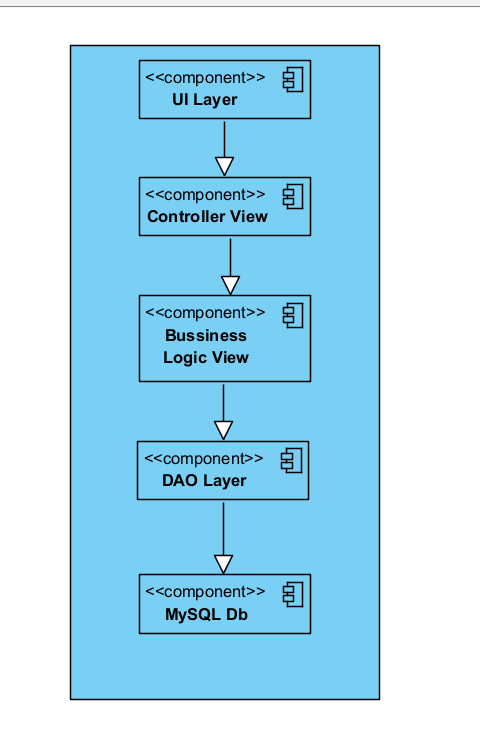
**A diagram of a diagram

AI-generated content may be incorrect.**

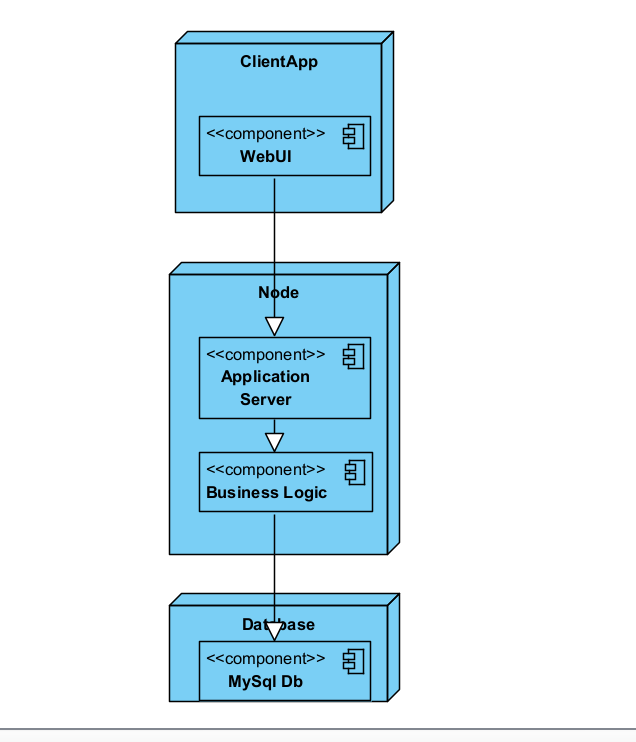
**b. UI Design (Description)**

* **Login Page:** Full-screen, modern black-orange theme
* **Dashboards:**
  + Admin: Manage users, view logs
  + Staff: Manage books, members
  + Member: View/reserve books

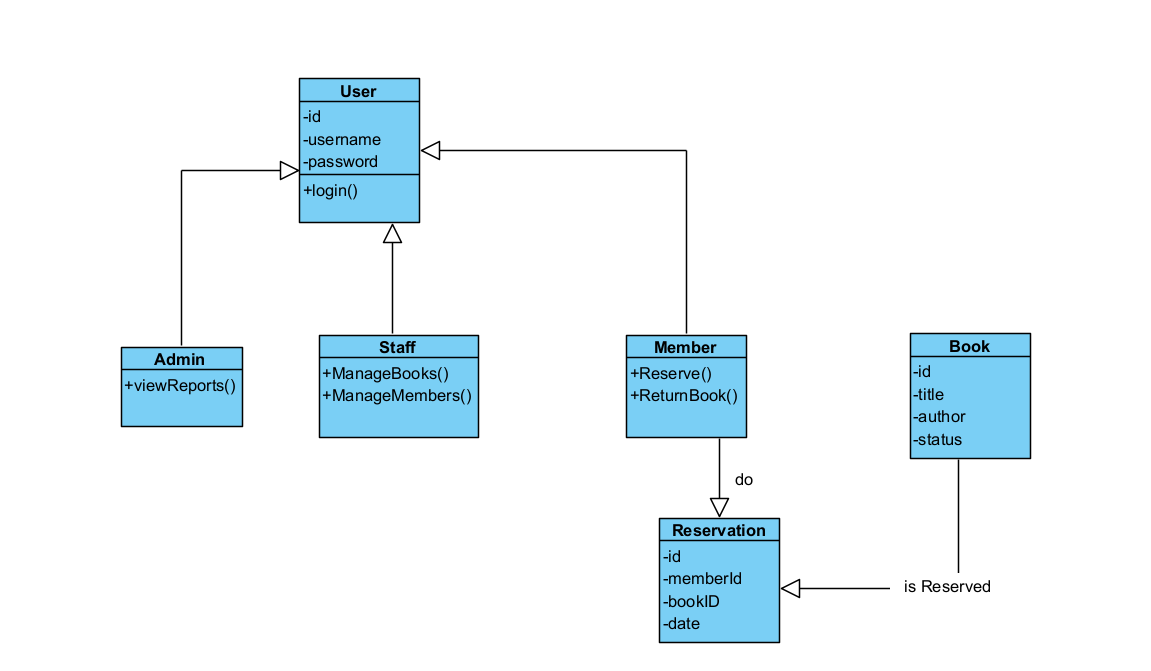
**c. Component Diagram**

****

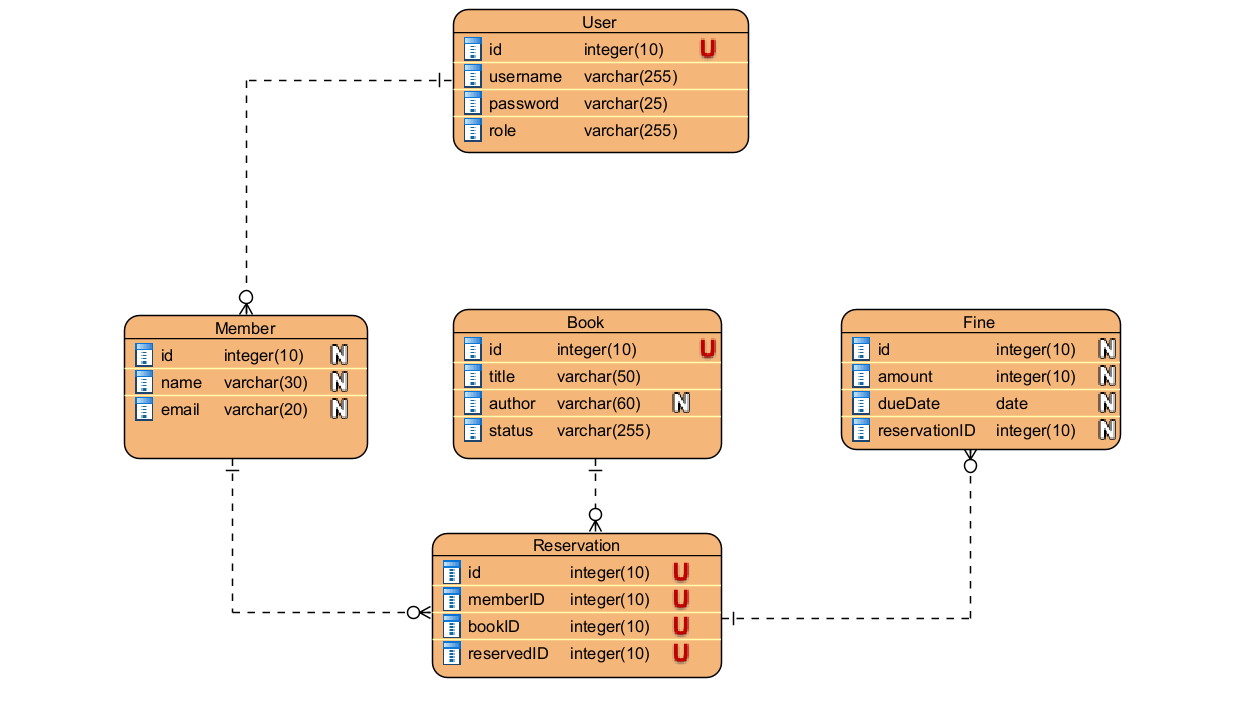
**d. Deployment Diagram**



**e. Class Diagram**

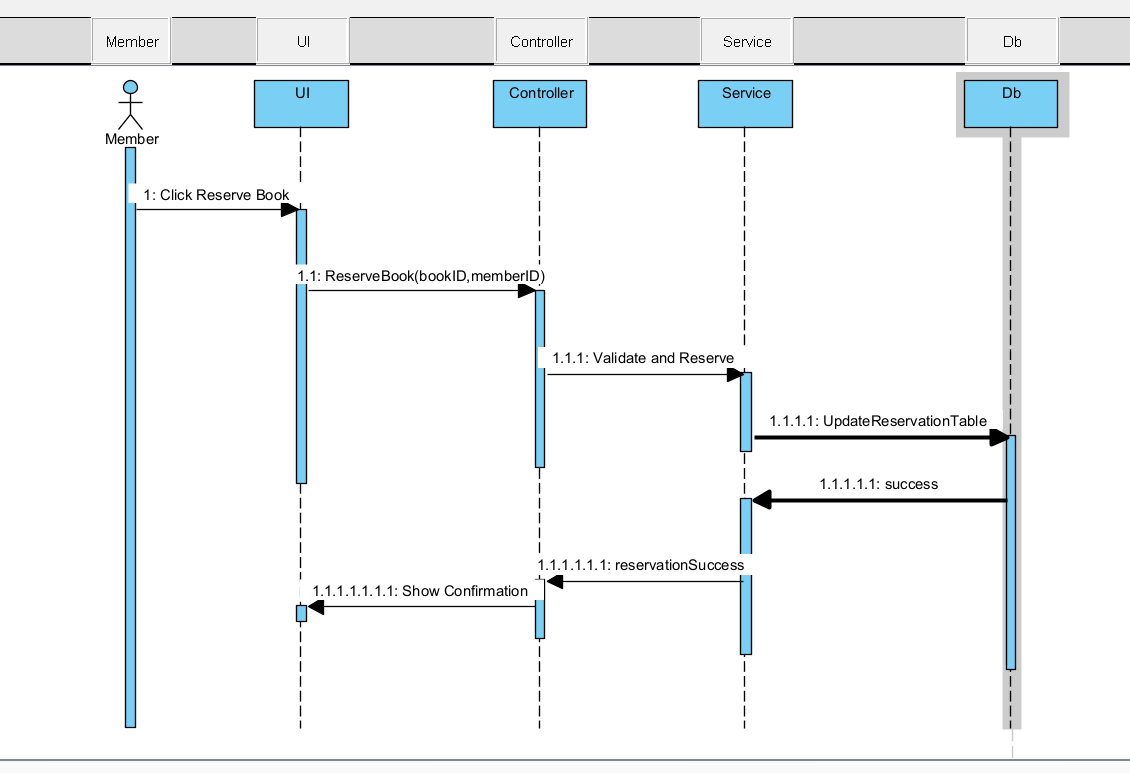
****

**f. Data Model**

****

**g. Dynamic Views**

**i. Sequence Diagram: Reserve Book**

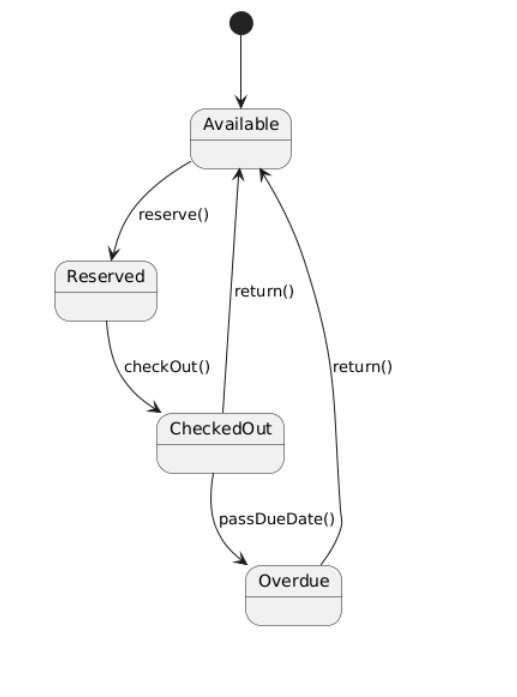


**ii. Activity Diagram: Book Return Process**

**A diagram of a process

AI-generated content may be incorrect.**

**iii. State Diagram: Book Lifecycle**



**Summary**

This document outlines a modular, scalable, and secure design for a Library Management System compliant with the project constraints and software engineering principles. All diagrams are created using PlantUML and can be rendered in IntelliJ, VS Code, or online tools.