# **Software Requirements Specification (SRS)**

## **Library Management System (LMS)**

## **1. Introduction**

### **1.1 Purpose**

The purpose of this document is to define the **functional, non-functional, and technical** requirements of the **Library Management System (LMS)**. The LMS will automate **book management, user registration, borrowing and returning of books, overdue tracking, and fines calculation**.

### **1.2 Document Conventions**

* **Functional Requirements (FR)** are labeled as **FR1, FR2, …**
* **Non-functional Requirements (NFR)** are labeled as **NFR1, NFR2, …**
* **Priority Levels:** High, Medium, Low

### **1.3 Intended Audience and Usage**

This document is intended for:

* **Developers** – Implementing the system.
* **Project Managers** – Managing timelines and milestones.
* **Testers** – Validating system functionality.
* **Stakeholders** – Understanding business objectives.

### **1.4 Scope**

The **Library Management System (LMS)** is a **full-stack web application** designed for:

* **Librarians (Admins)**: Manage books, track users, handle lending, and calculate overdue fines.
* **Library Members**: Search books, borrow, renew, and return books.

The system will be **developed using Spring Boot (Java), React.js (Vite), and MySQL**.

## **2. Overall Description**

### **2.1 Product Perspective**

The LMS is a **standalone web application** that will replace traditional, manual book-tracking methods.

### **2.2 Product Functions**

| **Functionality** | **Description** |
| --- | --- |
| **User Authentication** | Users can register, log in, and reset passwords (JWT-based authentication). |
| **Book Management** | Librarians can add, edit, delete, and search books. |
| **User Management** | Librarians can manage library members. |
| **Borrowing & Returning** | Members can borrow, renew, and return books. |
| **Overdue Tracking** | System calculates fines for overdue books. |
| **Notifications** | System sends reminders via email for due books. |

### **2.3 User Characteristics**

| **User Type** | **Technical Expertise** | **Access Level** |
| --- | --- | --- |
| **Librarian (Admin)** | Moderate | Manage books, members, and lending. |
| **Library Member** | Basic | Search, borrow, return books. |

### **2.4 Operating Environment**

| **Component** | **Specification** |
| --- | --- |
| **Backend** | Spring Boot, Java 17 |
| **Frontend** | React.js (Vite), JavaScript |
| **Database** | MySQL 8+ |
| **Hosting** | Cloud-based (AWS, DigitalOcean, or Firebase for frontend) |

### **2.5 Constraints**

* The system must support **100+ concurrent users**.
* Database transactions must follow **ACID compliance**.
* Role-based access control must be **strictly enforced**.

## **3. Functional Requirements**

### **3.1 User Authentication & Authorization**

| **ID** | **Requirement** | **Priority** |
| --- | --- | --- |
| **FR1** | Users must register with name, email, and password. | High |
| **FR2** | Login should be secured using **JWT authentication**. | High |
| **FR3** | Users must be assigned a **role (Admin or Member)**. | High |
| **FR4** | Users should be able to **reset their password via email verification**. | Medium |

### **3.2 Book Management**

| **ID** | **Requirement** | **Priority** |
| --- | --- | --- |
| **FR5** | Librarians can **add, edit, and delete books**. | High |
| **FR6** | Books should have attributes: **title, author, category, ISBN, copies available**. | High |
| **FR7** | Users should be able to **search books** by title, author, or category. | High |
| **FR8** | The system must track the **number of available copies** of each book. | High |

### **3.3 Borrowing & Returning Books**

| **ID** | **Requirement** | **Priority** |
| --- | --- | --- |
| **FR9** | Members can borrow books if copies are available. | High |
| **FR10** | The system should set a **due date (14 days from borrow date)**. | High |
| **FR11** | Members can **renew books twice** before they must be returned. | Medium |
| **FR12** | Members should be able to **return books** and update availability. | High |

### **3.4 Overdue Tracking & Fines**

| **ID** | **Requirement** | **Priority** |
| --- | --- | --- |
| **FR13** | System should calculate **$0.50/day as a fine for overdue books**. | High |
| **FR14** | Maximum fine per book is **$20**. | High |
| **FR15** | Members should be **restricted from borrowing** if total fines exceed **$10**. | High |
| **FR16** | System should send **email reminders** for due and overdue books. | Medium |

## **4. Non-Functional Requirements**

### **4.1 Performance Requirements**

| **ID** | **Requirement** | **Priority** |
| --- | --- | --- |
| **NFR1** | System should **support at least 100 concurrent users**. | High |
| **NFR2** | Book searches should return results within **1 second**. | High |
| **NFR3** | System should handle **500+ book transactions per day**. | Medium |

### **4.2 Security Requirements**

| **ID** | **Requirement** | **Priority** |
| --- | --- | --- |
| **NFR4** | User passwords must be stored using **bcrypt hashing**. | High |
| **NFR5** | Only **librarians** can add/edit/delete books. | High |
| **NFR6** | All API endpoints must require **JWT authentication**. | High |

### **4.3 Availability Requirements**

| **ID** | **Requirement** | **Priority** |
| --- | --- | --- |
| **NFR7** | System uptime should be **99.9%**. | High |
| **NFR8** | Automatic **database backup** should occur **daily**. | High |

## **5. System Design Overview**

### **5.1 System Architecture**

* **Backend:** Spring Boot (REST API)
* **Frontend:** React.js (Vite)
* **Database:** MySQL
* **Authentication:** JWT-based

### **5.2 Entity Relationship Diagram (ERD)**

| **Entity** | **Attributes** |
| --- | --- |
| **User** | id, name, email, password, role (ADMIN/MEMBER) |
| **Book** | id, title, author, category, copies\_available, isbn |
| **Member** | id, user\_id (FK), membership\_status |
| **Lending** | id, book\_id (FK), member\_id (FK), borrow\_date, return\_date, status |

## **6. Testing & Validation**

| **Test Type** | **Scope** |
| --- | --- |
| **Unit Testing** | Validate API responses (JUnit) |
| **Integration Testing** | Ensure frontend and backend work together |
| **User Acceptance Testing (UAT)** | Test book borrowing and returning workflow |
| **Security Testing** | Test authentication and unauthorized access prevention |

## **7. Deployment Plan**

| **Environment** | **Technology** |
| --- | --- |
| **Development** | Localhost, Docker |
| **Staging** | AWS EC2, RDS (MySQL) |
| **Production** | Cloud-based deployment |

## **8. Conclusion**

This **SRS** outlines the functional, non-functional, and technical requirements for the **Library Management System (LMS)**. It serves as a blueprint for **developers, testers, and stakeholders** to ensure successful project implementation.