Software Requirements Specification

for

Library Management System – Book Issue and Deposit

Prepared by

Ch. Vamshi Krishna Babu	**SE22UARI038**	**se22uari038@mahindrauniversity.edu.in**
Ch. Vamsi Krishna	**SE22UARI039**	**se22uari039@mahindrauniversity.edu.in**
K. Sri Ram Sharan	**SE22UARI070**	**se22uari070@mahindrauniversity.edu.in**
G. Rithvik Chandan	**SE22UARI144**	**se22uari144@mahindrauniversity.edu.in**
V. Raja	**SE22UARI178**	**se22uari178@mahindrauniversity.edu.in**
K. Bhuvan	**SE22UARI212**	**se22uari212@mahindrauniversity.edu.in**

Instructor: Avinash Arun Chauhan

Course: Software Engineering

Lab Section: AI

Teaching Assistant: Nartakannai K

Date: 10-03-2025

Contents

1 Introduction

1.1 Document Purpose

This document specifies the software requirements for the Library Management System – Book Issue and Deposit. The system aims to automate library operations such as book borrowing, returning, and fine calculation. The document serves as a reference for developers, testers, and stakeholders to understand system functionality and constraints.

1.2 Product Scope

The system provides a web-based solution for managing library books efficiently. It allows students and faculty to borrow and return books digitally while ensuring real-time book availability tracking. The system will help reduce manual work, improve accuracy, and enhance the user experience for both library members and administrators.

1.3 Intended Audience and Document Overview

This document is intended for:
- **Developers** – To understand system requirements and implementation constraints.
- **Testers** – To verify system functionality.
- **Project Managers** – To track progress.
- **Clients (University Library)** – To review system features and expectations.

1.4 Definitions, Acronyms and Abbreviations

- **LMS** – Library Management System
- **UI** – User Interface
- **DBMS** – Database Management System
- **React.js** – JavaScript library for building user interfaces

1.5 Document Conventions

This document follows the IEEE SRS format. Standard fonts include Arial size 11 or 12. Headings are bolded, and section numbering follows IEEE recommendations.

1.6 References and Acknowledgments

- **IEEE Software Engineering Standards**
  Referred for guidelines on software requirement specifications, design principles, and documentation practices.
- **Java Official Documentation**
  Consulted for backend development, REST API implementation, and handling server-side operations.
- **React.js Official Documentation**
  Used for designing and developing the frontend user interface.

- **PostgreSQL Official Documentation**
  Used as a reference for database schema design, query optimization, and data management.

2 Overall Description

2.1 Product Overview

The **Library Management System (LMS)** is a comprehensive software solution designed to streamline and automate the process of managing a library's daily operations. It provides an intuitive user interface for library members and staff while ensuring robust backend functionality for efficient data management and system security.

The system enables users to search, borrow, and return books, while administrators and librarians can manage inventory, monitor transactions, and enforce borrowing policies. Additionally, the system calculates fines for overdue returns and maintains user transaction histories.

This system follows a **client-server architecture**, with the **React.js frontend** interacting with a **Java backend** via RESTful APIs, and **PostgreSQL** as the underlying database for persistent data storage. The application offers scalability, role-based access, and an intuitive interface for a seamless library management experience.

2.2 Product Functionality

The system provides the following features:
- **User Authentication** – Secure login for students, faculty, and librarians.
- **Book Issue and Return** – Digital borrowing and deposit tracking.
- **Fine Calculation** – Automatic calculation of overdue fines.
- **Book Search and Filter** – Find books by title, author, or genre.
- **Admin Dashboard** – Librarians can add, update, or remove books.
- **Real-time Book Availability Updates**

2.3 Design and Implementation Constraints

- The frontend must be developed in **React.js.**
- **PostgreSQL (cloud-based free-tier)** will be used for data storage.
- The system must support up to **100 concurrent users.**

2.4 Assumptions and Dependencies

- Users must have university credentials for access.
- Book records will be manually updated by librarians.
- The system will primarily be accessed via desktops and laptops.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The primary user interface is a web-based application accessible through modern browsers. The interface consists of two dashboards:

- **Student/User Dashboard**
  - View available books
  - Search and filter books by title, author, or genre
  - Request issue/return of books
  - View currently borrowed books and due dates
  - Receive notifications for due dates/fines
- **Librarian/Admin Dashboard**
  - Add new books or remove outdated ones
  - Approve/decline issue and return requests
  - Manage user accounts
  - View reports of fines, issued/returned books, and inventory status

The UI design follows a clean layout with a sidebar menu for navigation, tables for displaying data, and modal forms for actions like book issue/return. The user interacts via keyboard and mouse/touch (for mobile-responsive versions).

3.1.2 Hardware Interfaces

The system runs on standard web servers and client devices (PCs, laptops, smartphones). No specialized hardware is required.

- **Server**: Cloud-hosted PostgreSQL database and Java backend deployed on a standard cloud service (AWS/Heroku).
- **Clients**: End-users access the system through web browsers.
  For barcode scanning (optional future enhancement), standard barcode scanner hardware can be integrated to speed up book management.

3.1.3 Software Interfaces

- **Frontend-Backend Interface**:
  RESTful API endpoints built using Java (Spring Boot framework) handle data transactions between the React.js frontend and the Java backend. The frontend communicates with these APIs over HTTP/HTTPS, sending requests and receiving responses in JSON format. These

endpoints manage operations such as user authentication, book issue and return processes, fine calculations, and search/filter functionalities.

- **Database Interface**:
  PostgreSQL database handles all data storage and retrieval operations for books, users, transactions, and fines.
- **Authentication System**:
  The authentication system is implemented using Java (Spring Security framework). It handles login/logout functionality and session management, including user roles such as **Admin**, **Librarian**, and **User**. The system ensures secure access control through role-based authorization and uses JWT (JSON Web Tokens) for stateless authentication across the React.js frontend and Java backend.

3.2 Functional Requirements

3.2.1.1 F1: The system shall allow users to register and log in.

- Users (students) and librarians can create accounts or log in securely.
- Role-based access controls will restrict features based on user type.

3.2.1.2 F2: The system shall allow users to search and filter books.

- Users can search books by title, author, genre, or availability.
- Filter options will include categories like fiction, non-fiction, academic, etc.

3.2.1.3 F3: The system shall allow users to issue and return books.

- Users can request to issue a book, and librarians can approve/decline the request.
- Users can initiate a return request before or on the due date.

3.2.1.4 F4: The system shall calculate fines for overdue books.

- The system will automatically calculate fines based on overdue days.
- Users will receive notifications about pending fines.

3.2.1.5 F5: The system shall allow librarians to manage inventory.

- Librarians can add, update, or remove book records.
- Librarians can track which books are issued and available.

3.2.1.6 F6: The system shall generate reports for admins.

- Reports will include issued books, returned books, overdue items, and fine summaries.

3.3 Use Case Model

3.3.1 Use Case #1 (User Registration and Login – U1)

- **Author**: Ch. Vamsi Krishna
- **Purpose**: To allow users (students and librarians) to securely register and log into the system to access their respective dashboards.
- **Requirements Traceability**: FR1 (User Authentication), FR2 (Role-Based Access Control)
- **Priority**: High
  (Essential for accessing all other system functions. Without this, users can't interact with the system.)
- **Preconditions**:
  - The user has internet access and a device with a browser.
  - The user is not already logged into the system.
- **Postconditions**:
  - The user is authenticated and redirected to their respective dashboard (Student or Librarian).
- **Actors**:
  - Primary: User (Student/Librarian)
  - Supporting: Authentication Service
- **Extends**: None
- **Flow of Events**:
  - **Basic Flow**:
    - User accesses the login/registration page.
    - User enters valid credentials (or registration details if new).
    - System validates the input.
    - On success, the system grants access and redirects to the user dashboard.
  - **Alternative Flow**:
    - User clicks “Forgot Password” link.
    - System sends password reset instructions to the registered email.
    - User resets password and logs in successfully.
  - **Exceptions**:
    - Invalid credentials: Display an error message and prompt retry.
    - Account locked due to multiple failed attempts: Show a locked account message and suggest contacting admin.
- **Includes**:
  - Password Reset Use Case (U3)
- **Notes/Issues**:
  - Ensure password strength and validation policies.
  - Implement CAPTCHA for preventing bot registration/login attacks.

3.3.2 Use Case #2 (Book Issue and Return – U2)

- **Author**: Ch. Vamsi Krishna

- **Purpose**: To allow users to request the issue or return of books and for librarians to approve/decline these requests. This ensures efficient management of book inventory and user transactions.
- **Requirements Traceability**: FR3 (Book Issue/Return), FR5 (Inventory Management)
- **Priority**: High
  (This is a core functionality of the Library Management System; without it, book circulation won't be possible.)
- **Preconditions**:
  - User is logged in to the system.
  - User has no pending fines (for issuing a new book).
  - Requested book is available in stock.
- **Postconditions**:
  - If approved, the book is marked as issued to the user and inventory is updated.
  - If returned, the book is marked as available and any applicable fines are recorded.
- **Actors**:
  - Primary: Student/User
  - Secondary: Librarian (who approves/declines the request)
  - Supporting: Book Inventory System (Database)
- **Extends**:
  - Fine Calculation Use Case (U4) – for overdue returns.
- **Flow of Events**:
  - **Basic Flow (Book Issue)**:
    - User searches for a book and clicks "Request Issue."
    - The request is sent to the librarian for approval.
    - Librarian reviews the request and checks user eligibility (e.g., no overdue books/fines).
    - Librarian approves the request.
    - System updates the book status as "Issued" and assigns it to the user.
    - Due date is generated, and the user is notified.

Basic Flow (Book Return):

- User selects the issued book and clicks "Request Return."
- The request is sent to the librarian for confirmation.
- Librarian verifies the condition and return date.
- System updates the book status as "Available."
- If returned late, the system calculates and records fines.
- User receives confirmation of the return.

  - **Alternative Flow**:
    - If the book is not available, the system shows "Out of Stock" and provides an option to join a waitlist.
    - If the user has reached their issue limit, the system denies further issue requests.
  - **Exceptions**:
    - System failure during the transaction: Show an error message and retry the operation.
    - Database not updated: Librarian manually intervenes to update the book record.
- **Includes**:
  - Notification Use Case (U5) – sends alerts about due dates and return confirmations.
  - Fine Calculation Use Case (U4) – for overdue returns.
- **Notes/Issues**:
  - Ensure real-time update of inventory to avoid double-booking.
  - Consider adding an automatic reminder system for due dates.

3.3.3 Use Case #3 (Fine Calculation – U4)

- **Author**: Ch. Vamsi Krishna
- **Purpose**: Automatically calculate fines for overdue book returns and notify users about the due amount.
- **Requirements Traceability**: FR6 (Fine Management), FR5 (Inventory Management)
- **Priority**: Medium
  (While not blocking core operations, fines ensure timely returns and system discipline.)
- **Preconditions**:
  - The user has one or more books issued.
  - The due date for returning the book has passed.
- **Postconditions**:
  - The system calculates the fine based on the number of overdue days.
  - Fine is added to the user's account and is displayed in their dashboard.
  - The user is notified via email/notification.
- **Actors**:
  - Primary: System
  - Secondary: Librarian (may waive fines manually in special cases)
  - Supporting: User (who views and pays the fine)
- **Extends**:
  - Book Issue and Return Use Case (U2)
- **Flow of Events**:
  - **Basic Flow**:
    - System runs a daily check for overdue books.
    - For each overdue book, calculate the fine as:
      `Fine Amount = Number of Overdue Days × Fine Rate (e.g., ₹5 per day)`
    - Update the user's account with the total fine.
    - Notify the user via system notifications or email.
  - **Alternative Flow**:
    - Librarian reviews the fines and manually adjusts (waives/reduces) them if required.
    - Fine adjustments are logged for audit purposes.
  - **Exceptions**:
    - System time/date inconsistency: Ensure synchronization with the server time to avoid incorrect fine calculations.
    - Payment system failure during fine payment: Notify user and retry payment later.
- **Includes**:
  - Notifications Use Case (U5) – for alerting users about fines and payment reminders.
- **Notes/Issues**:
  - Consider adding support for online fine payments (integration with payment gateway).
  - Provide a clear fine history for transparency.

3.3.4 Use Case #4 (Notification System – U5)

- **Author**: Ch. Vamsi Krishna

- **Purpose**: To notify users and librarians about important actions such as book due dates, fines, successful book issues/returns, and other alerts.
- **Requirements Traceability**: FR7 (Notification System)
- **Priority**: Medium
  (Notifications are important for user engagement but not mandatory for core operations.)
- **Preconditions**:
  - User has an active account with valid email/contact details.
  - Trigger events like issue/return/fine due are detected by the system.
- **Postconditions**:
  - User receives timely alerts/notifications about relevant events.
  - User takes necessary actions based on notifications (return books, pay fines, etc.)
- **Actors**:
  - Primary: System
  - Secondary: Users, Librarians
- **Extends**:
  - Book Issue and Return (U2)
  - Fine Calculation (U4)
- **Flow of Events**:
  - **Basic Flow**:
    - Event triggers a notification (e.g., book due date approaching).
    - System generates a message template (email/SMS/notification).
    - Notification is sent to the user's registered email/phone/system dashboard.
    - System logs the notification for future reference.
  - **Alternative Flow**:
    - User unsubscribes from non-essential notifications but continues to receive critical alerts (fines, due dates).
  - **Exceptions**:
    - Email delivery failure: System retries sending notifications or marks as failed.
    - Invalid contact details: User is prompted to update their contact info.
- **Includes**:
  - Fine Calculation (U4), Book Issue/Return (U2)
- **Notes/Issues**:
  - Ensure notification templates are clear and concise.
  - Consider adding push notifications if you have a mobile app.

4 Other Non-functional Requirements

4.1 Performance Requirements

- **P1**: The system must be capable of handling simultaneous requests from at least **100 users** without degradation in performance.
- **P2**: The search functionality should return results within **2 seconds** for queries related to book availability or user accounts.
- **P3**: Fine calculation and notifications for overdue books must be processed daily between **12 AM and 2 AM**, ensuring all fines and alerts are updated before users log in the next day.
- **P4**: The system's uptime should be **99.9%**, excluding scheduled maintenance windows.

4.2 Safety and Security Requirements

- *The system must require **user authentication** (username and password) for all users accessing personal data or book issue/return functionalities.*
- All communication between client and server should be encrypted using **SSL/TLS** to ensure data privacy.
- Role-based access control must be implemented:
- **Users** can view/search books, issue/return books, and check fines.
- **Librarians** can manage book inventories, issue/return books on behalf of users, waive fines, and approve new user registrations.
- **Admins** can add/remove users and librarians, manage fine policies, and view system logs.
- Sensitive data (passwords) must be stored in the database using **secure hashing algorithms** (e.g., bcrypt).
- The system should automatically log out users after **10 minutes** of inactivity to prevent unauthorized access.
- Backups of the entire system database must be performed **daily** to protect against data loss.

4.3 Software Quality Attributes

4.3.1 4.3.1 Reliability

- The system should recover automatically from server crashes with minimal downtime (within **5 minutes**).
- Database transactions should be **ACID-compliant** to ensure data consistency during operations like issuing or returning books.

4.3.2 4.3.2 Availability

- The system should be available to users **24/7**, with planned maintenance scheduled during off-peak hours (e.g., **2 AM - 4 AM**).
- In case of an unexpected failure, the system should redirect users to a maintenance page with an estimated recovery time.

4.3.3 4.3.3 Usability

- The user interface should be intuitive and require **no more than 3 clicks** to perform common tasks such as searching for books, issuing a book, or checking fines.
- Tooltips and error messages must be provided to guide users in case of invalid inputs or system errors.

4.3.4 4.3.4 Maintainability

- The system must have a **modular code structure** to facilitate easy updates and feature additions.
- All code must be documented according to the **Doxygen** standard used in the project.

4.3.5 4.3.5 Scalability

- The system should be scalable to accommodate up to **1000 users** and a **book database of 100,000 entries** without significant performance impact.

5 Other Requirements

● **Data Backup**: Database must be backed up daily.
● **Legal Compliance:** System follows university IT policies.

Appendix A – Data Dictionary

Name	Type	Description	Possible Values / Format	Related Operations
User_ID	String	Unique identifier for each user	Alphanumeric (e.g., U12345)	Create User, Update User, Delete User, Search User
User_Role	Enum	Role of the user in the system	`Student`, `Faculty`, `Librarian`, `Admin`	Access Control, Permission Validation
Book_ID	String	Unique identifier for each book	Alphanumeric (e.g., B12345)	Add Book, Issue Book, Return Book, Delete Book
Book_Title	String	Title of the book	Any text	Search Book, Add Book, Update Book
Author	String	Author name of the book	Any text	Search Book, Add Book
Genre	String	Genre/category of the book	e.g., Fiction, Non-Fiction, Science, History	Search Book, Filter Book
Issue_Date	Date	Date when the book was issued	`YYYY-MM-DD`	Issue Book, Return Book, Overdue Calculation
Return_Date	Date	Expected date of return for the issued book	`YYYY-MM-DD`	Return Book, Fine Calculation
Fine_Per_Day	Constant	Fine charged per day after due date	Numeric (e.g., ₹5)	Fine Calculation
Max_Issue_Limit	Constant	Maximum number of books a user can issue	Numeric (e.g., 3 for students, 5 for faculty)	Issue Book Validation
Book_Status	Enum	Status of the book	`Available`, `Issued`, `Reserved`, `Lost`	Check Availability, Update Status
Transaction_ID	String	Unique identifier for each transaction	Alphanumeric (e.g., T56789)	Track Transaction, Generate Reports
Login_Status	Boolean	User login session status	`True` (Logged In), `False` (Logged Out)	Authenticate, Logout
Feedback	String	User-submitted feedback or suggestions	Any text (Max 500 characters)	Submit Feedback, View Feedback
Notification_Type	Enum	Type of notification to the user	`Due Reminder`, `Overdue Notice`, `Announcement`	Send Notification
Database_Backup_Frequency	Constant	Frequency of database backup operations	Daily (Scheduled at 00:00 IST)	Backup Operation

- **Constants**: Fixed values like fines and limits.
- **State Variables**: Elements that change based on actions (e.g., Book Status, Login Status).
- **Inputs/Outputs**: What users interact with (e.g., User_ID is input on login; Notification is an output).

Appendix B - Group Log

Date	Task	Member Responsible
01/02/2025	Project Planning	All Members
05/02/2025	Requirement Gathering	All Members
07/02/2025	Statement of Work(SoW)	Se22uari039
01/03/2025	Backend Development	Se22uari038, Se22uari212
01/03/2025	Frontend & UI Design	Se22uari070, Se22uari144, Se22uari178
10/03/2025	SRS	Se22uari039