Software Requirements Specification

for

Library Management System

Team Name: Team Innovators

Team Members:
SAAD NASIM
M.TAHA KHAN
ZARRAR JAFFAR
ALISHBA NAVEED

Submission Date: March 7, 2025

Contents

1 Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to outline the requirements for the Library Management System (LMS), a software solution designed to streamline library operations. The system aims to facilitate efficient management of books, borrowing processes, and inventory for students, librarians, and administrators, reducing manual effort and enhancing accessibility.

1.2 Scope

The Library Management System will provide functionalities including user registration, book catalog management, book search and borrowing, automated due date notifications, overdue fine management, and administrative reporting. Developed as a web-based application, it will integrate with the university's student database for authentication and email services for notifications, targeting students, librarians, and administrators as primary users.

1.3 Definitions, Acronyms, and Abbreviations

- **LMS**: Library Management System
- **User**: Refers to students, librarians, or administrators interacting with the system
- **Book**: A physical item in the library's inventory
- **Borrowing**: The process by which a student takes a book for a specified period
- **Overdue**: A state where a book is not returned by its due date

1.4 References

- IEEE 830-1998: IEEE Recommended Practice for Software Requirements Specifications
- Project Document.pdf
- Library_Management_Sprint_1.pdf

1.5 Overview

This document is structured as follows: Section 2 provides an overall description of the LMS, including its perspective, functions, user characteristics, constraints, and assumptions. Section 3 details the specific requirements, encompassing functional and non-functional requirements, a use case diagram, user stories with pre- and post-conditions, sequence diagrams, and a class diagram.

2 Overall Description

2.1 Product Perspective

The Library Management System is a standalone web application that interfaces with the university's student database for user authentication and email services for notifications. It aims to provide a centralized platform for library operations, enhancing efficiency and user experience.

2.2 Product Functions

The LMS will offer the following core functionalities:

- User registration and authentication
- Book catalog management (add, update, remove books)
- Book search and filtering
- Book borrowing and returning
- Automated due date reminders
- Overdue fine management
- Administrative reporting and analytics

2.3 User Characteristics

- **Students**: Require features to register, search for books, borrow and return them, and receive due date notifications.
- **Librarians**: Need tools to manage book inventory, update availability, process returns, and handle overdue fines.
- **Administrators**: Responsible for monitoring system performance, generating reports, and managing user access.

2.4 Constraints

- The system must be implemented using Java, Hibernate, Spring Boot, and MySQL database.
- Must support at least 500 concurrent users.
- Must maintain 99.5% uptime, excluding scheduled maintenance.

2.5 Assumptions and Dependencies

- The university's student database is accessible for authentication.
- Email services are available for sending notifications.
- Users possess basic computer literacy to navigate the web interface.

3 Specific Requirements

3.1 External Interface Requirements

- **User Interfaces**: Web-based interface accessible via standard browsers (e.g., Chrome, Firefox).
- **Hardware Interfaces**: Standard server and client hardware.
- **Software Interfaces**: Integration with university student database and SMTP email services.
- **Communication Interfaces**: HTTP/HTTPS for web access, SMTP for notifications.

3.2 Functional Requirements

The LMS shall provide the following functionalities:

1. **User Registration**: Allow students to create accounts with necessary details.
2. **User Authentication**: Enable secure login for all users (students, librarians, administrators).
3. **Book Search**: Permit students to search books by title, author, genre, or availability.
4. **Book Borrowing**: Allow students to borrow available books directly.
5. **Book Returning**: Enable students to return borrowed books.
6. **Book Management**: Allow librarians to add, update, and remove books from the inventory.
7. **Overdue Management**: Enable librarians to track overdue books and apply fines.
8. **Reporting**: Allow administrators to generate reports on library usage and system performance.

Additionally, the system shall send automated due date reminders to students via email.

3.2.1 Use Case Diagram

The following use case diagram illustrates the overall functionality of the LMS:

Description: The diagram includes three actors: Student, Librarian, and Administrator, with the following use cases:

- **Student**: Register Account, Login, Search for Books, Borrow Book, Return Book
- **Librarian**: Login, Add Book, Update Book, Remove Book, Manage Overdues
- **Administrator**: Login, Generate Reports, Monitor Performance

Associations connect actors to their respective use cases, with ”Login” shared across all actors.

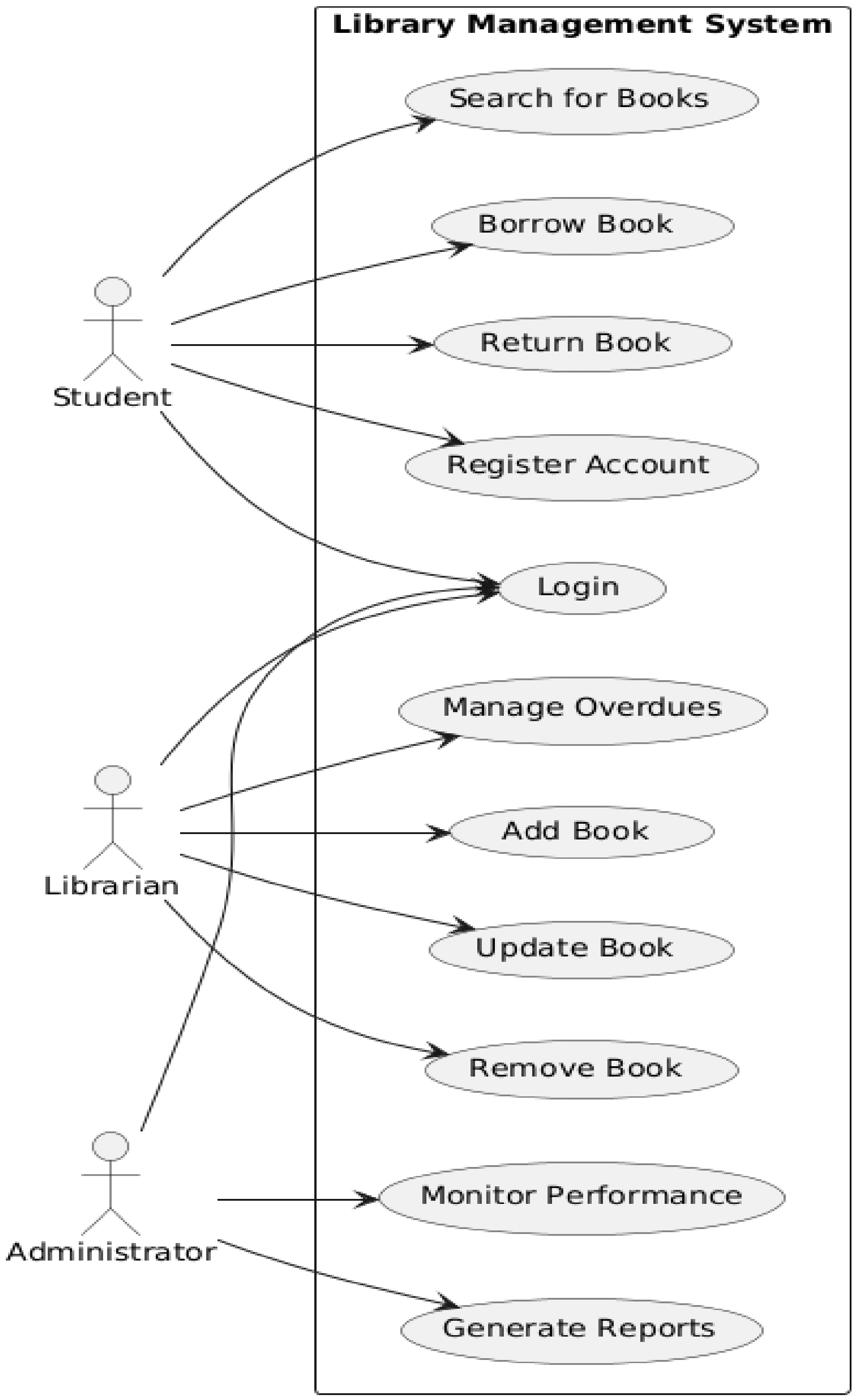

Figure 1: Use Case Diagram for Library Management System

3.2.2 User Stories

Based on the use case diagram, the following user stories are defined:

1. **US1**: As a student, I want to register an account so that I can access the library system.
2. **US2**: As a user, I want to log in to the system so that I can use its features.
3. **US3**: As a student, I want to search for books so that I can find the books I need.
4. **US4**: As a student, I want to borrow a book so that I can take it home for studying.
5. **US5**: As a student, I want to return a book so that I can avoid overdue fines.
6. **US6**: As a librarian, I want to add a book to the inventory so that it is available for borrowing.
7. **US7**: As a librarian, I want to update book information so that the catalog remains accurate.
8. **US8**: As a librarian, I want to remove a book from the inventory so that it is no longer available.
9. **US9**: As a librarian, I want to manage overdue books and fines so that students return books on time.
10. **US10**: As an administrator, I want to generate reports so that I can analyze library usage.
11. **US11**: As an administrator, I want to monitor system performance so that I can ensure it runs smoothly.

Pre- and Post-Conditions for User Stories:

- **US1: Register Account**
  - *Preconditions*: Student is not registered; registration page is accessible.
  - *Flow*: (1) Open registration page, (2) Enter details (name, student ID, email, password), (3) Submit form.
  - *Postconditions*: Student account is created; confirmation email is sent.
- **US2: Login**
  - *Preconditions*: User has an account; login page is accessible.
  - *Flow*: (1) Enter username and password, (2) Click login.
  - *Postconditions*: User is authenticated and redirected to the dashboard.
- **US3: Search for Books**
  - *Preconditions*: Student is logged in; search page is accessible.
  - *Flow*: (1) Enter search criteria (title, author, genre), (2) View results.

  - *Postconditions*: List of matching books is displayed.

- **US4: Borrow Book**
  - *Preconditions*: Student is logged in; book is available.
  - *Flow*: (1) Select book, (2) Click "Borrow," (3) Confirm request.
  - *Postconditions*: Book is marked as borrowed; borrowing record is created.
- **US5: Return Book**
  - *Preconditions*: Student is logged in; has borrowed books.
  - *Flow*: (1) Select borrowed book, (2) Click "Return."
  - *Postconditions*: Book is marked as returned; borrowing record is updated.
- **US6: Add Book**
  - *Preconditions*: Librarian is logged in; book management page is accessible.
  - *Flow*: (1) Enter book details (title, author, ISBN), (2) Save.
  - *Postconditions*: Book is added to the inventory.
- **US7: Update Book**
  - *Preconditions*: Librarian is logged in; book exists in inventory.
  - *Flow*: (1) Select book, (2) Edit details, (3) Save.
  - *Postconditions*: Book information is updated.
- **US8: Remove Book**
  - *Preconditions*: Librarian is logged in; book exists in inventory.
  - *Flow*: (1) Select book, (2) Click "Remove," (3) Confirm.
  - *Postconditions*: Book is removed from the inventory.
- **US9: Manage Overdues**
  - *Preconditions*: Librarian is logged in; overdue books exist.
  - *Flow*: (1) View overdue books, (2) Notify students, (3) Apply fines if necessary.
  - *Postconditions*: Overdue records are updated; fines are applied.
- **US10: Generate Reports**
  - *Preconditions*: Administrator is logged in; reporting tool is accessible.
  - *Flow*: (1) Select report type, (2) Generate report.
  - *Postconditions*: Report is generated and displayed.
- **US11: Monitor Performance**
  - *Preconditions*: Administrator is logged in; monitoring tool is accessible.
  - *Flow*: (1) View performance metrics.
  - *Postconditions*: Performance data is displayed.

3.2.3 Sequence Diagrams

The following sequence diagrams depict the three main activities of the LMS:

1. **Borrowing a Book Description**:

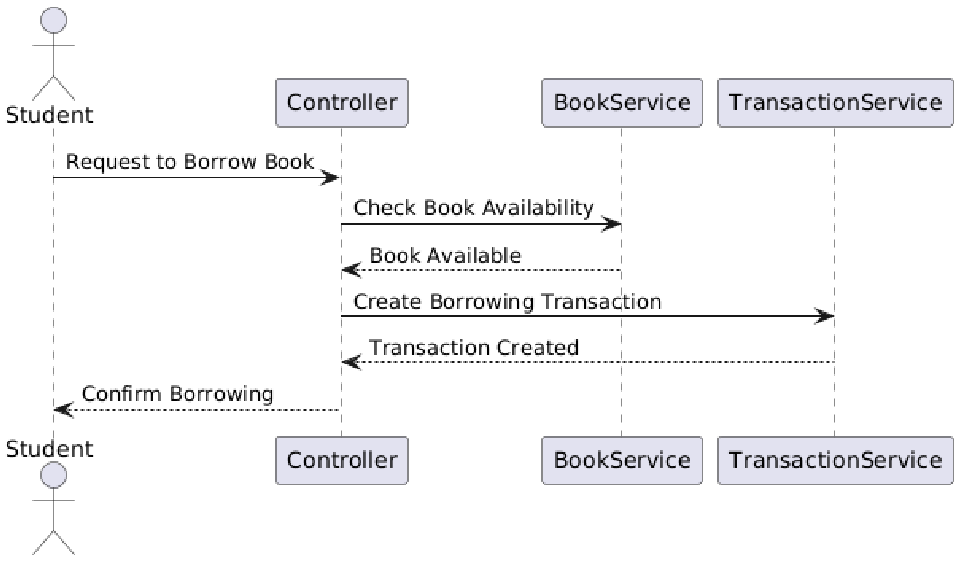

Figure 2: Sequence Diagram: Borrowing a Book

- Student logs in.
- Student searches for a book; system displays results.
- Student selects a book and clicks ”Borrow.”
- System checks availability; if available, updates book status and creates a borrowing record.
- System sends confirmation to the student.

2. **Returning a Book Description**:

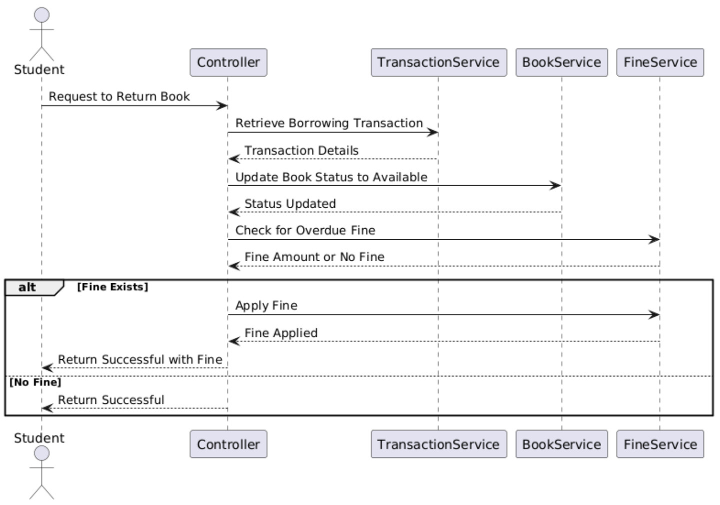

Figure 3: Sequence Diagram: Returning a Book

- Student logs in.
- Student views borrowed books and selects one to return.
- Student clicks ”Return”; system updates book status to available and updates the borrowing record.
- System checks for overdue status; if overdue, calculates and applies a fine, notifying the student.

3. **Adding a Book to Inventory Description**:

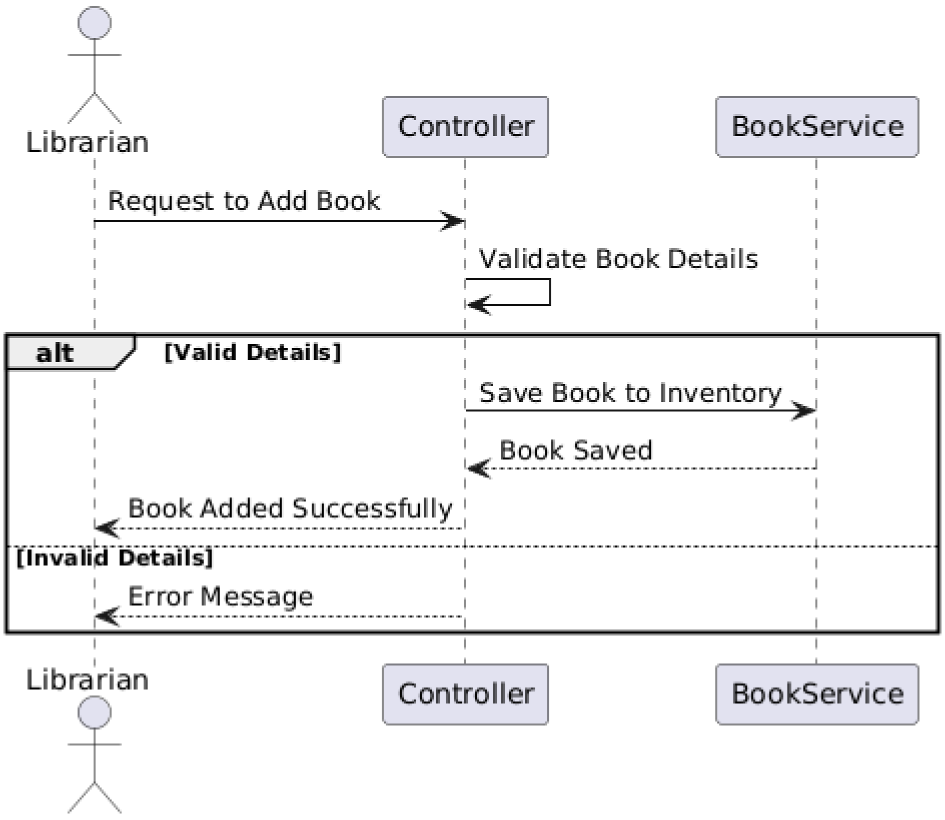

Figure 4: Sequence Diagram: Adding a Book

- Librarian logs in.
- Librarian opens the book management page.
- Librarian enters book details; system validates input.
- System saves the book to the database and confirms addition to the librarian.

3.3 Non-Functional Requirements

3.3.1 Product Requirements

- **Performance**: The system shall support up to 500 concurrent users without performance degradation.
- **Security**: User credentials and transaction data shall be encrypted using industry-standard methods.

- **Availability**: The system shall maintain 99.5% uptime, excluding scheduled maintenance.
- **Usability**: The system shall allow users to complete tasks within three clicks.

3.3.2 Organizational Requirements

- The system shall be developed using Java, Hibernate, Spring Boot, and MySQL database.

3.3.3 External Requirements

- **Scalability**: The system shall support future integration with digital library services.
- **Compliance**: The system shall adhere to data protection regulations for user information.

4 Appendix

4.1 Class Diagram

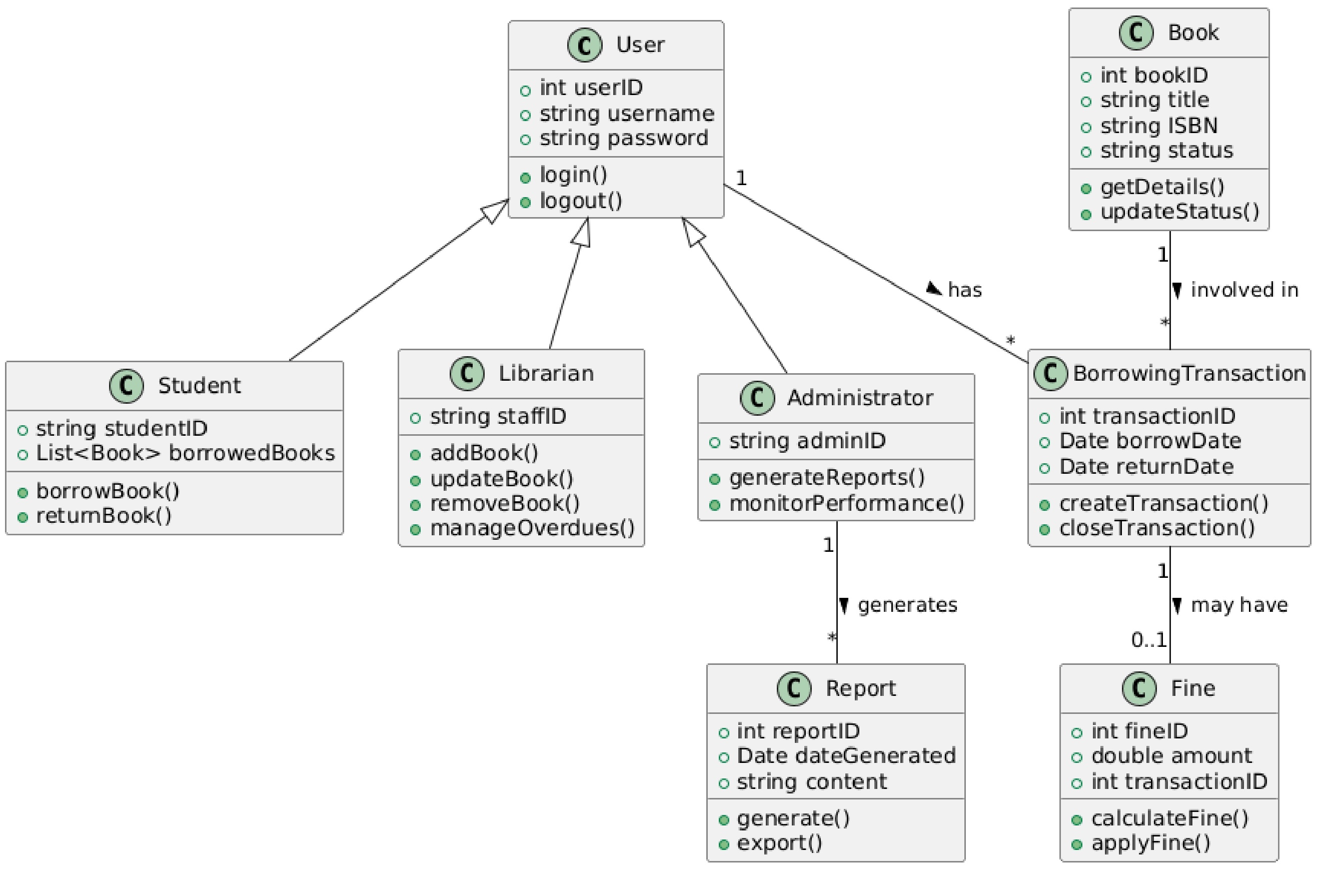

Figure 5: Class Diagram for Library Management System

Description: The class diagram includes the following classes and relationships:

- **User**
  - Attributes: userID, name, email, password, role
  - Methods: login(), logout()
- **Student** (extends User)
  - Attributes: studentID
  - Methods: register(), borrowBook(), returnBook()
- **Librarian** (extends User)
  - Methods: addBook(), updateBook(), removeBook(), manageOverdues()
- **Administrator** (extends User)
  - Methods: generateReport(), monitorPerformance()
- **Book**
  - Attributes: bookID, title, author, genre, ISBN, status
  - Methods: getDetails(), updateStatus()
- **BorrowingTransaction**
  - Attributes: transactionID, userID, bookID, borrowDate, dueDate, returnDate
  - Methods: createTransaction(), updateTransaction()
- **Fine**
  - Attributes: fineID, transactionID, amount, status
  - Methods: calculateFine(), payFine()
- **Report**
  - Attributes: reportID, type, dateGenerated
  - Methods: generateReport()

Relationships:

- User 1—* BorrowingTransaction (one user can have multiple transactions)
- Book 1—* BorrowingTransaction (one book can be part of multiple transactions over time)
- BorrowingTransaction 1—0..1 Fine (a transaction may or may not have a fine)
- Administrator 1—* Report (one administrator can generate multiple reports)