SOFTWARE DESIGN DOCUMENT (SDD)

for

Library Management System

By

Loura Alammour, Fredrik Nyhlén, Muhammad Sufian Tariq
Sukaina Akar and Alfons Krook

Supervisor

Wincent Stålbert Holm

1. Introduction

1.1. Purpose

This System Design Document (SDD) outlines the design and architecture for the Library Management System. The system aims to streamline library operations by providing an efficient platform for administrators and users to manage book related activities, communication, and account information. This document serves as a blueprint for the development, ensuring alignment between functional requirements and system design.

1.2 Scope

The Library Management System is implemented as a web-based application to ensure accessibility across devices and platforms. It caters to modern libraries seeking to provide a self-service experience for users while maintaining effective administrative control. The project context includes:

- The need for a centralized system for managing books, users, and borrowing activities.
- The elimination of traditional paper based records in favor of a more scalable, digital solution.
- Enhanced communication between administrators and users through a message-based chat feature.

This system is particularly beneficial for libraries with a growing user base and diverse book inventories, as it reduces administrative overhead and enhances user experience. The Library Management System facilitates core library operations, providing distinct functionalities for both administrators and users.

1.3 Overview

The Library Management System is built using a modern web development stack, ensuring scalability, reliability, and ease of use. The application architecture is designed to enable seamless interaction between the frontend and backend components while efficiently storing and retrieving data. The frontend employs HTML, CSS, Bootstrap, and JavaScript for a responsive user experience. The backend leverages Java with the Spring Boot framework to manage business logic and provide RESTful APIs for communication. JSON is used for data storage and exchange. For more detailed description and diagrams look at section 2 and 3 of this document.

1.4 Reference Material

- Spring Boot Framework Documentation: https://spring.io/projects/spring-boot
- Bootstrap Framework Documentation: https://getbootstrap.com/
- GitHub Repository: SufianT/demo

2. System overview

The Library Management System is designed to digitize and streamline library operations, enabling both administrators and users to interact with the system seamlessly. It combines a responsive user interface with robust backend functionality to handle essential library processes like book borrowing, returning, fine tracking, and message-based chat communication through a messaging system.

By combining a dynamic web interface with backend automation, the system aims to reduce manual errors, improve operational efficiency, and enhance user satisfaction.

The system focuses on:

1. Managing book inventory and borrowing/returning processes.
2. Providing tools for user account management.
3. Facilitating communication between users and administrators.

2.1 Design

Frontend

- The frontend is developed using **HTML**, **CSS**, **Bootstrap**, and **JavaScript** to deliver a responsive, user-friendly interface.
- It provides interactive pages for user account management, book browsing, borrowing, returning, and fine tracking.
- **JavaScript** is used to dynamically populate book cards on the frontend. Data received from the backend, such as book inventory and user-specific details, triggers JavaScript functions that render and update the UI in real time.
- Bootstrap is used to streamline the design process and ensure a consistent UI/UX experience across devices.

Backend

- The backend is powered by **Java** with the **Spring Boot framework** to handle business logic, API endpoints, and secure communication.
- **RESTful APIs** facilitate communication between the frontend and backend, with the backend processing **JSON** into Java objects and then sending the data to the frontend in a format suitable for rendering.
- Key features like user authentication, book borrowing/return management, fine calculation, and chat services are handled in the backend.
- The backend processes data (e.g., book inventory, borrowing history) by reading JSON into Java objects. It then sends this data to the frontend in a format that triggers JavaScript functions to dynamically render content, such as book cards, on the user interface.

Database

- Data is stored and retrieved in **JSON format** to ensure a flexible and lightweight structure.
- Information such as user accounts, book inventory, borrowing history, fines, and chat logs is efficiently managed by the backend.

2.2 Functionality

- **Administrator Capabilities**:
  - Add books to the library database.
  - Respond to user queries via a message based chat.
  - Monitor borrowed books and overdue returns.
- **User Capabilities**:
  - Create and manage user accounts.
  - Search, borrow, and return books.
  - Track borrowing history, current borrowed books, and return deadlines.
  - View and pay fines for overdue or non-returned books.
  - Access and manage notifications.
  - Communicate with administrators through a message based chat feature.

3. System Architecture

3.1 Architectural Design

The diagram illustrates the high level architectural design of the **Library Management System**, showcasing the modular structure and interconnections between the major subsystems.

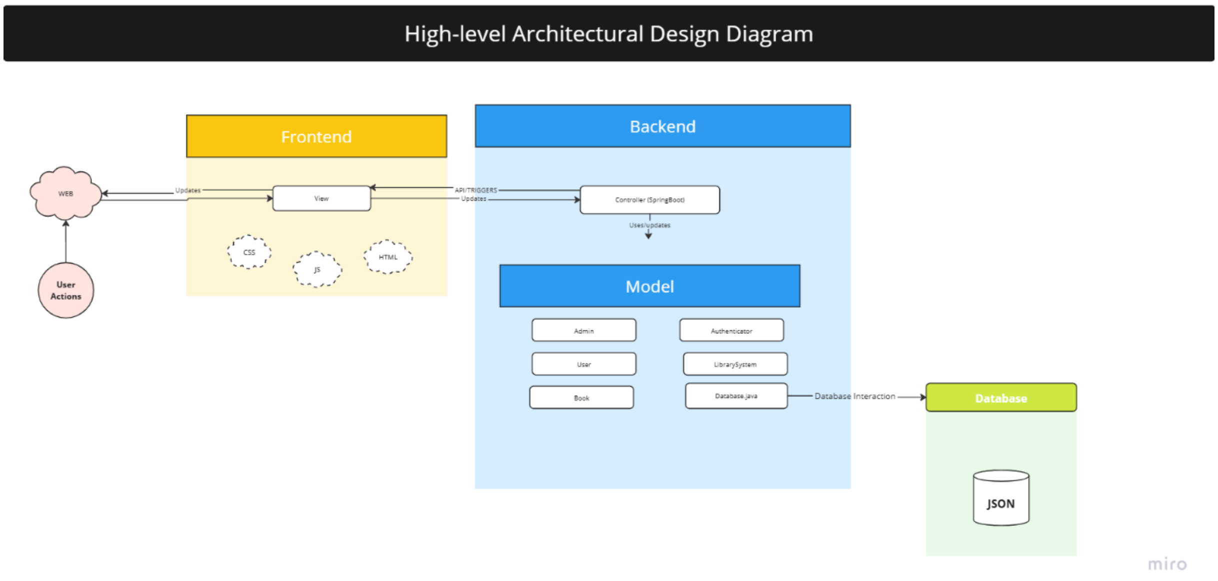
1. **Frontend Subsystem**: This layer represents the user facing interface built using **HTML, CSS, and JavaScript**. It handles user actions (e.g., searching for books, borrowing, or returning books) and communicates with the backend via RESTful API calls, dynamically updating the UI using JavaScript functions.
2. **Backend Subsystem**: Powered by **Spring Boot**, the backend processes user requests, executes business logic, and interacts with the database. It consists of controllers for handling API calls and a model layer comprising key components such as User, Admin, Book, and LibrarySystem.
3. **Database Subsystem**: The database layer stores all system data in a **JSON-based structure** and responds to database queries made by the backend, ensuring efficient data retrieval and updates.

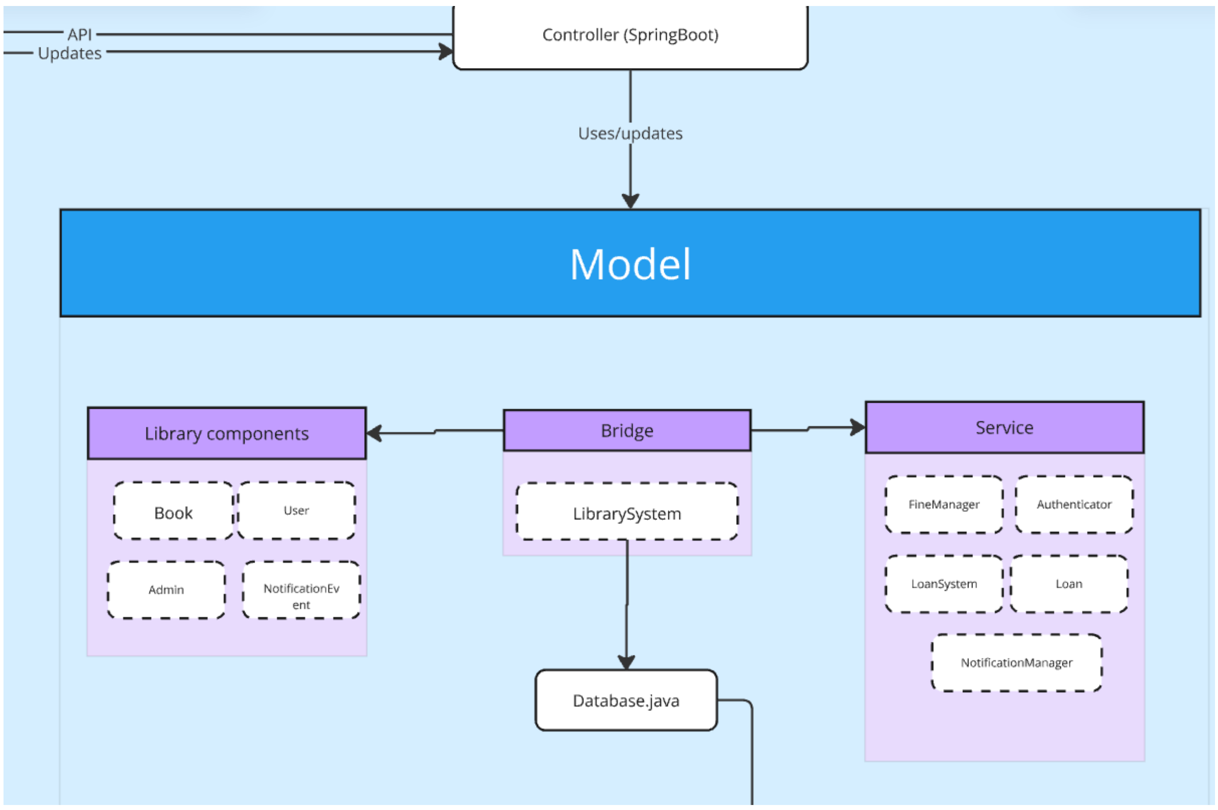
The interconnections between these subsystems are represented by arrows:

- **API calls** connect the frontend to the backend.
- **Database queries** link the backend to the database.

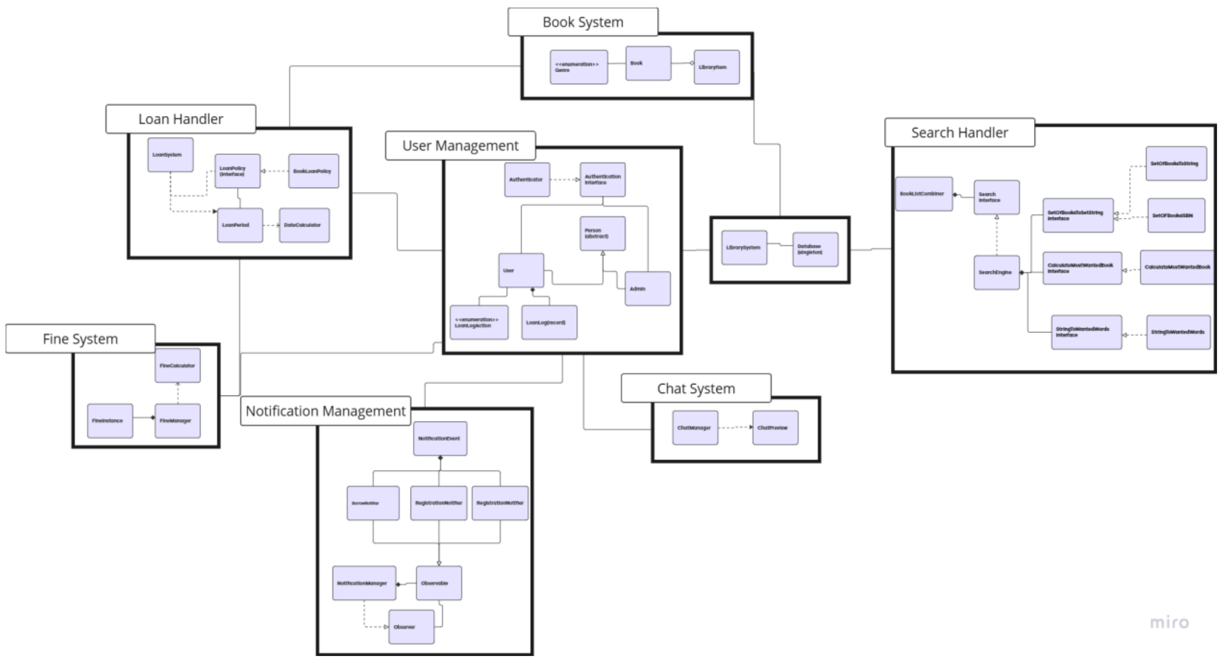
3.2 Decomposition Description

- **Subsystem Interaction Diagram**





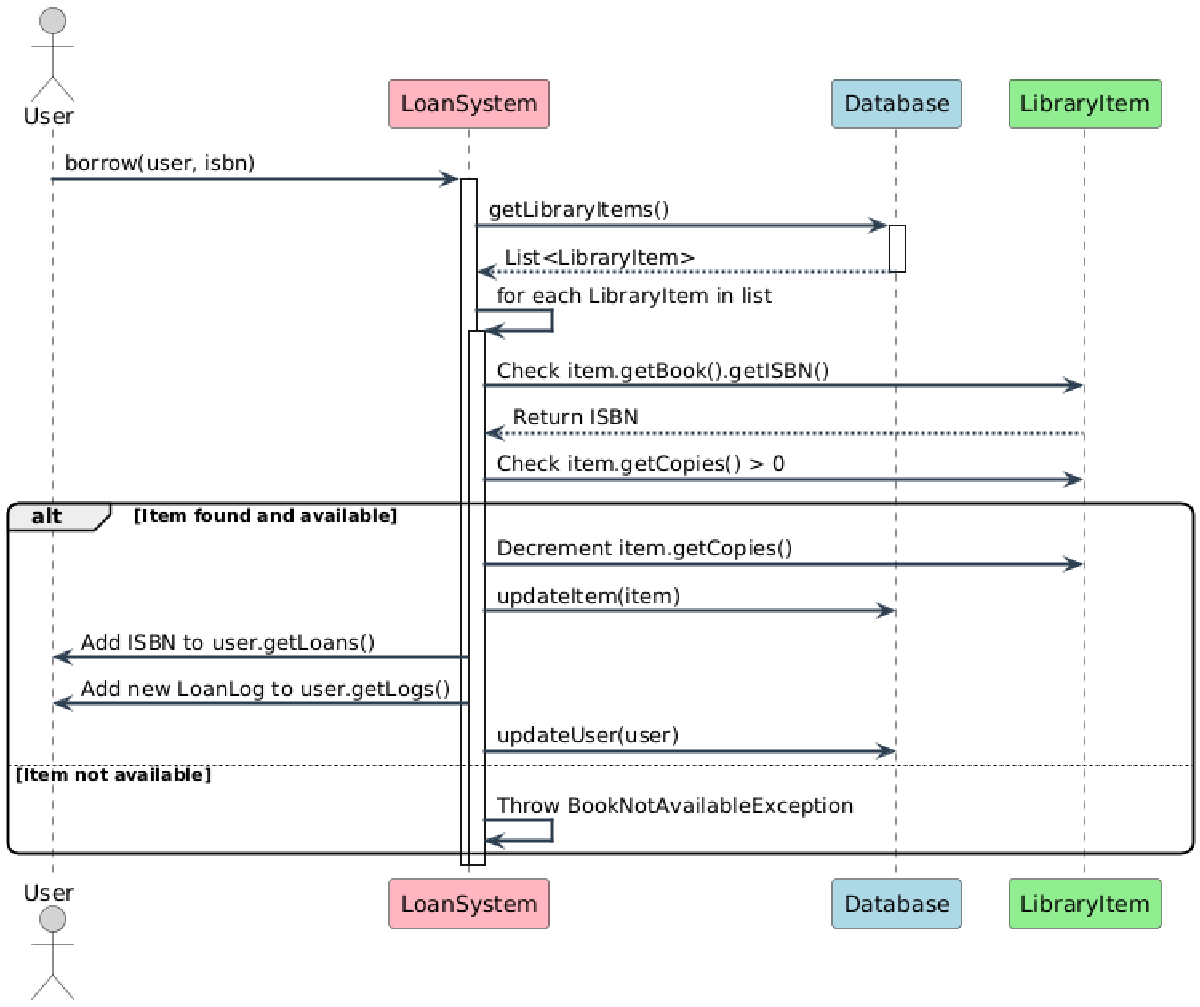
- **Class Diagram**:



Instruction: If the image of the diagram is not clear, please refer to the link provided under the appendix to view the diagram directly in Miro.

- **Sequence Diagram**

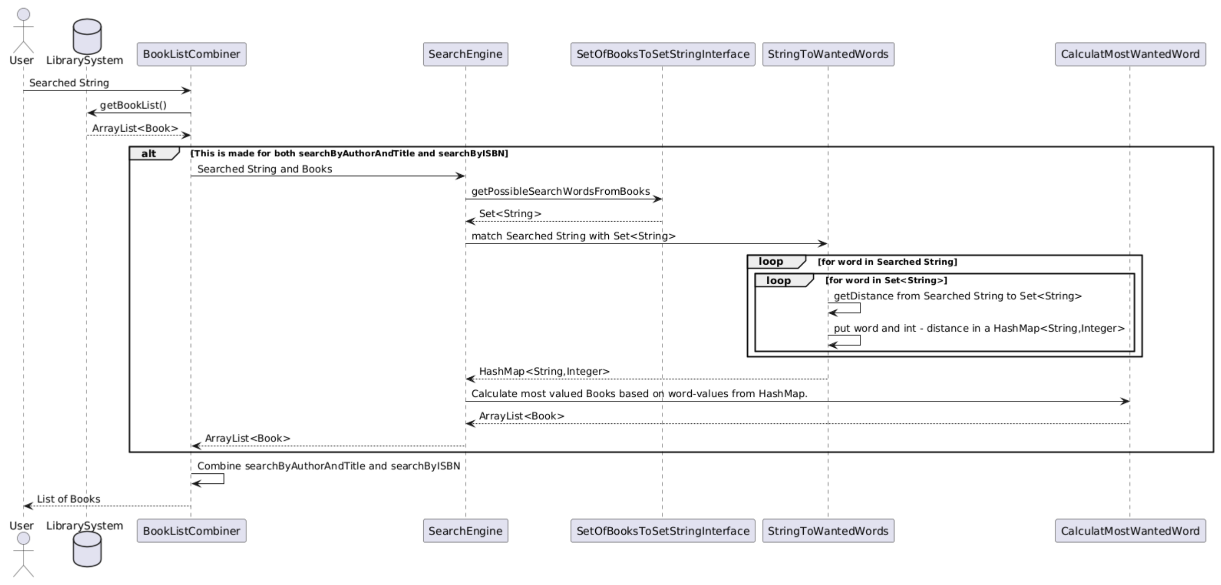
1. LoanSystem Sequence Diagram


User
LoanSystem
Database
LibraryItem
borrow(user, isbn)
getLibraryItems()
List<LibraryItem>
for each LibraryItem in list
Check item.getBook().getISBN()
Return ISBN
Check item.getCopies() > 0
alt
[Item found and available]
Decrement item.getCopies()
updateItem(item)
Add ISBN to user.getLoans()
Add new LoanLog to user.getLogs()
updateUser(user)
[Item not available]
Throw BookNotAvailableException
User
LoanSystem
Database
LibraryItem

*The diagram illustrates the sequence of interactions for the **borrow** method in the **LoanSystem** subsystem. All other methods in the subsystem, such as return, Fine, and similar operations, follow the same logical flow with minor variations.tö*

2. Search Engine Sequence Diagram

Frame 1


User
LibrarySystem
BookListCombiner
SearchEngine
SetOfBooksToSetStringInterface
StringToWantedWords
CalculatMostWantedWord
Searched String
getBookList()
ArrayList<Book>
alt
[This is made for both searchByAuthorAndTitle and searchByISBN]
Searched String and Books
getPossibleSearchWordsFromBooks
Set<String>
match Searched String with Set<String>
loop
[for word in Searched String]
loop
[for word in Set<String>]
getDistance from Searched String to Set<String>
put word and int - distance in a HashMap<String,Integer>
HashMap<String,Integer>
Calculate most valued Books based on word-values from HashMap.
ArrayList<Book>
ArrayList<Book>
Combine searchByAuthorAndTitle and searchByISBN
List of Books

*This diagram shows the **initialization** and **lookup** processes of the SearchEngine subsystem*

4. Human Interface Design

4.1 Overview of User Interface

The Library Management System provides a responsive, web-based interface for two primary user roles: **Users** and **Administrators**. Each role has access to specific features based on their needs. Feedback is provided through dynamic elements like modal pop-ups, notifications, and status updates to enhance usability and accessibility.

a. User Role

Users can access a personalized dashboard to manage their library activities. The interface allows them to:

1. **Account Management**:
   - Create an account or log in using their credentials.
2. **Search and Browse Books**:
   - Use a search bar to filter the book inventory by title, author, or ISBN.
   - View book details (title, author, genre, and availability) in a visually appealing format.
3. **Borrow and Return Books**:
   - Borrow available books and receive feedback (e.g., success notifications or unavailability messages).
   - Track current borrowed books, return deadlines, and borrowing history directly from the dashboard.
   - Return books through the dashboard.
4. **Fines and Notifications**:
   - Receive reminders when the return deadline for borrowed books is approaching.
   - View fines for overdue or non-returned books in the "Fine Due" section.
5. **Messaging with Administrators**:
   - Send queries or issues to administrators via a message-based chat interface, and view responses in the message panel.

b. Administrator Role

Administrators access the system via a secure login, requiring their credentials and a unique, pre-assigned **adminKey**. Administrators cannot create accounts via the web interface; instead, new administrator accounts are manually added by authorized personnel through the backend or database.

Once logged in, administrators can:

1. **Manage Book Inventory**:
   - Add books from the library system.
2. **Respond to User Queries**:
   - Address user questions or concerns through the message-based chat interface.

5. Appendices

UML: UML Diagram - Miro