CSE 3105 / CSE 3137

OBJECT-ORIENTED ANALYSIS AND DESIGN

FALL 2024

COURSE PROJECT: *Library Management System (LMS)*

Requirements Analysis Document

Group 5

Berke Alpaslan – 210316016

Mert Doğan Aygün – 210316076

Gül Yeşilgil – 220316055

Birhat Taş – 190315010

Tural Mammadov – 190316001

Aziz Berk Yıldırım - 210316055

10 December 2024

Table of Contents

Table of Figures

1 Introduction

The Library Management System (LMS) is designed to streamline and modernize the university library’s operations, providing students and staff with an efficient way to manage library resources. This system aims to address the growing demand for a digital solution that enables users to borrow, return, and search for books with ease, while also supporting library administrators in tracking inventory and managing user accounts within a user-friendly, automated environment.

The LMS will incorporate essential functions such as book borrowing and return management, book search and reservation capabilities, real-time inventory tracking, and user notifications for overdue items. Additionally, the system will offer administrators tools for managing the collection, generating reports, and overseeing user activities. Through these features, LMS is expected to significantly enhance accuracy, efficiency, and overall user satisfaction within the library.

Aligned with the university’s objective to improve digital services and make library resources more accessible, the LMS project will draw insights from similar systems in other academic institutions that have demonstrated notable improvements in operational efficiency and user satisfaction. By adapting these best practices to our university's specific needs, the LMS aims to provide a tailored and effective solution.

The primary objective of the LMS is to deliver an intuitive, reliable, and user-friendly system that elevates the library experience for students, faculty, and staff. The success of this project will be measured by its ability to handle book checkouts and returns efficiently, achieve high user adoption and satisfaction rates, reduce the time library staff spend managing inventory and accounts, and increase the timely return of books through regular, automated notifications.

2 Current System

The university library currently operates on a manual system where book borrowing, returns, and inventory tracking depend on physical records. This process is time-consuming and prone to errors, with no automated notifications for overdue books, resulting in delays. The new Library Management System (LMS) will digitize these tasks, enhancing efficiency, accuracy, and accessibility for all users.

3 Proposed System

The proposed system aims to replace the current manual and inefficient processes with a fully automated Library Management System. This system will allow users to search, borrow, return, and reserve books digitally. Key functionalities include real-time updates on book availability, automatic tracking of due dates, and notifications for reservations. The system will streamline library operations by reducing errors and processing times, improving the user experience, and enabling better management of library resources. The following subsections will detail the system’s requirements and analysis model.

3.1 Functional Requirements

- **Book Borrowing and Returning**
    - The system must allow users to borrow books based on their membership type and borrowing limits.
    - Users should be able to return borrowed books and receive a confirmation of the return.

- **Book Reservation**
    - Users must be able to reserve books that are currently unavailable.
    - Reserved books should be held for a specified period after they become available, and the user must be notified.

- **Inventory Management**
    - Librarians should be able to add, update, and remove book records from the inventory.
    - The system must keep track of available stock for each book.

- **Low Stock Alerts**
    - The system must send notifications to librarians when a book's stock is below a defined threshold.

- **Duplicate Book Prevention**
    - The system must prevent duplicate entries when adding new books to the inventory.

- **User Notifications**
    - The system must notify users about reservation availability, overdue books, and applied fines.

- **Membership Management**
    - The system must allow librarians to register new users, update user information, and deactivate memberships when necessary.

3.2 Nonfunctional Requirements

- **Usability**
  - The system interface must be intuitive and user-friendly, enabling users and librarians to perform operations with minimal training.
  - Support for multiple device types, including desktops and tablets, should be provided.
- **Reliability**
  - The system should have a 99.9% uptime to ensure availability during library operating hours.
  - Data integrity must be maintained, with proper backups to prevent data loss.
- **Performance**
  - The system must handle up to 500 concurrent users without significant performance degradation.
  - Responses to user actions (e.g., searching for a book) should be completed within 2 seconds.
- **Supportability**
  - The system must be designed with modular architecture to allow for future enhancements.
  - Technical support documentation should be available for system administrators.
- **Implementation**
  - The system must be developed using scalable technologies, ensuring adaptability to growing user bases.
  - Compliance with industry-standard security practices (e.g., encrypted data storage) is mandatory.
- **Interface**
  - The system must integrate with email services to send notifications.
  - APIs should be provided to enable integration with external systems, such as e-payment gateways for fine collection.
- **Operational**
  - The system must support multilingual options to accommodate a diverse user base.
  - Regular maintenance windows must be scheduled and communicated to users.
- **Legal**
  - The system must comply with data protection regulations such as GDPR for user data privacy.

3.3 System Models

3.3.1 Scenarios

a. **Main Scenario: Book Borrowing and Returning Process**

Participating Actors:

- **Joe**: Librarian
- **Alice**: Library Member

Flow of Events:

1. Alice logs into the library management system to either borrow a new book or return one that she previously borrowed.
2. She searches for a book by title, author, or genre.
3. The system displays the search results, including book details, availability, and location within the library.
4. Once Alice finds an available book, she selects it and initiates a borrowing request.
5. The system validates Alice's request, updates the book status to "borrowed," and links it to her account.
6. The system sets a due date for the book and sends a confirmation to Alice, including the return date.
7. When returning a book, Alice logs back in, selects "return," and confirms the action.
8. The system updates the book's status to "available" and removes it from Alice's active loans.

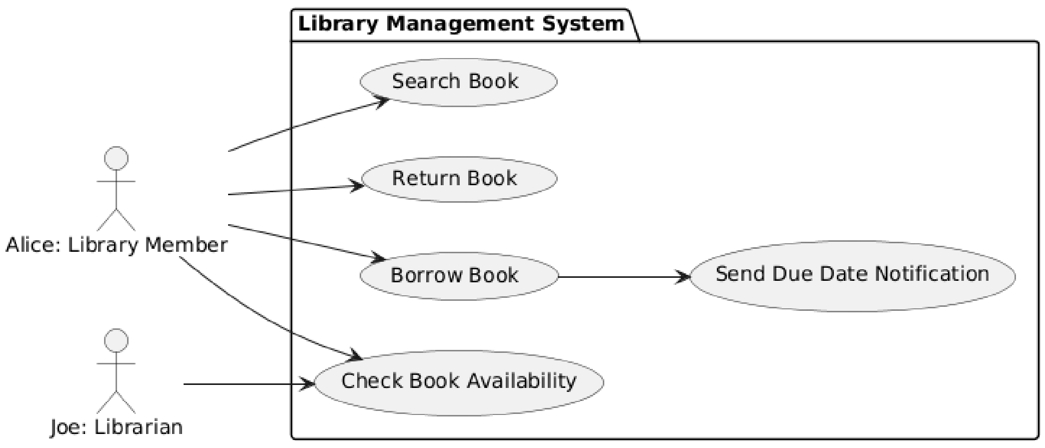


Figure 1 - Refining Main Scenario: Book Borrowing and Returning Process

b. **Side Scenario 1: Book Reservation Process**

Participating Actors:

- **Donald**: Library Member
- **Emma**: Library Admin

Flow of Events:

1. Donald logs into the system and searches for a book he would like to borrow.
2. The system shows that the book is currently borrowed.
3. Donald selects the reservation option, creating a request to reserve the book.
4. The system confirms the reservation, notifying Donald that he will receive an alert when the book becomes available.
5. Once the book is returned, the system sends a notification to Donald, indicating it is ready for pickup.
6. Donald has a limited time to borrow the reserved book before it becomes available to other members.

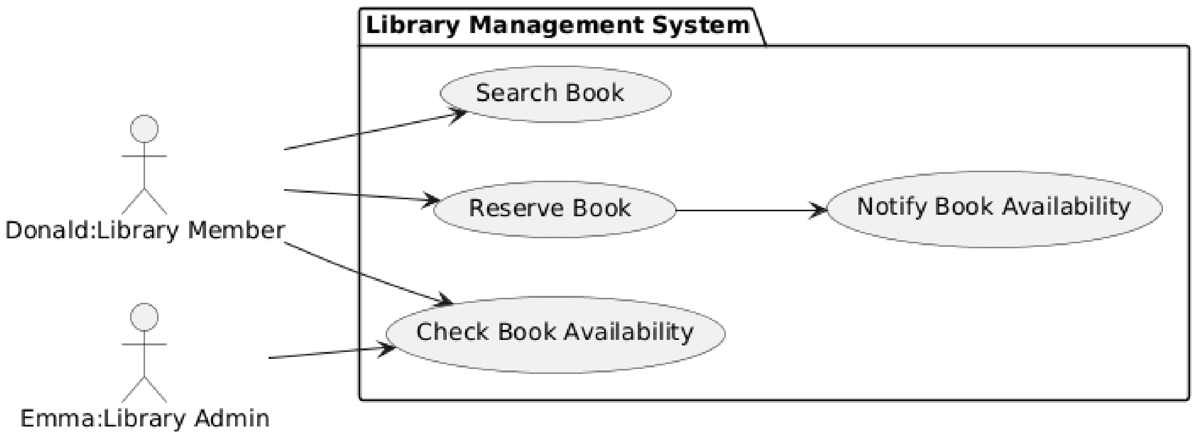


Figure 2- Refining Side Scenario 1: Book Reservation Process

c. **Side Scenario 2: Library Inventory and Stock Management**

Participating Actors:

- **Sarah**: Library Admin
- **Mike**: Librarian

Flow of Events:

1. Sarah, the library admin, logs into the system to check inventory and track low-stock items.
2. Sarah reviews book statuses (borrowed, reserved, or available).
3. When adding a new book to the system, Sarah inputs its details (title, author, publisher, year).
4. The system stores the new book in the inventory and makes it available for member search and borrowing.
5. The system also generates alerts for any book that falls below the designated stock level, prompting Mike to initiate restocking.

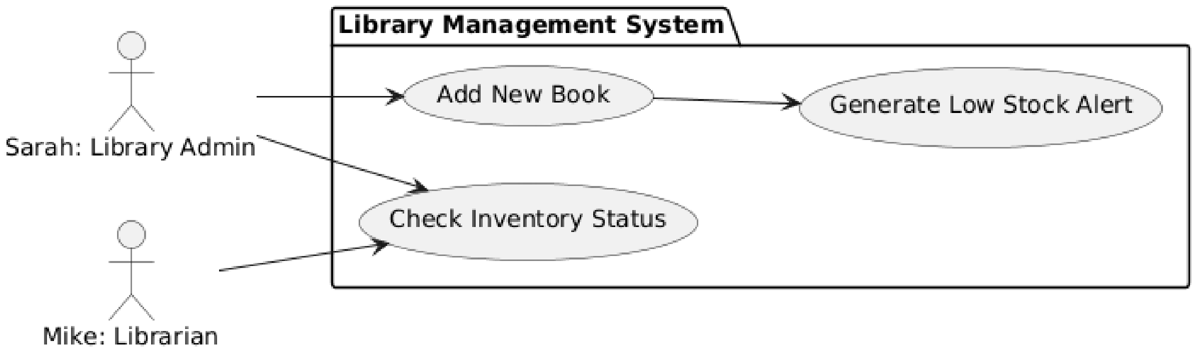


Figure 3 - Refining Side Scenario 2: Library Inventory and Stock Management

3.3.2 Use Case Model

a) Main Scenario: Book Borrowing and Returning Process

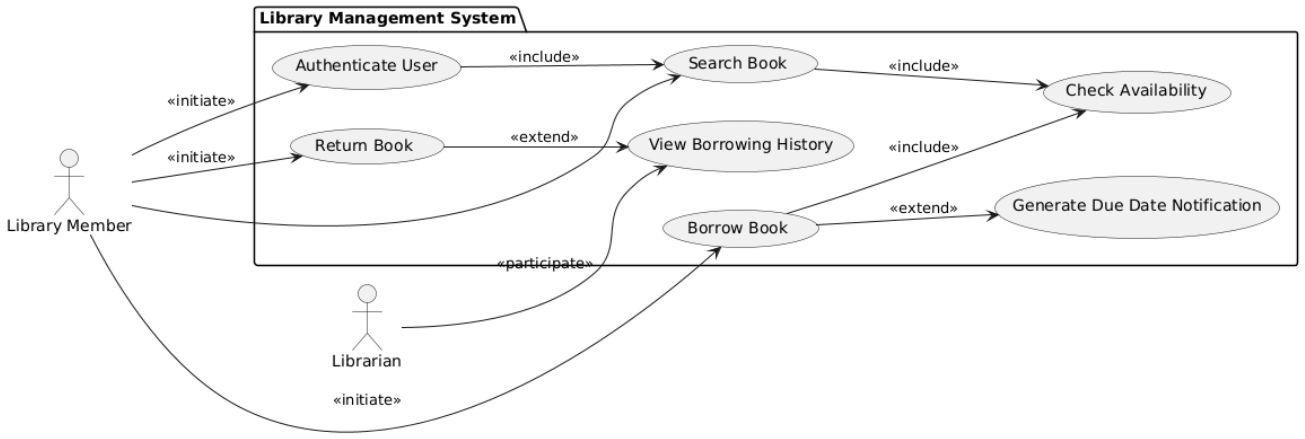


Figure 4 - Use-Case Model of Main Scenario: Book Borrowing and Returning Process

- **Participating Actors**

  Initiated by: Library Member

  Communicates with: Librarian

- **Flow of Events**

  1. Library Member logs into the system.
  2. The system authenticates the user.
  3. Member searches for a book by title, author, or genre.
  4. The system displays search results, including book availability and location.
  5. Member requests to borrow a book.
  6. The system checks book availability.
  7. If the book is available, the system processes the borrowing request and updates the book status to "borrowed."
  8. The system sets a due date and sends a notification to the member.
  9. For returning a book, the member logs into the system, selects the "return" option, and confirms the action.
  10. The system updates the book status to "available" and removes it from the member's active loans.

- **Entry Condition**

  The member is logged into the system and has access to the library catalog.

- **Exit Condition**

  The book status is successfully updated to "borrowed" or "available," and notifications (if applicable) have been sent.

b) Side Scenario 1: Book Reservation Process

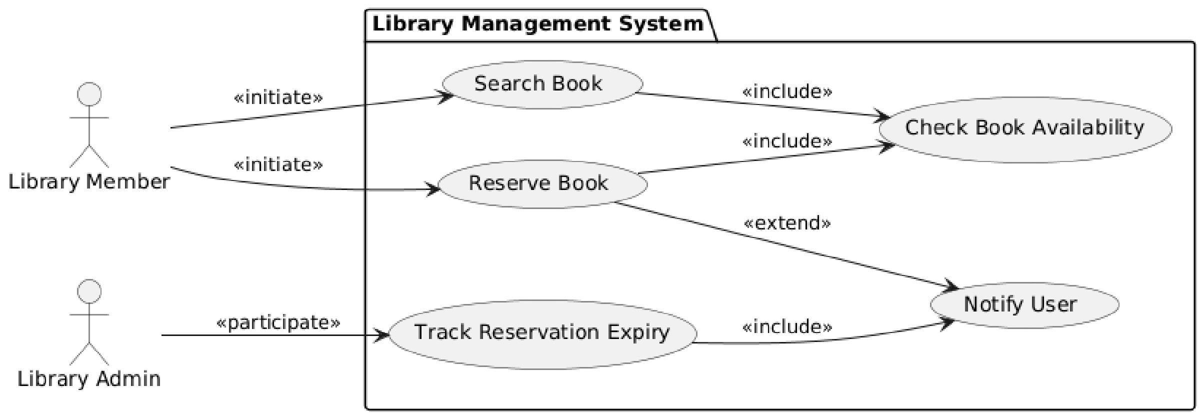


Figure 5 - Use-Case Model of Side Scenario 1: Book Reservation Process

- **Participating Actors**

  Initiated by: Library Member

  Communicates with: Library Admin

- **Flow of Events**

  1. Library Member logs into the system and searches for a desired book.
  2. The system checks the availability of the book.
  3. If the book is unavailable, the member initiates a reservation request.
  4. The system processes the reservation and sends a confirmation to the member.
  5. When the book is returned, the system notifies the member about its availability.
  6. The member has a limited timeframe to borrow the reserved book.

- **Entry Condition**

  The member is logged into the system and has located an unavailable book in the catalog.

- **Exit Condition**

  The book is reserved for the member, and notifications have been sent, or the reservation is canceled if not claimed within the allocated time.

c) Side Scenario 2: Library Inventory and Stock Management

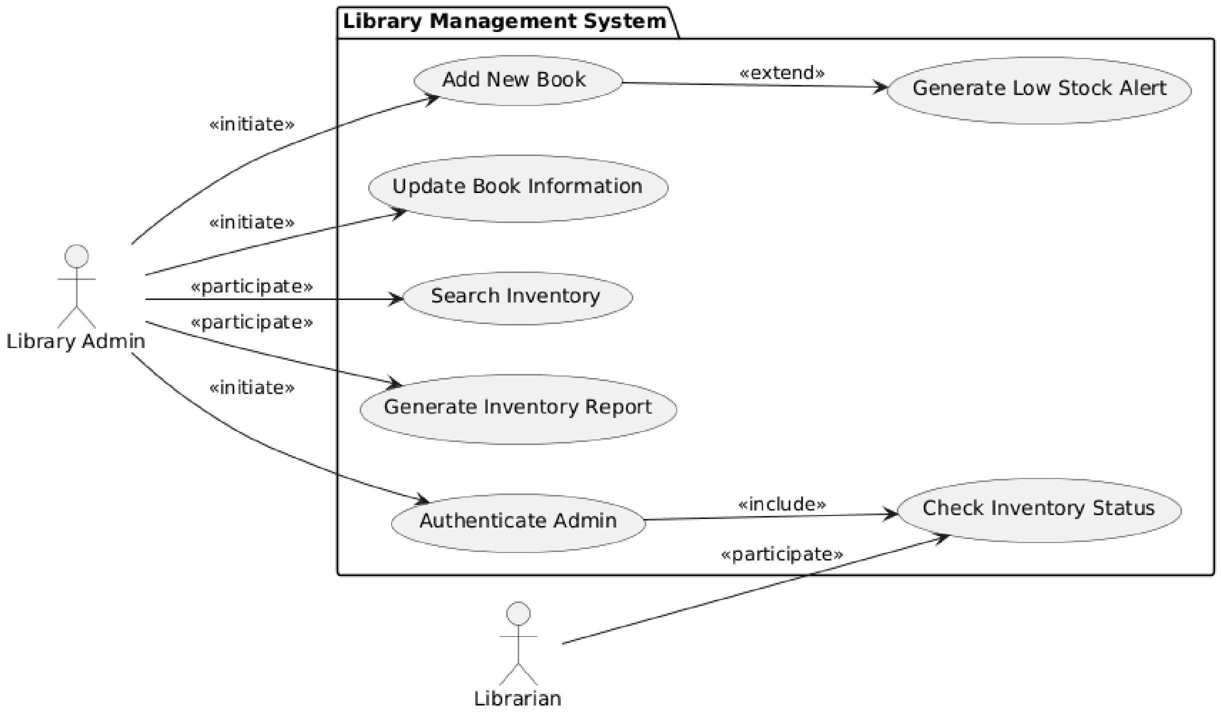


Figure 6 - Use-Case Model of Side Scenario 2: Library Inventory and Stock Management

- **Participating Actors**

  Initiated by: Library Admin

  Communicates with: Librarian

- **Flow of Events**

  1. Library Admin logs into the system.
  2. The system authenticates the admin.
  3. Admin views the current inventory status, including borrowed, reserved, and available books.
  4. Admin adds a new book to the inventory by entering its details (e.g., title, author, publisher, year).
  5. The system validates and stores the new book details in the database.
  6. The system generates low-stock alerts if any book's stock falls below a set threshold.
  7. Admin updates the book's information if needed (e.g., correcting metadata or adding additional copies).

- **Entry Condition**

  The admin is logged into the system and has permissions for inventory management.

- **Exit Condition**

  The inventory is updated with new or modified book records, and alerts for low-stock books are visible in the admin interface.

3.3.3 Object Model

I. Main Scenario: Book Borrowing and Returning Process

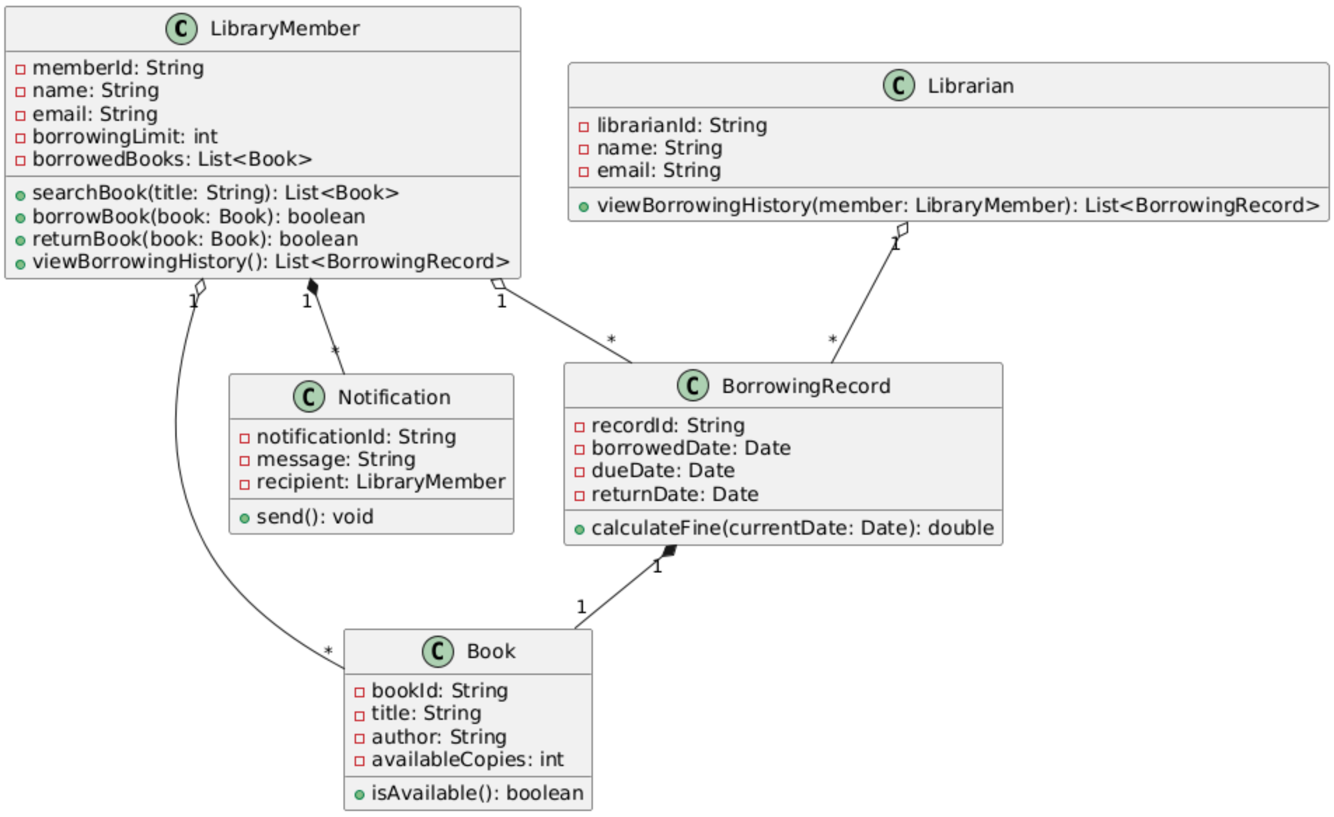


Figure 7 - Object Model of Main Scenario: Book Borrowing and Returning Process

1. **LibraryMember**
   - **Attributes**: memberId, name, email, borrowingLimit, borrowedBooks
   - **Operations**:
     - **searchBook(title: String): List<Book>:** Searches for books by title.
     - **borrowBook(book: Book): boolean:** Borrows an available book if the borrowing limit is not exceeded.
     - **returnBook(book: Book): boolean:** Returns a borrowed book.
     - **viewBorrowingHistory(): List<BorrowingRecord>:** Displays the borrowing history.

- **Relationships**:
  - **Aggregation with BorrowingRecord:** A member has a list of borrowing records.
  - **Aggregation with Book:** A member borrows multiple books.

2. **Librarian**
   - **Attributes**: librarianId, name, email
   - **Operations**:
     - **viewBorrowingHistory(member:LibraryMember): List<BorrowingRecord>:** Views the borrowing history of a specific member.
   - **Relationships**:
     - **Aggregation with BorrowingRecord:** The librarian manages borrowing records.
3. **Book**
   - **Attributes**: bookId, title, author, availableCopies
   - **Operations**:
     - **isAvailable(): boolean:** Checks the availability of the book.
   - **Relationships**:
     - **Composition with BorrowingRecord:** A borrowing record must reference a specific book.
4. **BorrowingRecord**
   - **Attributes**: recordId, borrowedDate, dueDate, returnDate
   - **Operations**:
     - **calculateFine(currentDate: Date): double:** Calculates the fine based on the current date.
   - **Relationships**:
     - **Composition with Book:** A record is always linked to a book.
5. **Notification**
   - **Attributes**: notificationId, message, recipient
   - **Operations**:
     - **send(): void:** Sends notifications to the library member.

- **Relationships**:
    - **Aggregation with LibraryMember:** Notifications are linked to members.

- **Entity Objects:**
    - **Library Member**
    - **Librarian**
    - **Notification**
    - **BorrowingRecord**
    - **Book**

4 Textual Explanation:

1. **Book:** This class represents a book in the library with attributes for title, author, and availability. It manages borrowing and returning operations.
2. **User:** Represents a library member with methods for authentication and managing their borrowing history.
3. **Notification:** Responsible for notifying users about due dates.

II. Side Scenario 1: Book Reservation Process

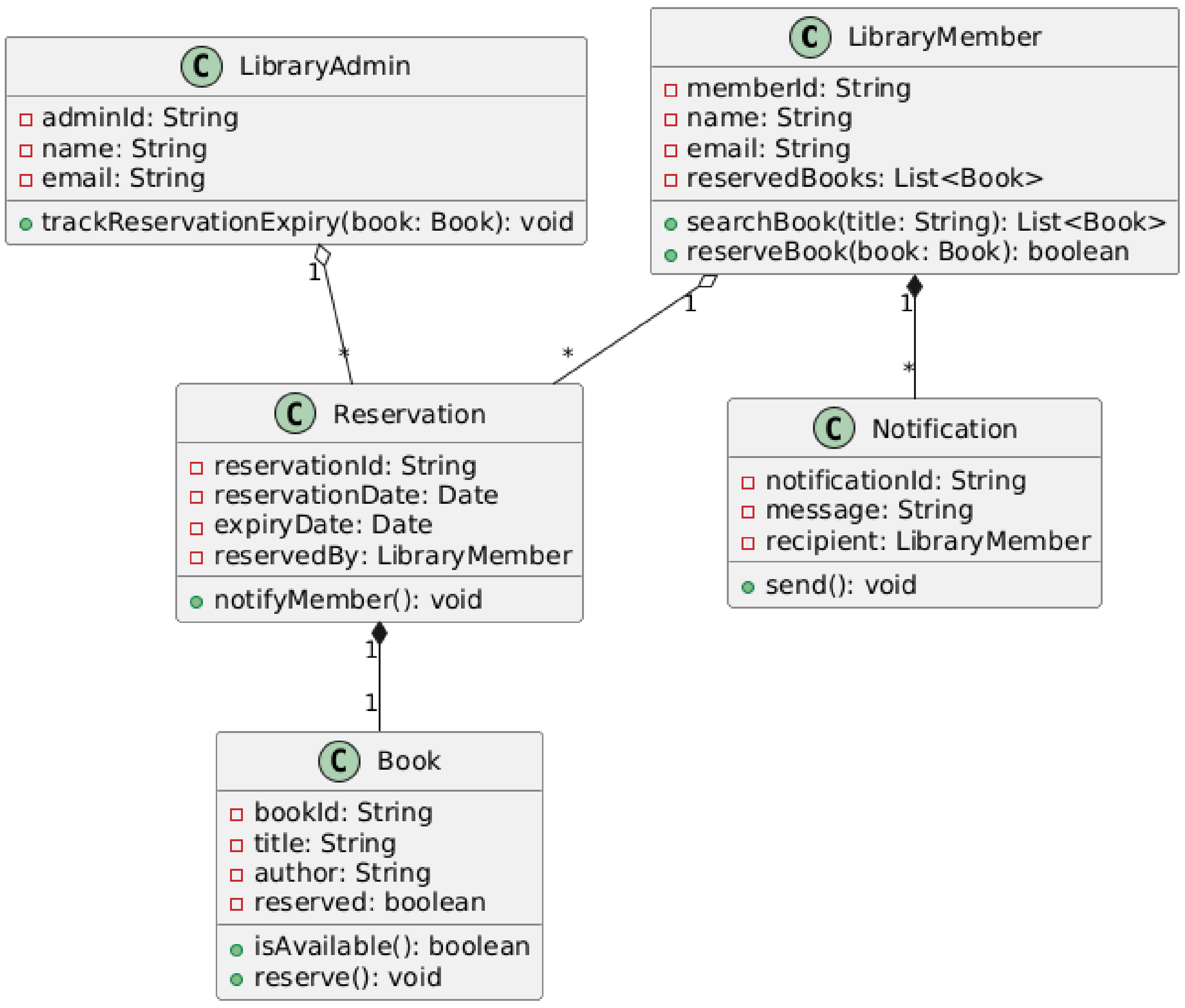


Figure 8 - Object Model of Side Scenario 1: Book Reservation Process

1. **LibraryMember**
   - **Attributes:** memberId, name, email, reservedBook
   - **Additional Relationship:**
     - Aggregation with Reservation: A member can have multiple reservations.
2. **LibraryAdmin**
   - **Attributes:** adminId, name, email
   - **Operations:**
     - **trackReservationExpiry(book: Book): void:** Tracks expired reservations and takes action.

3. **Book**
    - **Attributes:** bookId, title, author
    - **Additional Attribute: reserved:** boolean to track reservation status.
    - **Additional Operation:**
        - **reserve(): void:** Marks the book as reserved.
4. **Reservation**
    - **Attributes:** reservationId, reservationDate, expiryDate, reservedBy
    - **Operations:**
        - **notifyMember(): void:** Notifies the member about reservation updates.
    - **Relationships:**
        - **Composition with Book:** A reservation must reference a book.
5. **Notification**
    - **Attributes:** notificationId, message, recipient

- **Entity Objects**
    - **LibraryAdmin**
    - **LibraryMember**
    - **Reservation**
    - **Notification**
    - **Book**

5 Textual Explanation:

1. **Reservation:** Manages the reservation process, tracking expiry and associated books.
2. **Notification:** Sends alerts to the user about reservation expiry or updates.

III. Side Scenario 2: Library Inventory and Stock Management

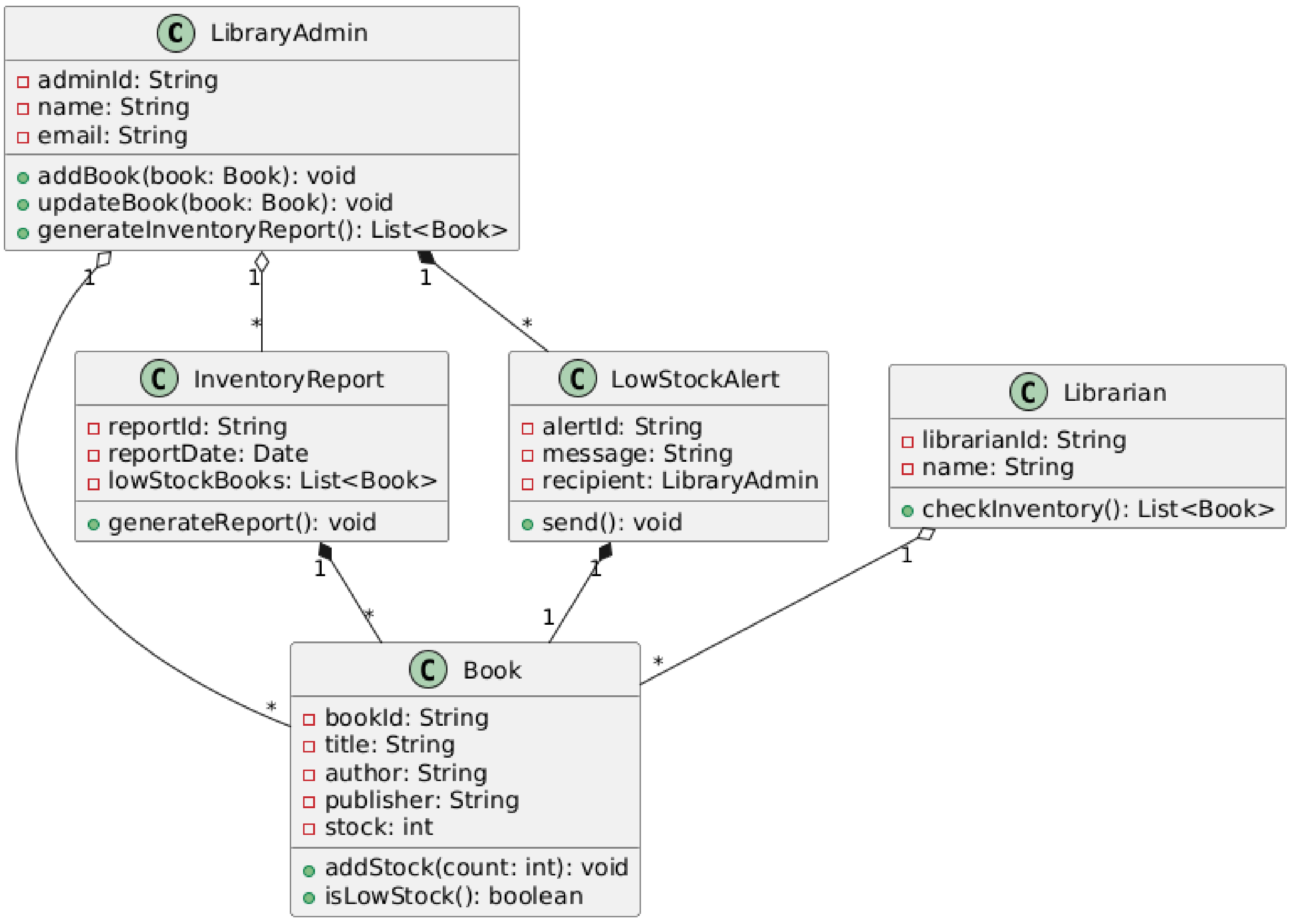


Figure 9 - Object Model of Side Scenario 2: Library Inventory and Stock Management

1. **LibraryAdmin**
    - **Attributes:** adminId, name, email
    - **Additional Operation**:
        - **generateInventoryReport(): List<Book>:** Generates an inventory report.
2. **Librarian**
    - **Attributes:** librarianId, name
    - **Additional Operation:**
        - **checkInventory(): List<Book>:** Checks book inventory
3. **Book**
    - **Attributes:** bookId, title, author

- **Additional Attributes**: publisher, stock
- **Additional Operations**:
    - **addStock(count: int):** void: Increases stock by the given count.
    - **isLowStock(): boolean:** Checks if the book stock is below a threshold.

4. **InventoryReport**
    - **Attributes**: reportId, reportDate, lowStockBooks
    - **Operations**:
        - generateReport(): void: Generates a list of low-stock books.
5. **LowStockAlert**
    - **Attributes**: alertId, message, recipient
    - **Operations**:
        - send(): void: Sends a low-stock alert to the admin.

- **Entity Objects**
    - LibraryAdmin
    - InventoryReport
    - LowStockAlert
    - Librarian
    - Book

6 Textual Explanation:

1. **Inventory**: Tracks stock levels and generates alerts for low stock.
2. **Book**: Represents each individual book, with attributes for title, author, and stock details.

6.1.1 Dynamic Models

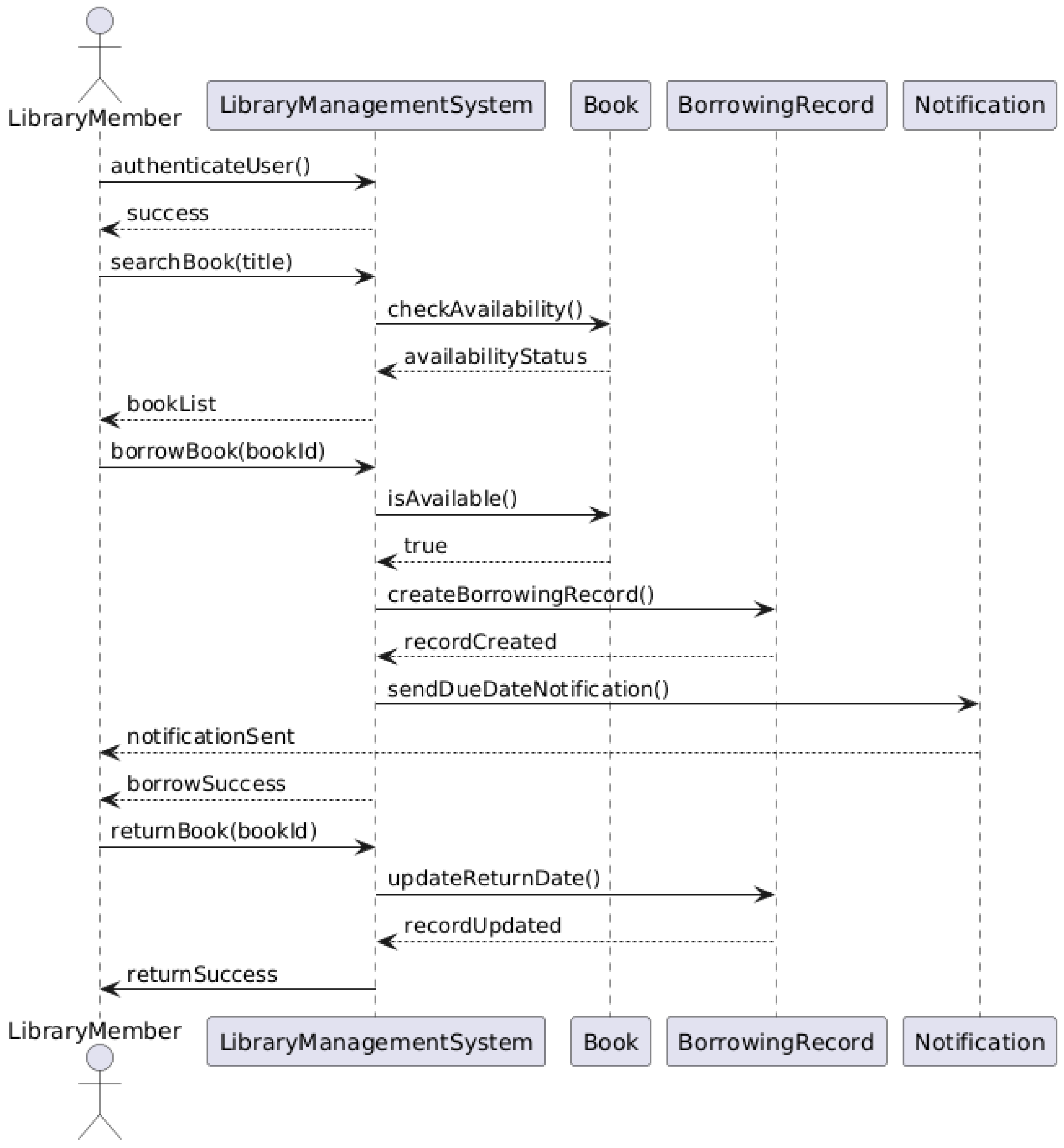


Figure 10 - Dynamic (Sequence) Model of Main Scenario: Book Borrowing and Returning Process

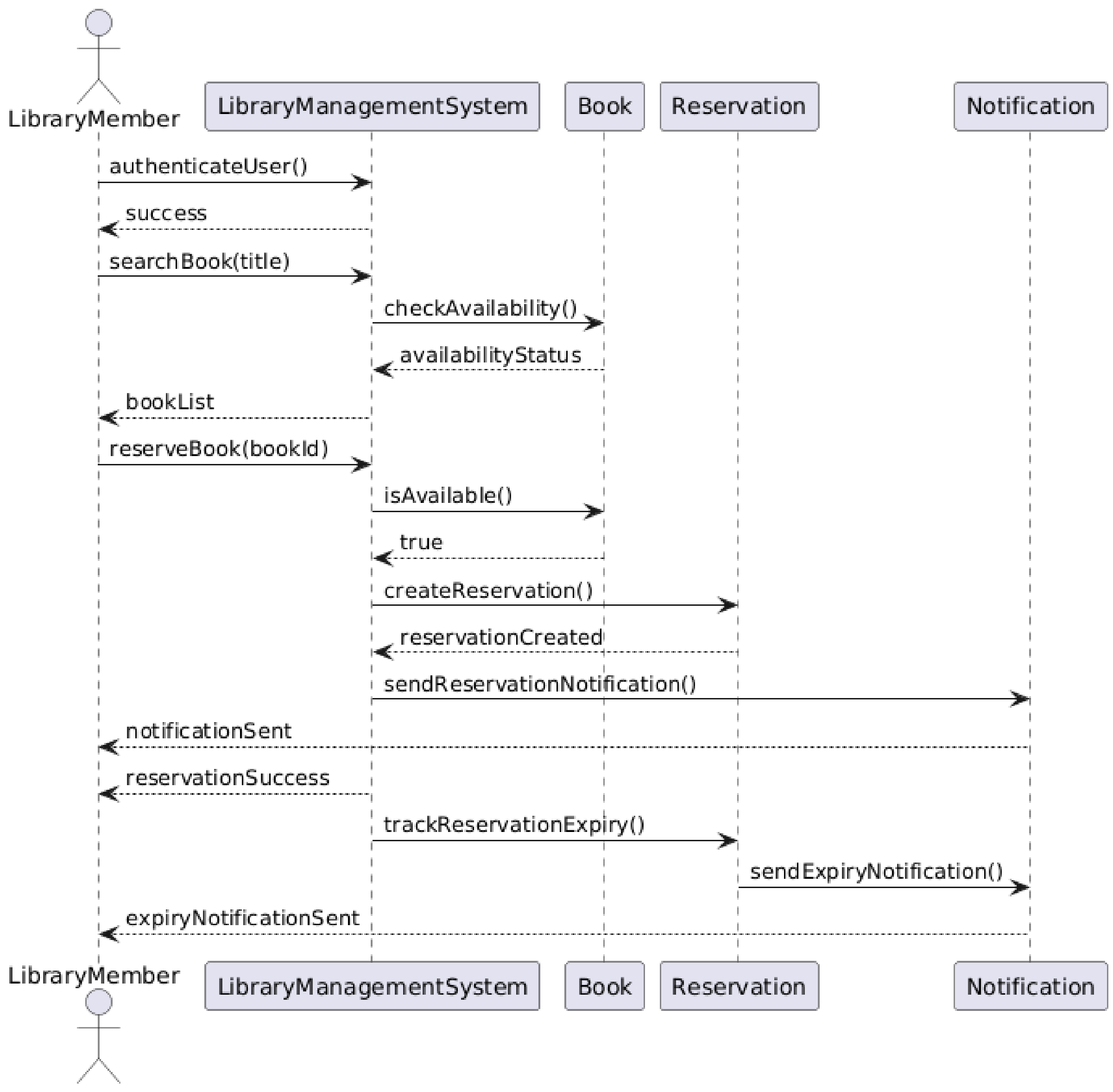


Figure 11 - Dynamic (Sequence) Model of Side Scenario 1: Book Reservation Process

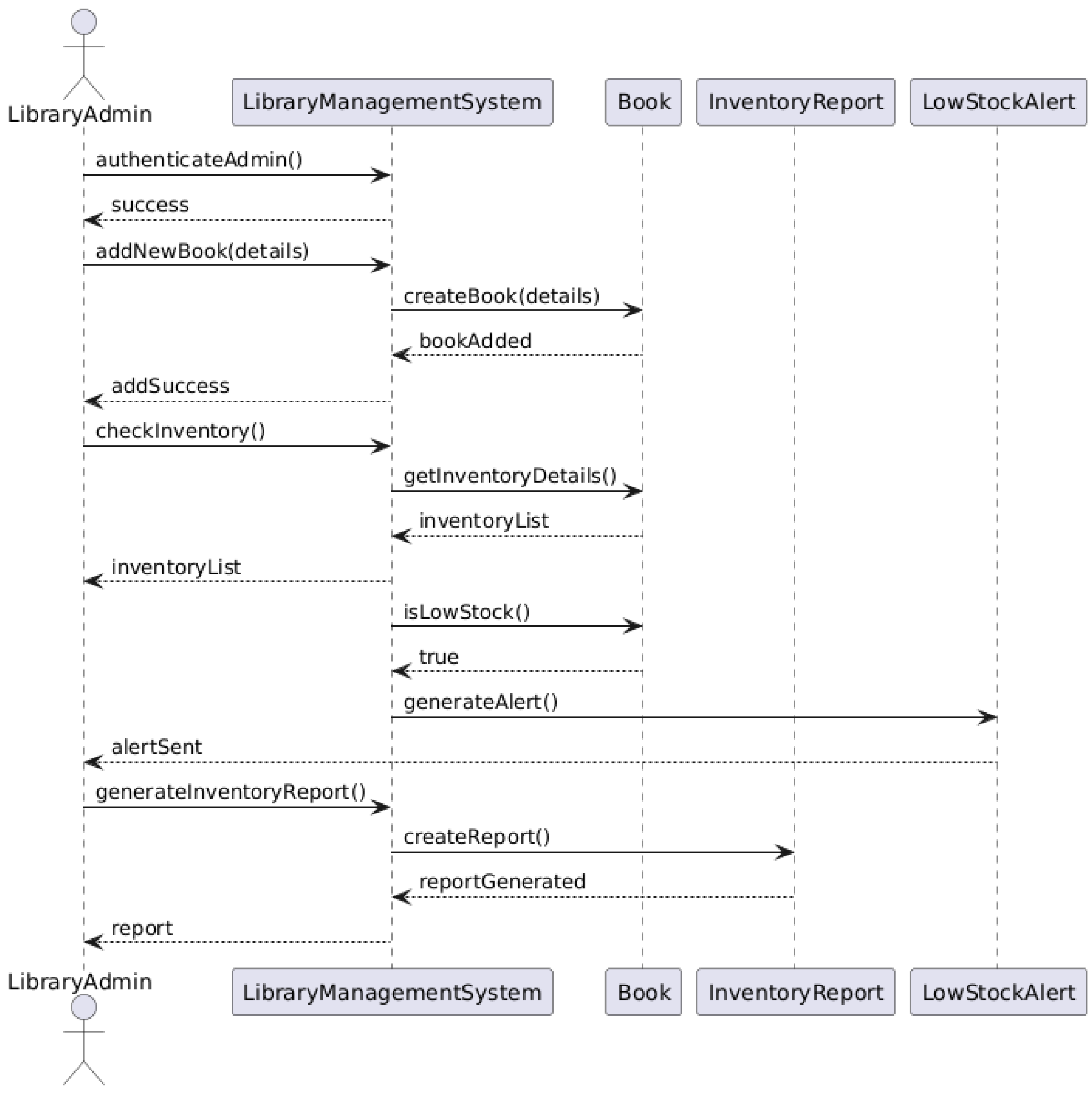


Figure 12 - Dynamic (Sequence) Model of Side Scenario 2: Library Inventory and Stock Management

6.1.2 User Interface Mock-ups

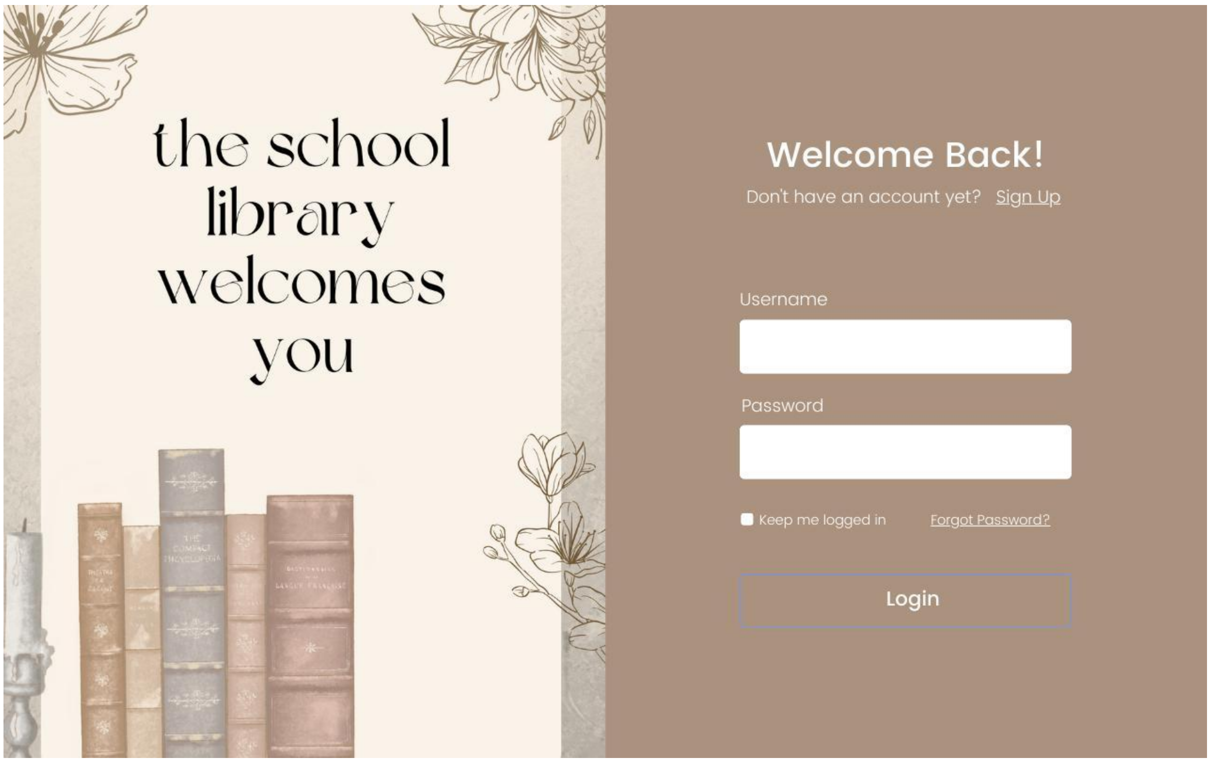


Figure 13 - User Login UI Mock-Up

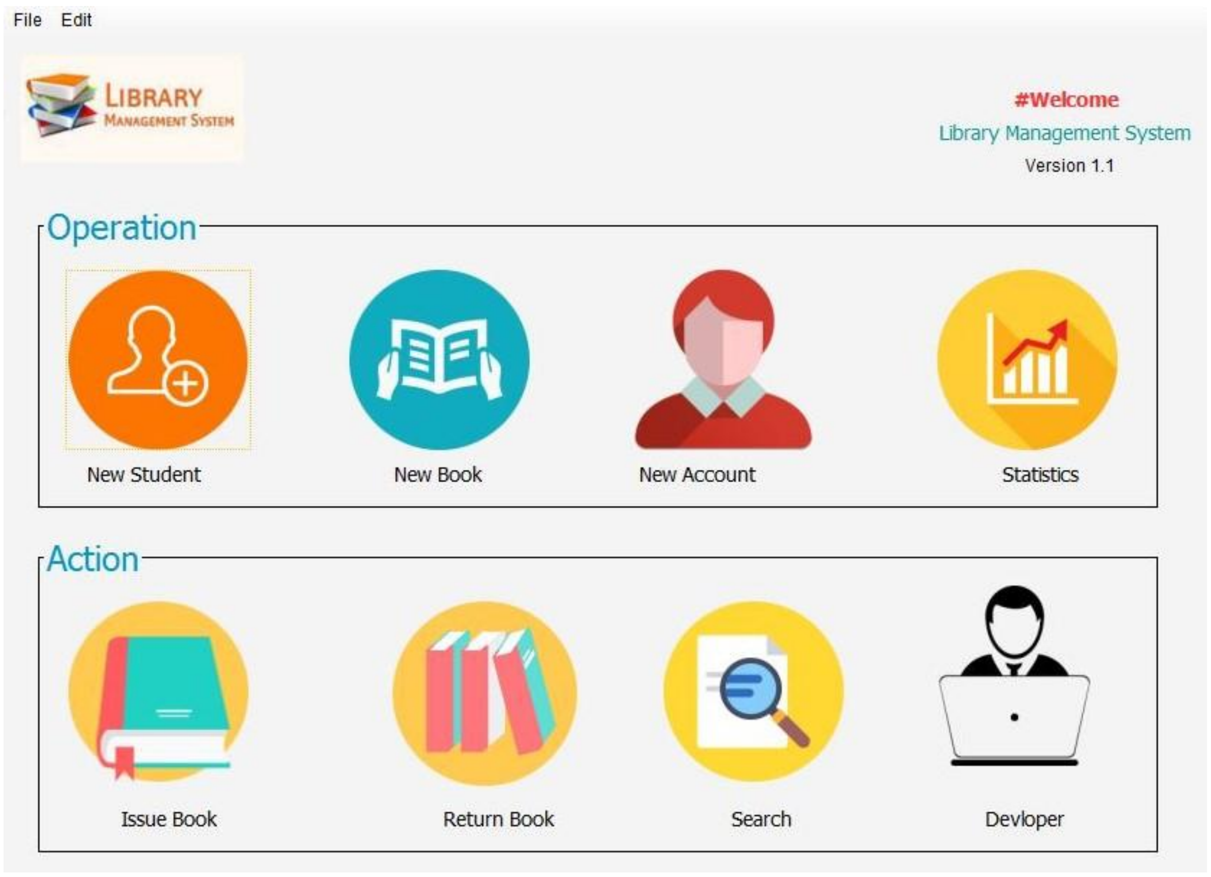
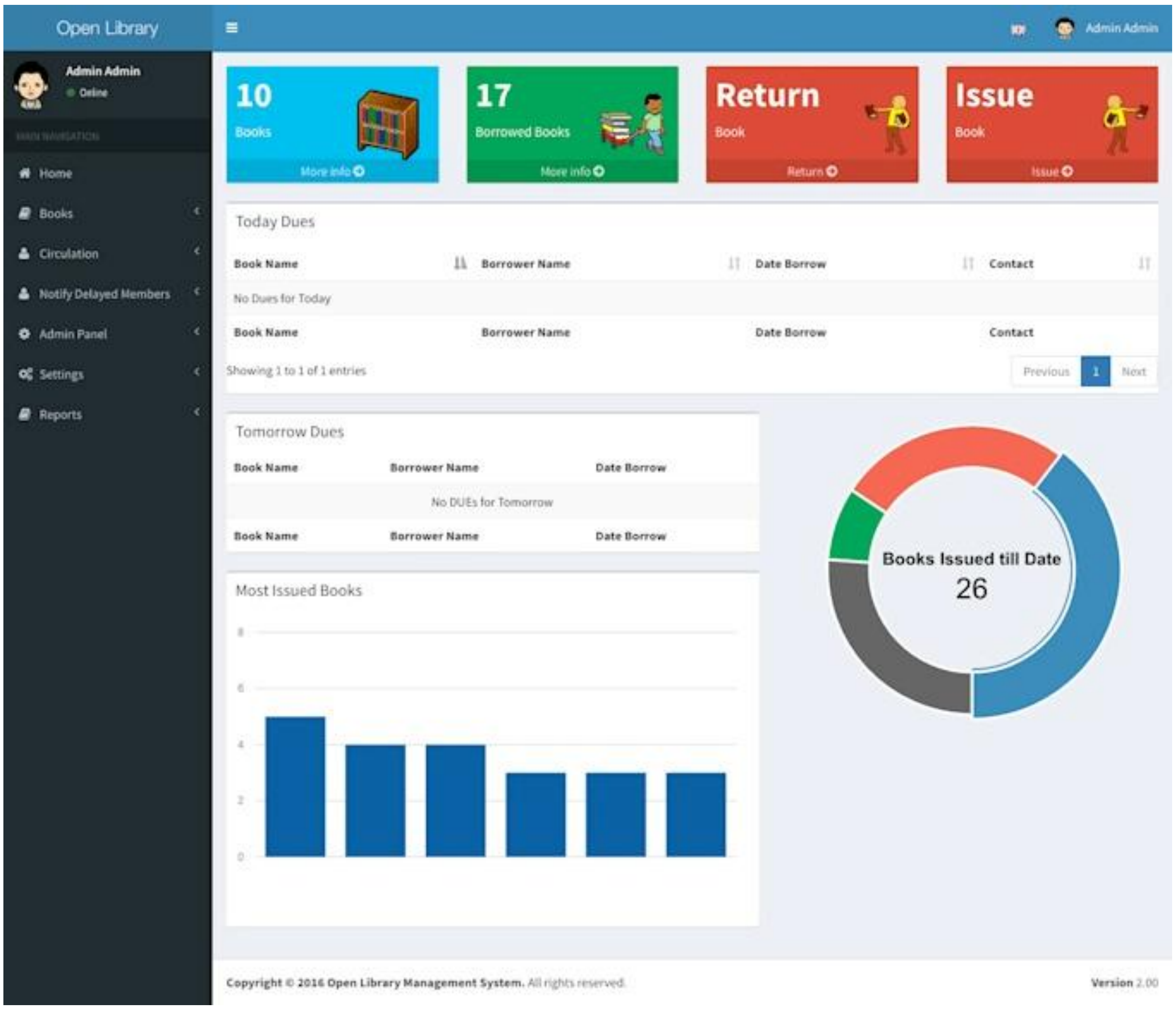


Figure 14 - Default Page UI Mock-Up



Today Dues

Book Name	Borrower Name	Date Borrow	Contact
No Dues for Today			
Book Name	Borrower Name	Date Borrow	Contact

Showing 1 to 1 of 1 entries

Tomorrow Dues

Book Name	Borrower Name	Date Borrow
No DUEs for Tomorrow		
Book Name	Borrower Name	Date Borrow

Copyright © 2016 Open Library Management System. All rights reserved.

Version 2.00

Figure 15 - Main Menu UI Mock-Up

7 Glossary

1. **Book Borrowing**
   - **Definition**: The process where a user borrows a book from the library system for a specific period.
   - **Context**: Users can borrow books according to specific rules and limitations.

2. **Book Returning**
   - **Definition**: The process where a user returns a borrowed book to the library.
   - **Context**: Users are required to return books on time; otherwise, a penalty may be applied.

3. **Reservation**
   - **Definition**: The action where a user requests a book that is currently unavailable, entering a waitlist.
   - **Context**: The user is notified when the book becomes available and must collect it within a specified period.

4. **Fine**
   - **Definition**: A late fee imposed on users who return borrowed books after the due date.
   - **Context**: It is used to encourage users to return books on time.

5. **Inventory**
   - **Definition**: The system that keeps records of all books available in the library.
   - **Context**: Used for managing book stocks, adding new books, updating, or removing them.

6. **Low Stock Alert**
   - **Definition**: A system notification sent to the librarian when the stock of a specific book is running low.
   - **Context**: Ensures that popular books are reordered in time.

7. **Duplicate Book Check**
   - **Definition**: A mechanism that prevents the same book from being added multiple times by mistake.
   - **Context**: Ensures unique book entries in the system.

8. **Librarian**
    - **Definition**: The individual responsible for managing the library and assisting with user operations.
    - **Context**: Handles tasks such as adding, removing, updating books, and managing stock.

9. **User**
    - **Definition**: An individual who borrows, reserves, or uses other library services.
    - **Context**: One of the primary actors in the system who interacts using their membership information.

10. **System Boundary**
    - **Definition**: The boundary between the library automation system and the external world.
    - **Context**: Defines which operations and actors are part of the system.

11. **Functional Requirements**
    - **Definition**: Requirements that define what the system should do.
    - **Context**: Includes user-specific operations such as borrowing, returning, and reserving books.

12. **Non-Functional Requirements**
    - **Definition**: Requirements that define the quality attributes of the system, such as performance, usability, and security.
    - **Context**: Shapes system behavior and user experience.

13. **Notification**
    - **Definition**: Information messages sent to the user by the system.
    - **Context**: Used to inform users about reservations, late fines, or available books.

14. **Maximum Borrowing Limit**
    - **Definition**: The maximum number of books a user can borrow simultaneously.
    - **Context**: Varies depending on the user type registered in the system.

15. **Expiration**
    - **Definition**: The automatic termination of an operation, such as a reservation, once its time limit is exceeded.
    - **Context**: Ensures operations are invalidated when the user exceeds the allocated timeframe.

8 Appendix

- Annex – I: Distribution of Work
- Annex – II: Meeting Minutes

Annex – I

Distribution of Work

Our project team is composed of six members, with each person contributing equally to different project phases to maintain balanced responsibilities:

1. Requirements Analysis – ***Mert Doğan Aygün***: Responsible for gathering initial requirements, documenting system needs, and validating requirements with stakeholders.
2. System Design – ***Berke Alpaslan and Gül Yeşilgil***: Focused on structuring the system's architecture, creating UML diagrams (use-case, object, and sequence diagrams), and defining the relationships between system components.
3. Detailed Design & Dynamic Models – ***Tural Mammadov and Aziz Berk Yıldırım***: Responsible for developing dynamic models, specifying detailed interactions, and refining use cases with precise object behaviors.
4. Documentation & Quality Assurance – ***Birhat Taş*****:** Handles project documentation, reviews for clarity and accuracy, and ensures the final document aligns with the client's requirements.

Annex – II

Meeting Minutes

Date:	24.11.2024
Location:	Microsoft Teams
Duration:	90 min.
Participants:	Berke Alpaslan, Mert Doğan Aygün, Gül Yeşilgil, Birhat Taş, Tural Mammadov, Aziz Berk Yıldırım

Content of the meeting (briefly explain the agenda, decisions, work distributions, etc.)

We discussed the possible changes or additions that could be made to the v2 version in order to finalize our RAD report.

Annex – II

Meeting Minutes

Date:	01.12.2024
Location:	Microsoft Teams
Duration:	120 min.
Participants:	Berke Alpaslan, Mert Doğan Aygün, Gül Yeşilgil, Birhat Taş, Tural Mammadov, Aziz Berk Yıldırım
Content of the meeting (briefly explain the agenda, decisions, work distributions, etc.)	
We created object models for our scenarios.	

Annex – II

Meeting Minutes

Date:	06.12.2024
Location:	Microsoft Teams
Duration:	180 min.
Participants:	Berke Alpaslan, Mert Doğan Aygün, Gül Yeşilgil, Birhat Taş, Tural Mammadov, Aziz Berk Yıldırım

Content of the meeting (briefly explain the agenda, decisions, work distributions, etc.)
We created sequence diagrams for our scenarios, evaluated the mock-ups we prepared, added them to our report, and completed it. Then, we reviewed and evaluated the final version of our report one last time.