LAURENT Sacha – SIMON Eliot

BooknBorrow

Book-borrowing Management
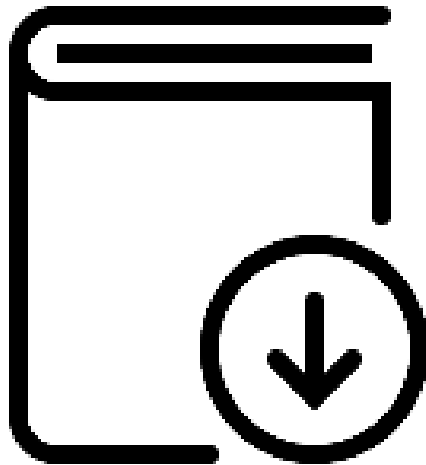

TABLE OF CONTENTS

Project presentation

BooknBorrow book-borrowing Management

Website for book-borrowing. BooknBorrow is a platform for users to borrow books from multiple libraries and keep records of their readings and eventual fines if they bring back books too late.

Members

- LAURENT Sacha
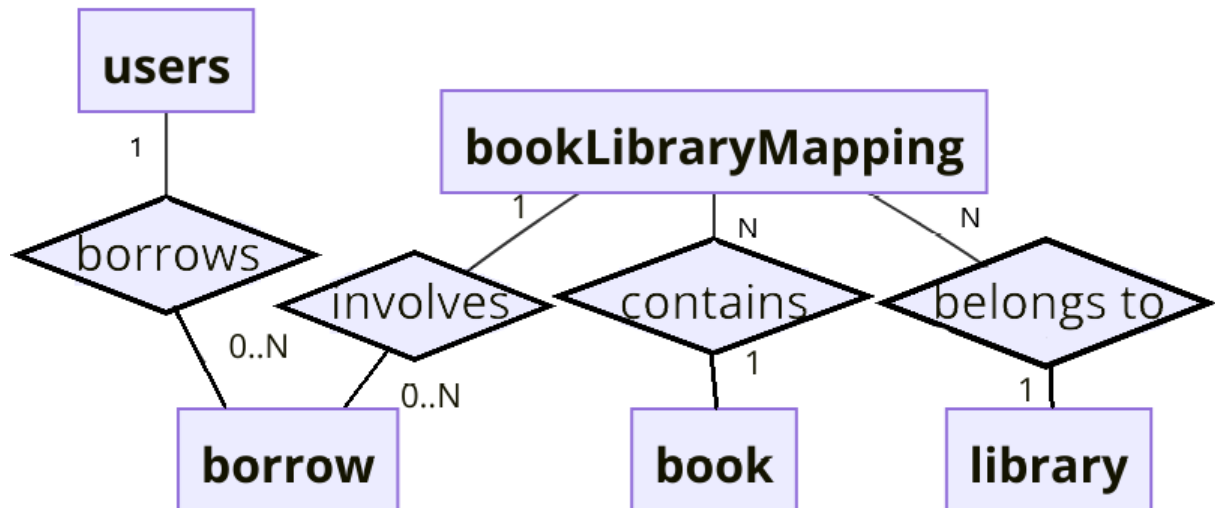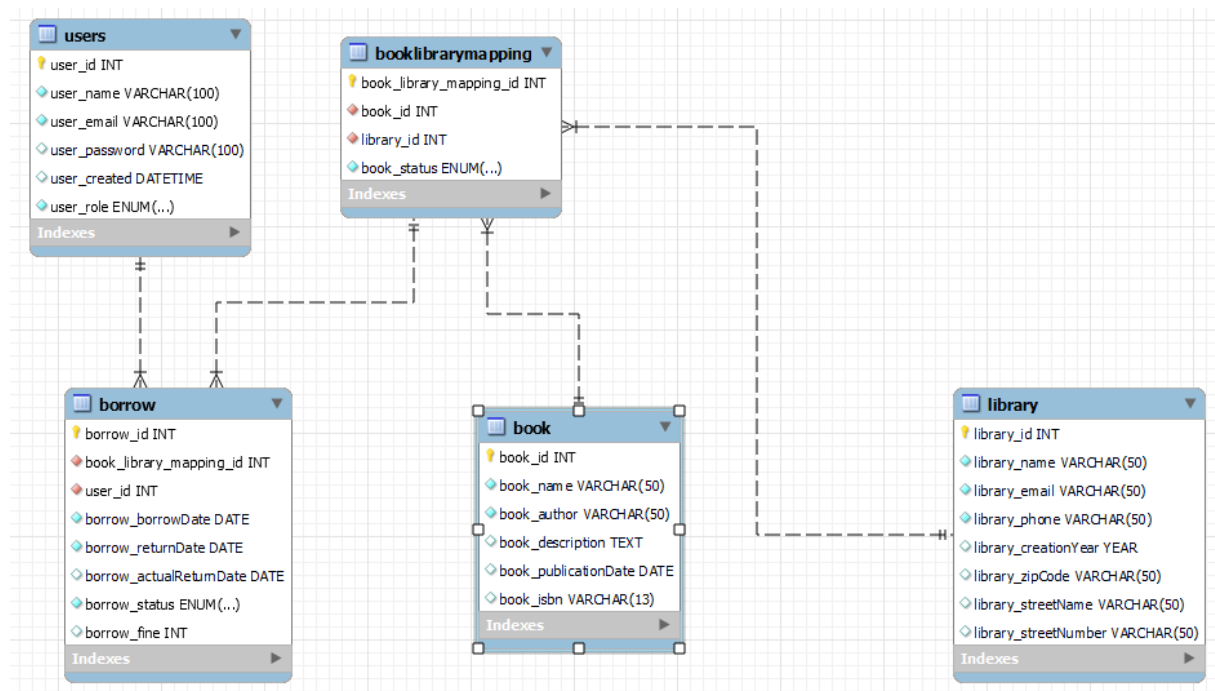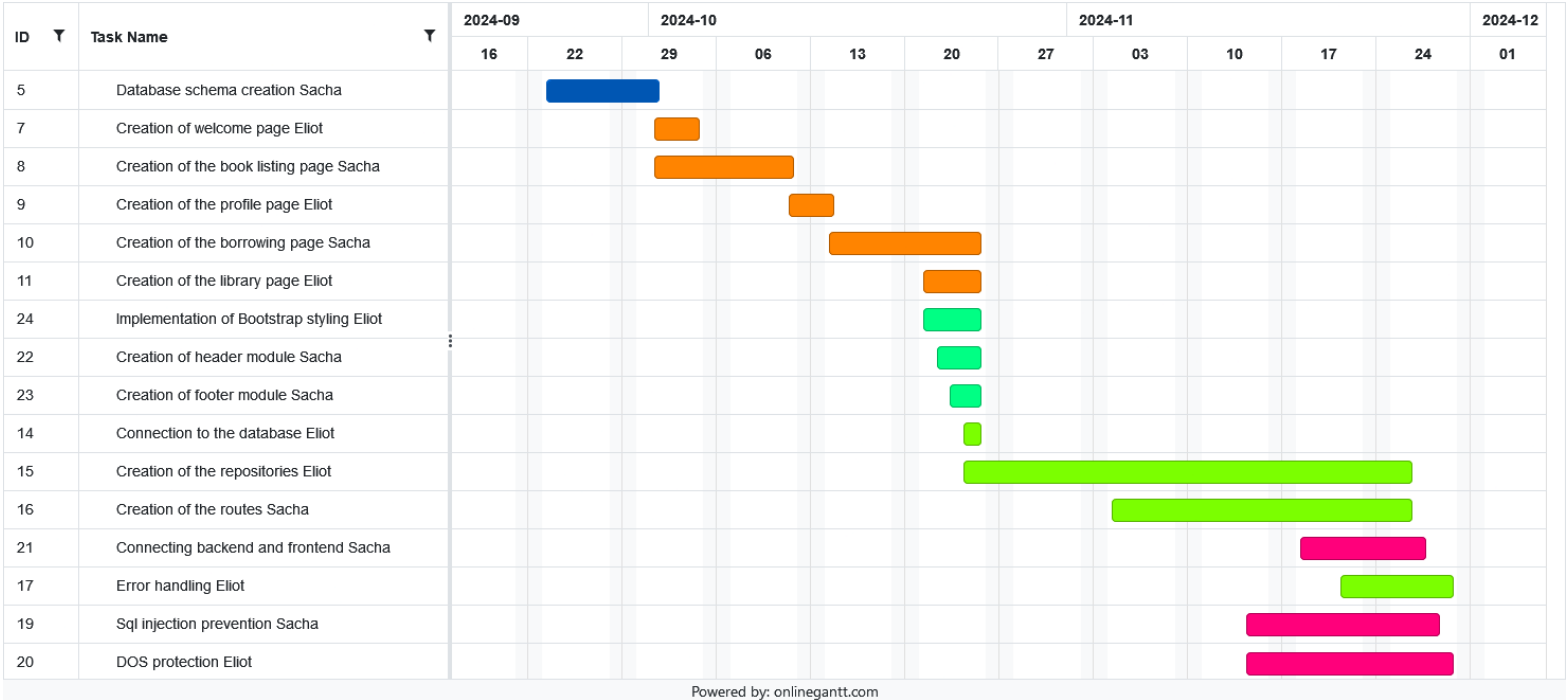- SIMON Eliot

Database diagrams

Entity-relationship diagram

## Table structure diagram

Planned gantt diagram

ID	Task Name	2024-09		2024-10				2024-11					2024-12
		16	22	29	06	13	20	27	03	10	17	24	01
5	Database schema creation Sacha		■										
7	Creation of welcome page Eliot			■									
8	Creation of the book listing page Sacha			■	■								
9	Creation of the profile page Eliot				■								
10	Creation of the borrowing page Sacha					■	■						
11	Creation of the library page Eliot						■						
24	Implementation of Bootstrap styling Eliot						■						
22	Creation of header module Sacha						■						
23	Creation of footer module Sacha						■						
14	Connection to the database Eliot						■						
15	Creation of the repositories Eliot							■	■	■	■		
16	Creation of the routes Sacha								■	■	■		
21	Connecting backend and frontend Sacha										■		
17	Error handling Eliot										■	■	
19	Sql injection prevention Sacha									■	■		
20	DOS protection Eliot									■	■		

Powered by: onlinegantt.com

Legend:

■	Clientside
■	Styling
■	Serverside
■	Safety

Use case diagrams

Borrowing Process Use Case Diagram
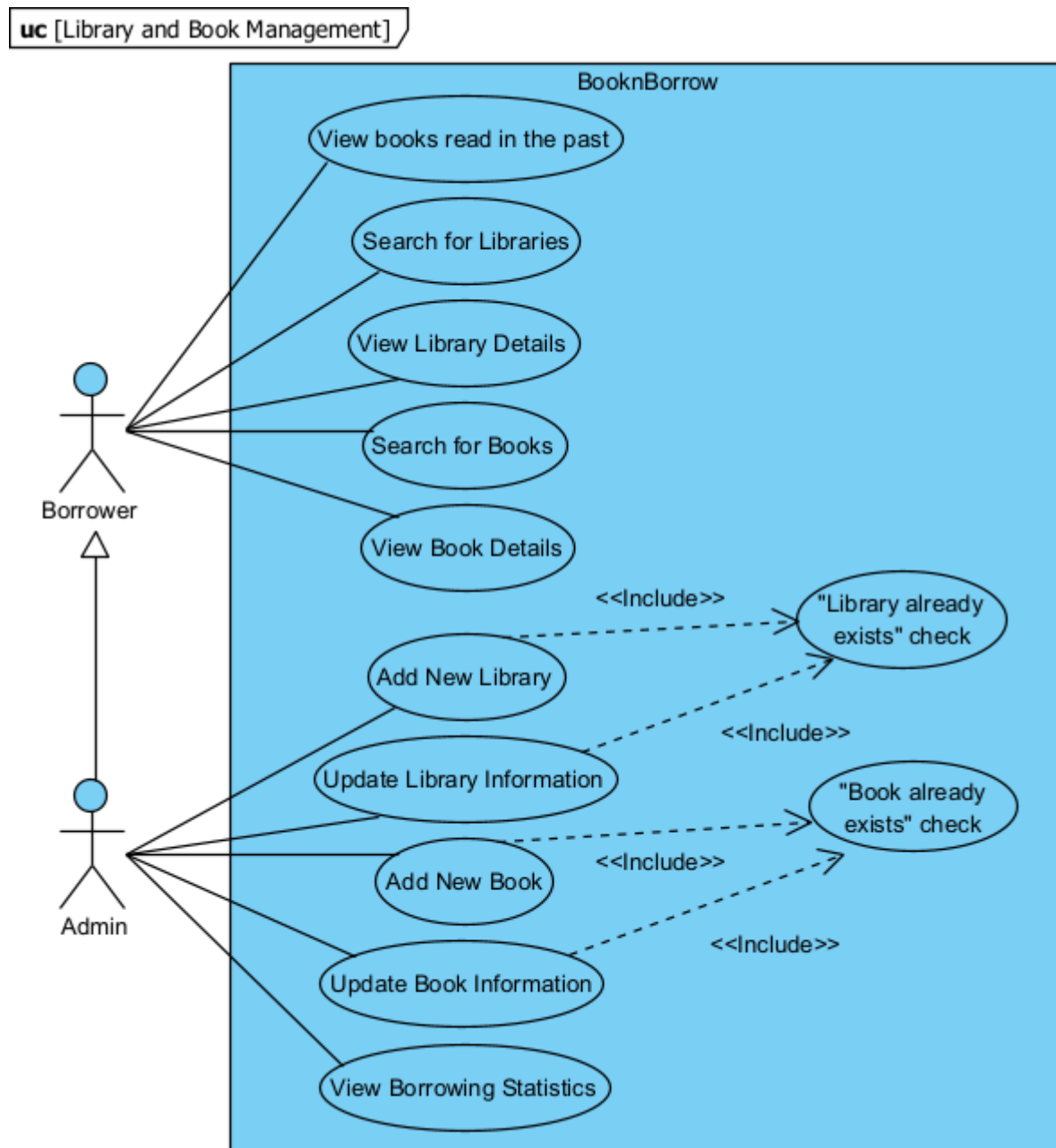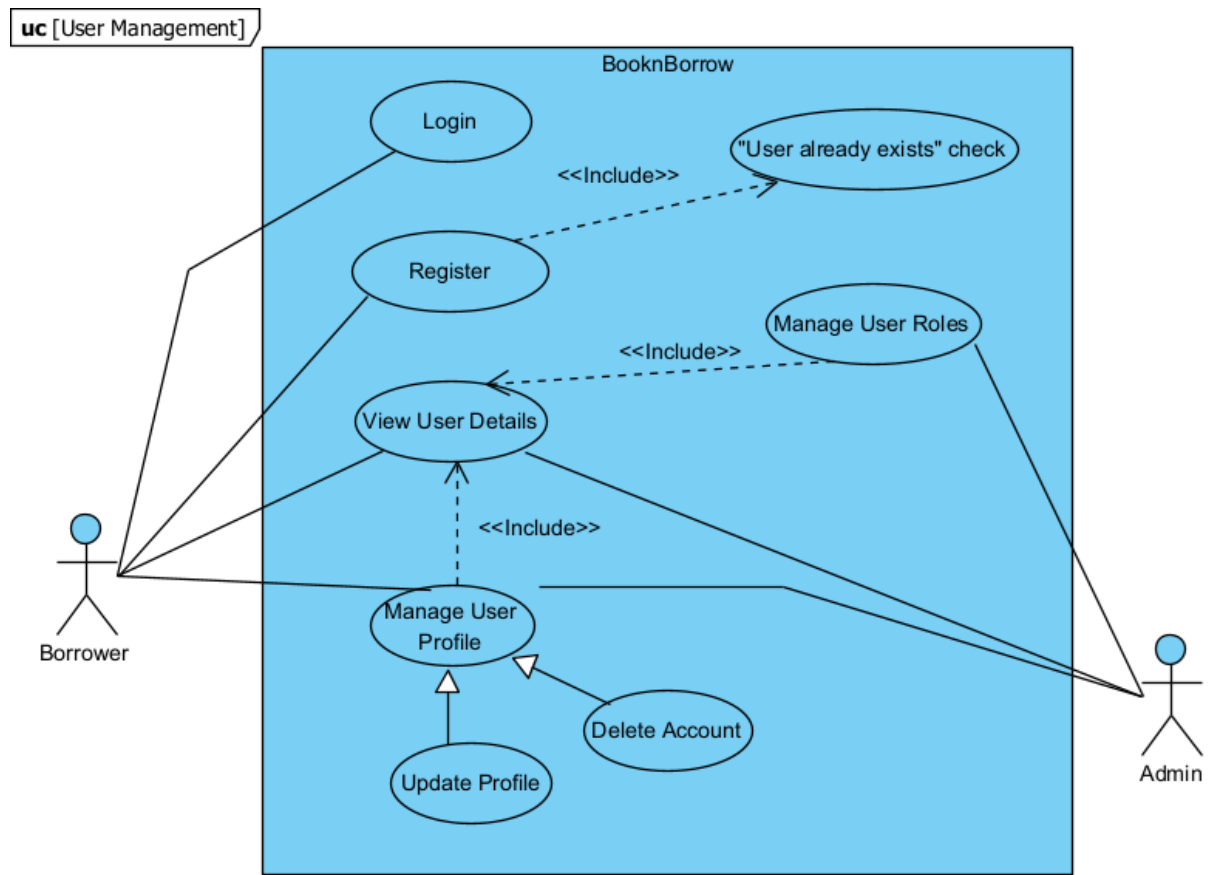
uc [Borrowing Process]
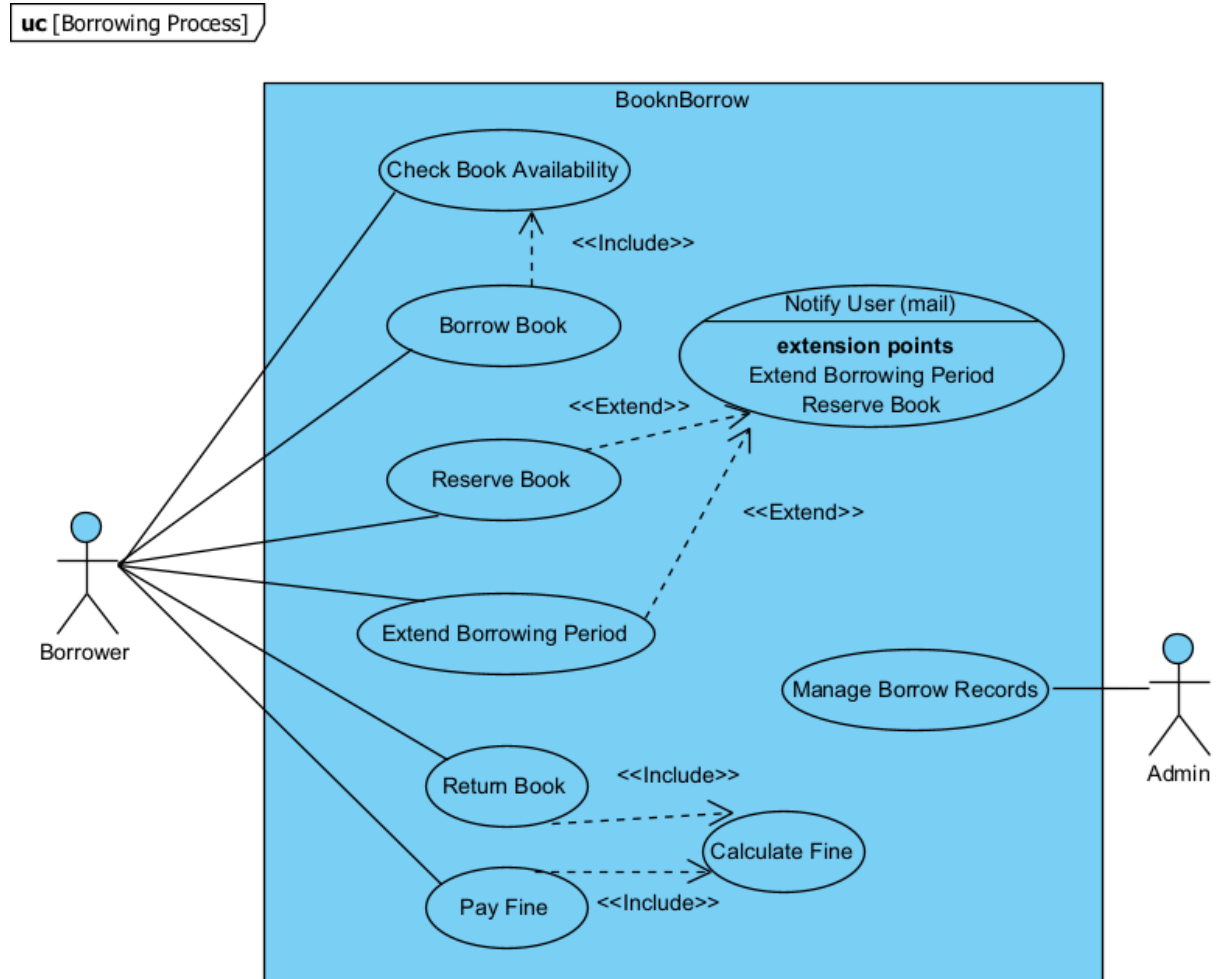
- **Actors**: User, Admin
- **Use Cases**: Borrow Book, Reserve Book, Return Book, Check Book Availability, Extend Borrowing Period, Pay Fine (User), Manage Borrow Records (Admin).

Library and Book Management Use Case Diagram

uc [Library and Book Management]

BooknBorrow

View books read in the past

Search for Libraries

View Library Details

Search for Books

View Book Details

Add New Library

Update Library Information

Add New Book

Update Book Information

View Borrowing Statistics

Borrower

Admin

<<Include>> "Library already exists" check

<<Include>>

"Book already exists" check

<<Include>>

<<Include>>

- **Actors**: Admin, User
- **Use Cases**: Add New Library (Admin), View Library Details, Add New Book (Admin), View Book Details, View books read in the past, Update Book Information (Admin), Update Library Information (Admin), Search for Books, Search for Libraries, View Borrowing Statistics (Admin).
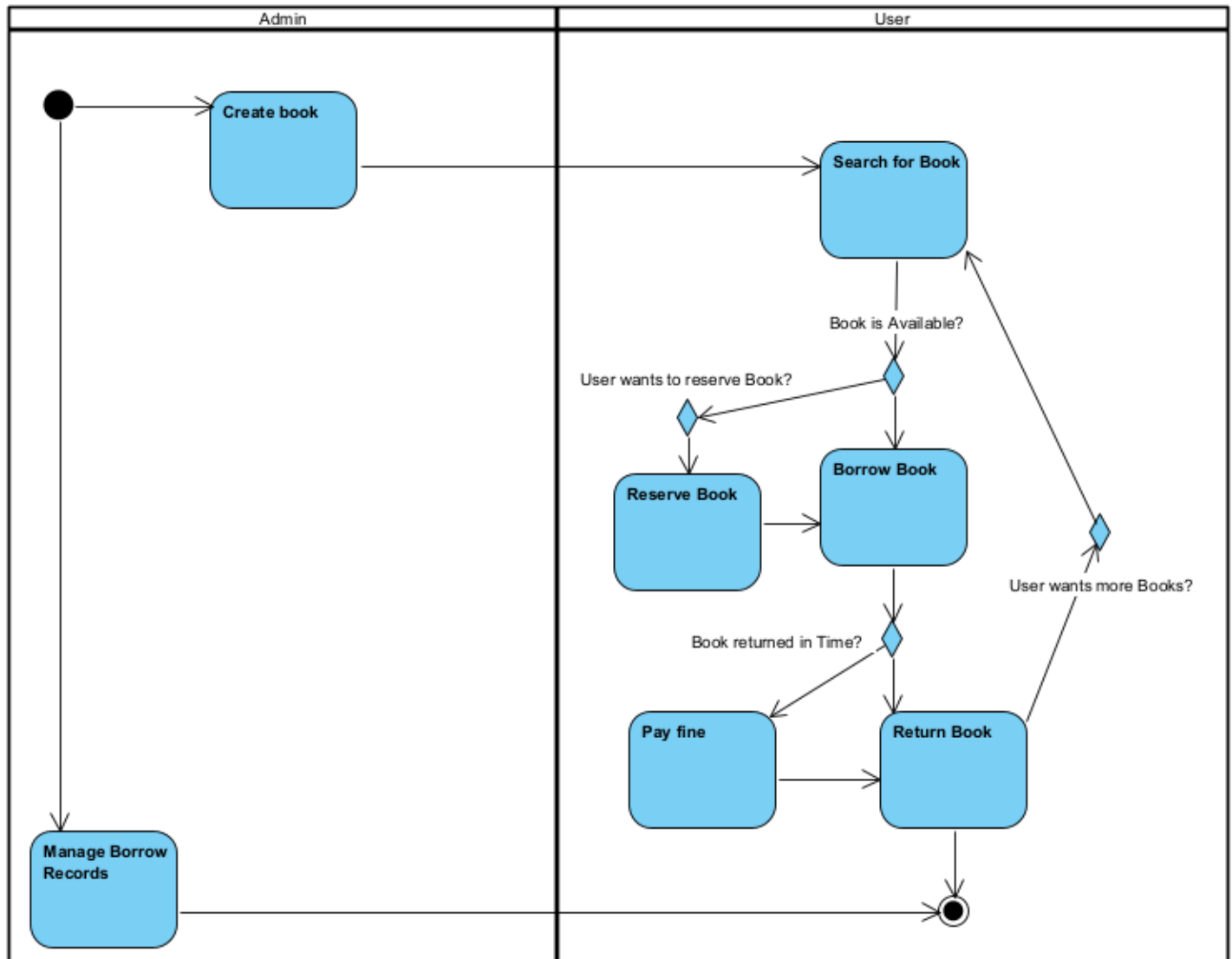
User Management Use Case Diagram:

- **Actors**: Admin, User
- **Use Cases**: Register, Login, Update Profile, View User Details, Manage User Roles (Admin only), Delete Account, Manage Users (Admin only).

Activity diagram

Borrowing Process

act [Borowwing Process]

Actors: Admin, User

The Admin can initiate the process by creating a new book entry, which becomes accessible to the User for searching and borrowing. The User starts by searching for a book. If the book is available, the User can choose to borrow it or reserve it if needed. Once borrowed, the User must return the book on time. If returned late, a fine is imposed. The process also allows the User to borrow additional books after completing a transaction. Finally, the Admin manages the borrowing records to complete the process.

User Management

act [User Management]
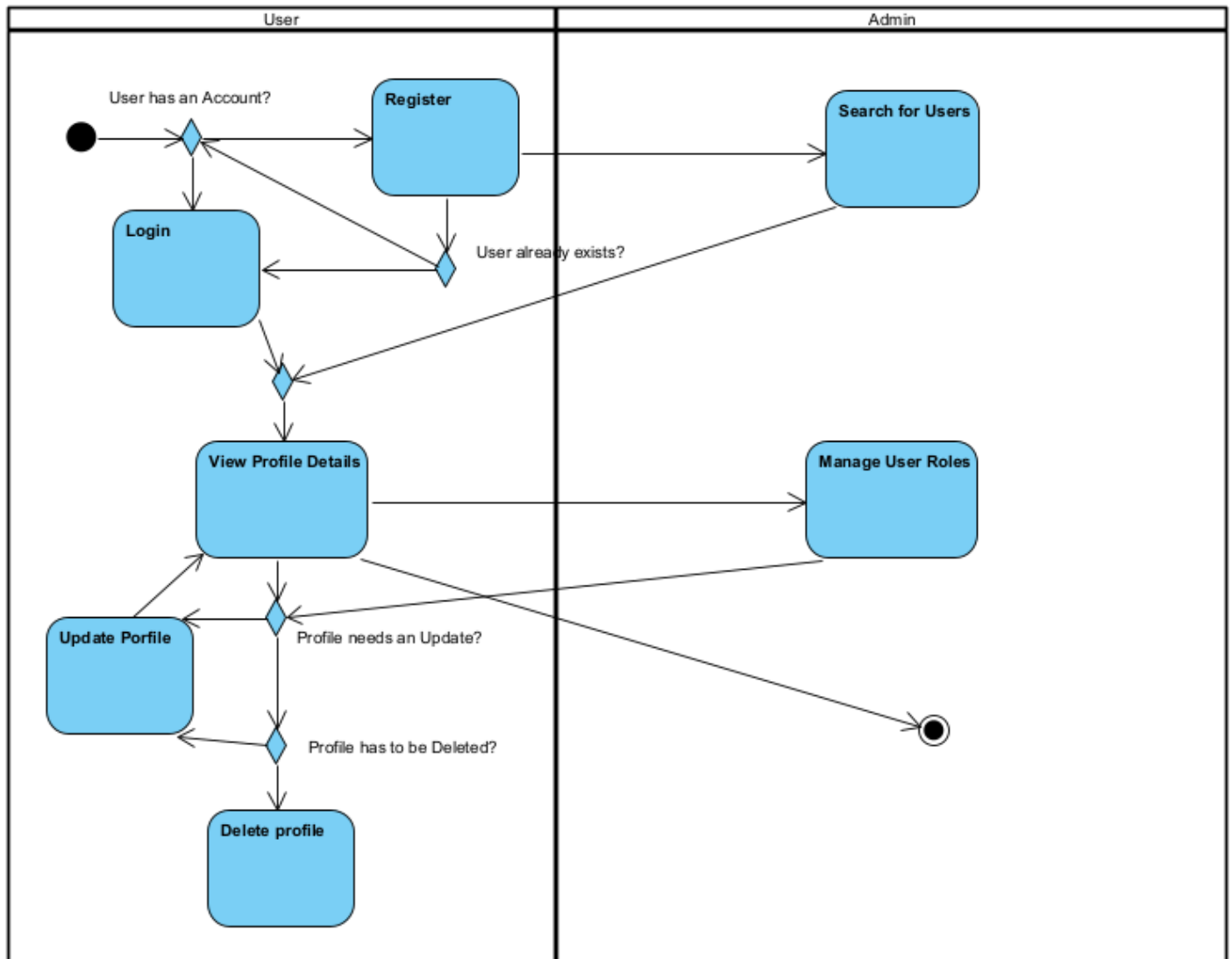
Actors: Admin, User

For Users, the process begins by checking if they already have an account. If they don't, they are prompted to register. If an account exists, they proceed to log in. Once logged in, the User can view their profile details. If necessary, the profile can be updated or deleted, depending on the User's needs.

On the Admin side, they can search for users and manage user roles, ensuring proper access control within the system. The diagram shows the various pathways that can be followed depending on the User's actions, such as updating or deleting their profile, while also illustrating the admin's ability to oversee user roles and account management.

Sequence diagrams

Book Filering
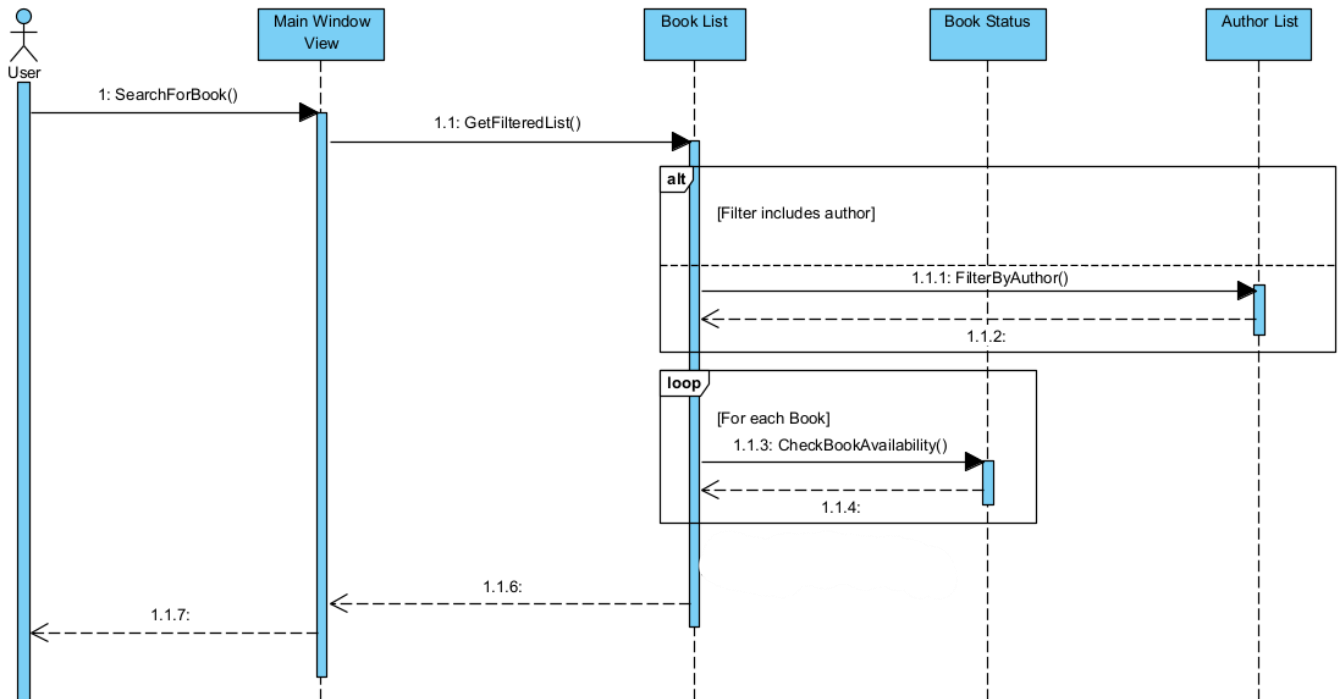

The Sequence begins with the User initiating a SearchForBook() request in the Main Window View. The Main Window View then sends a request to retrieve a filtered list of books (GetFilteredList()) to the Book List component. If the filter criteria include an author, the Book List interacts with the Author List to apply FilterByAuthor(), updating the list accordingly. Once filtered, the Book List updates, and the filtered list is returned back through the Main Window View to the User.
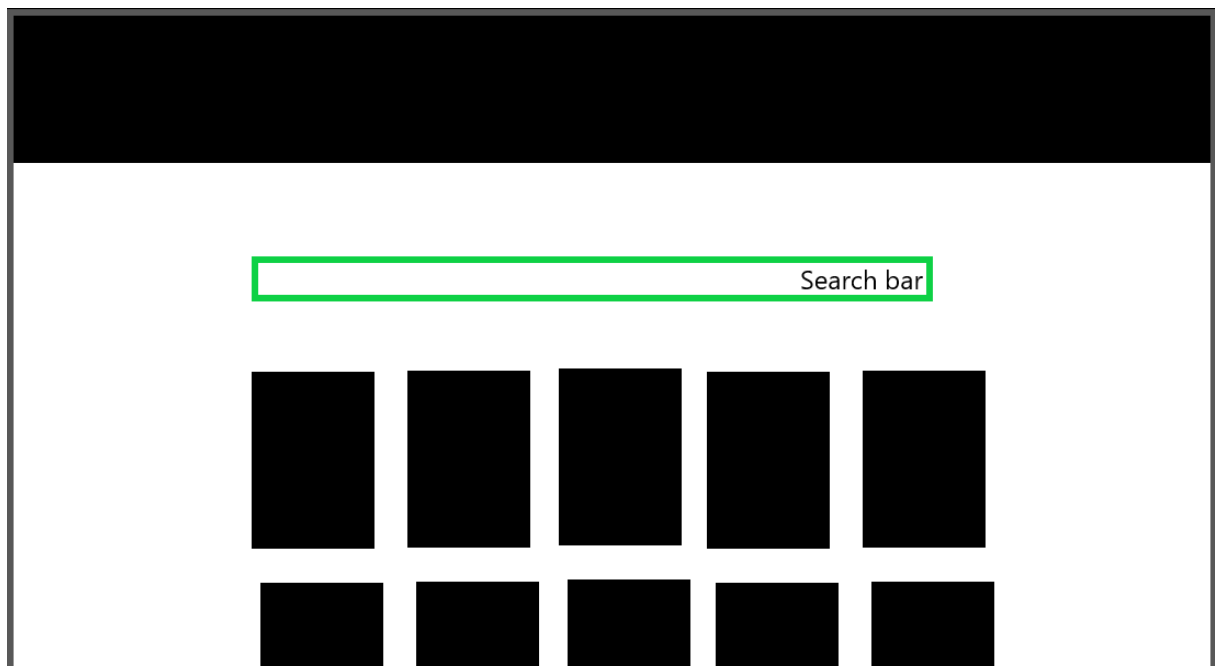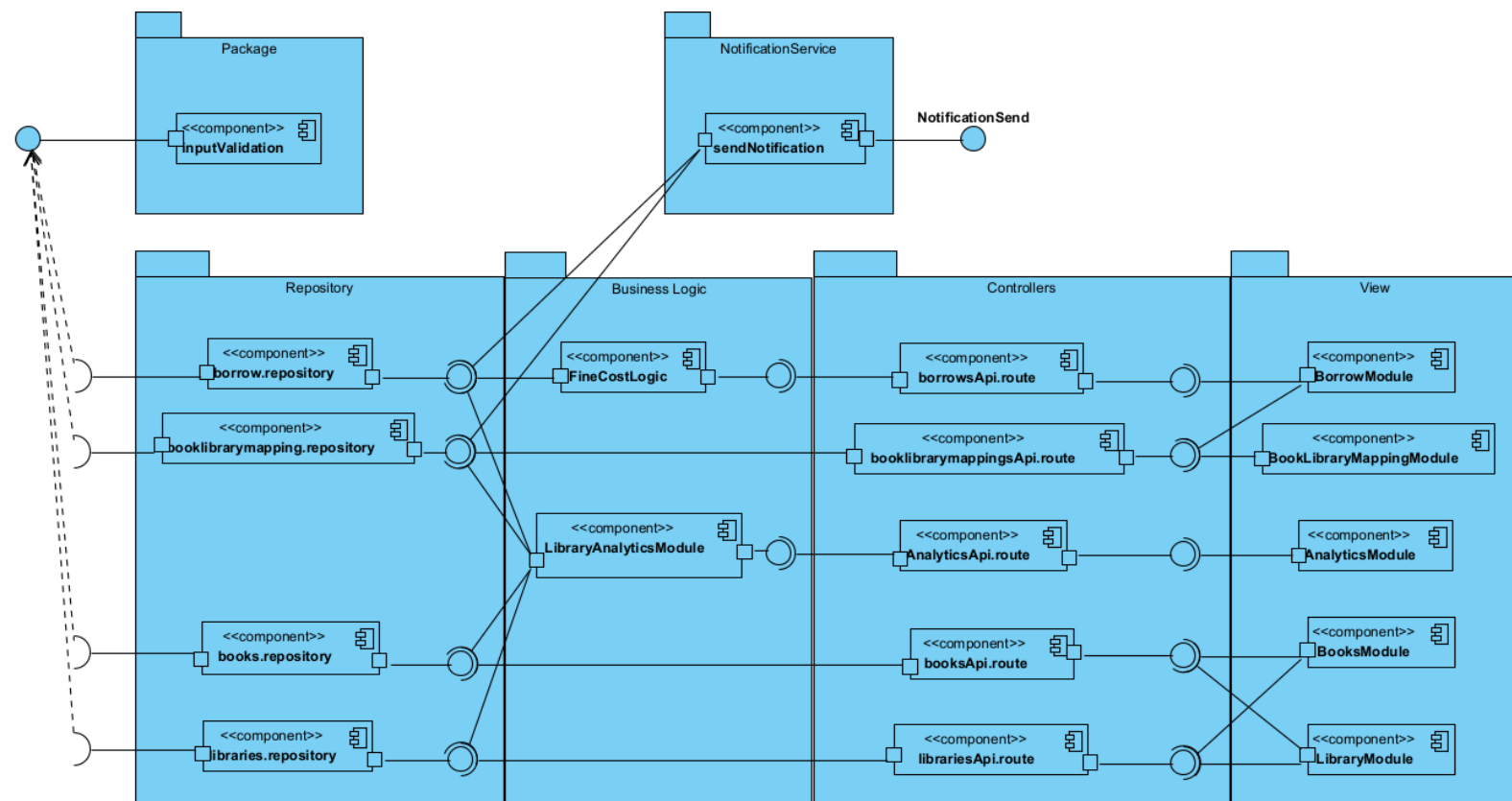
Login
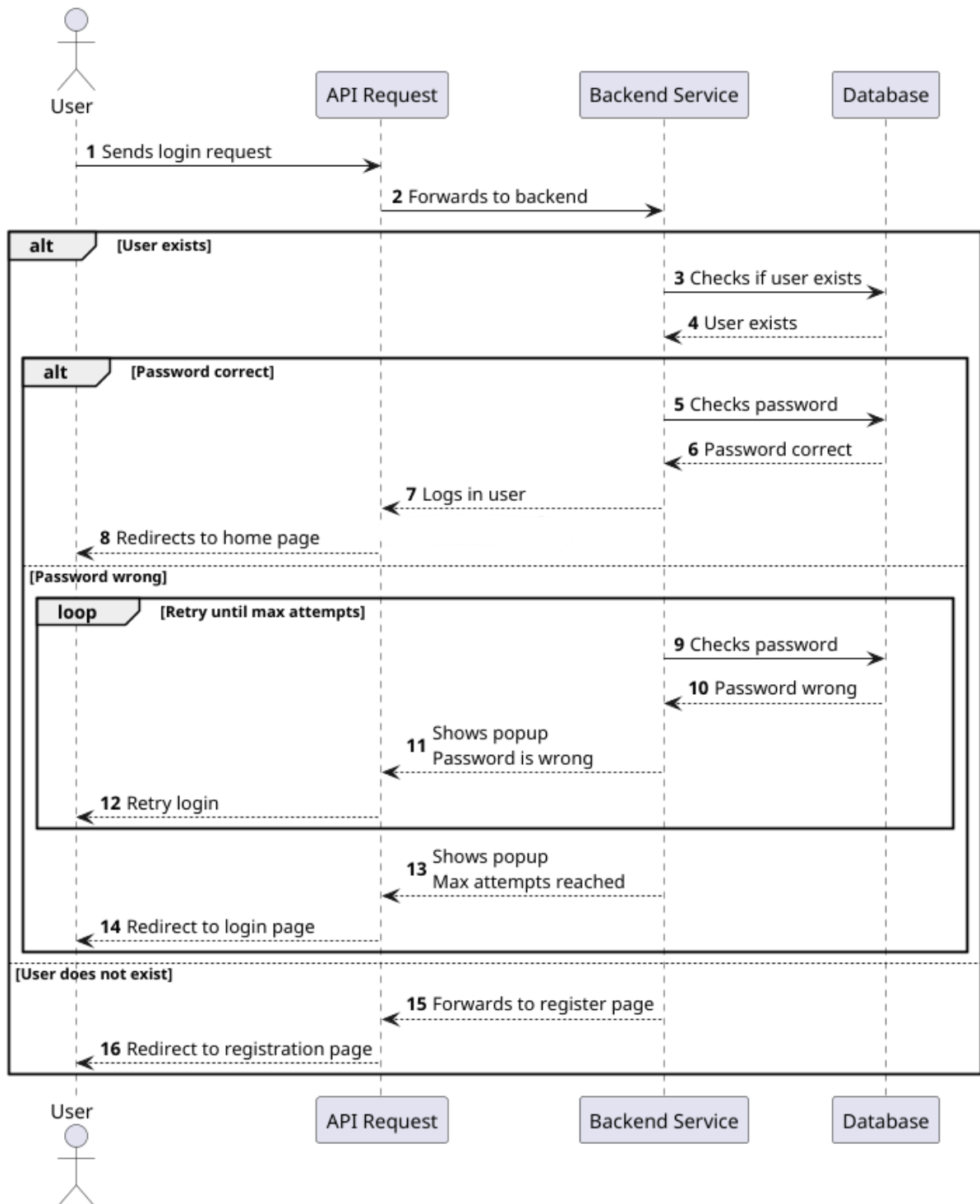
Wireframe

Front page
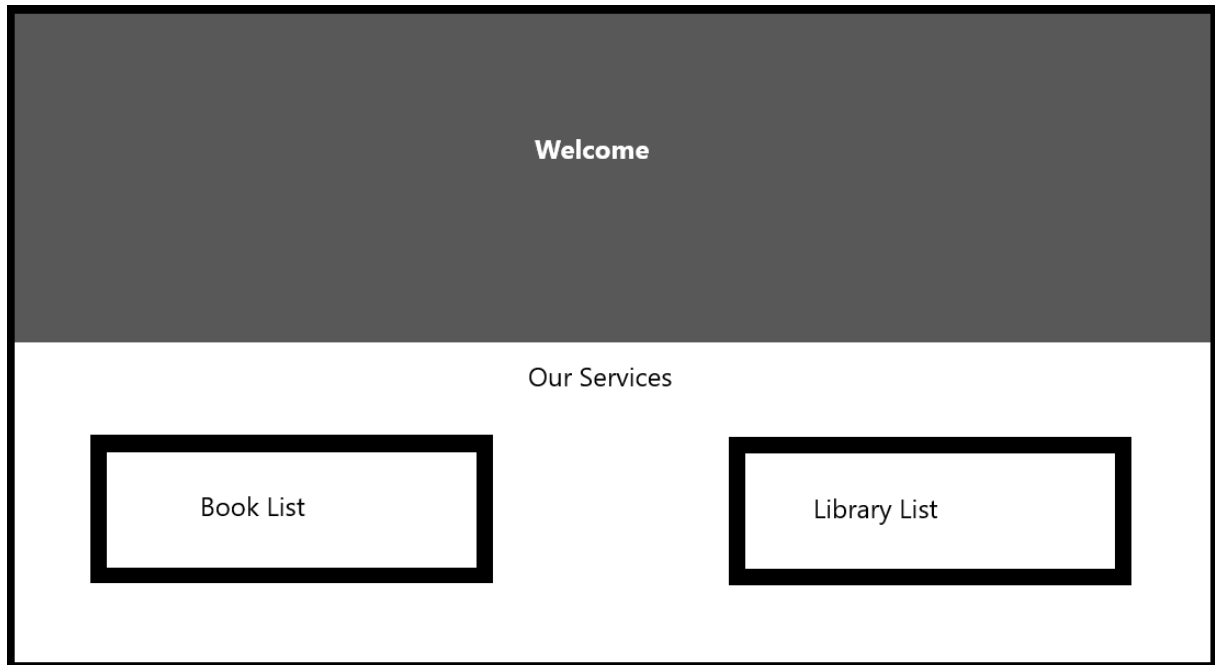
Book list page

Component diagram

Subsystems:

- **Repository Layer** consists of components like books.repository, libraries.repository, and booklibrarymapping.repository, which manage database interactions for specific entities.
- **Business Logic Layer** includes a FineCostLogic component to handle fine calculations.
- **Controllers Layer** contains API route components such as booksApi.route and librariesApi.route, which coordinate between the business logic and the user-facing modules.
- **View Layer** comprises modules like BooksModule and LibraryModule, responsible for presenting data and interacting with the controllers.
- **InputValidation package** makes the system immune to SQL injection

Class diagram
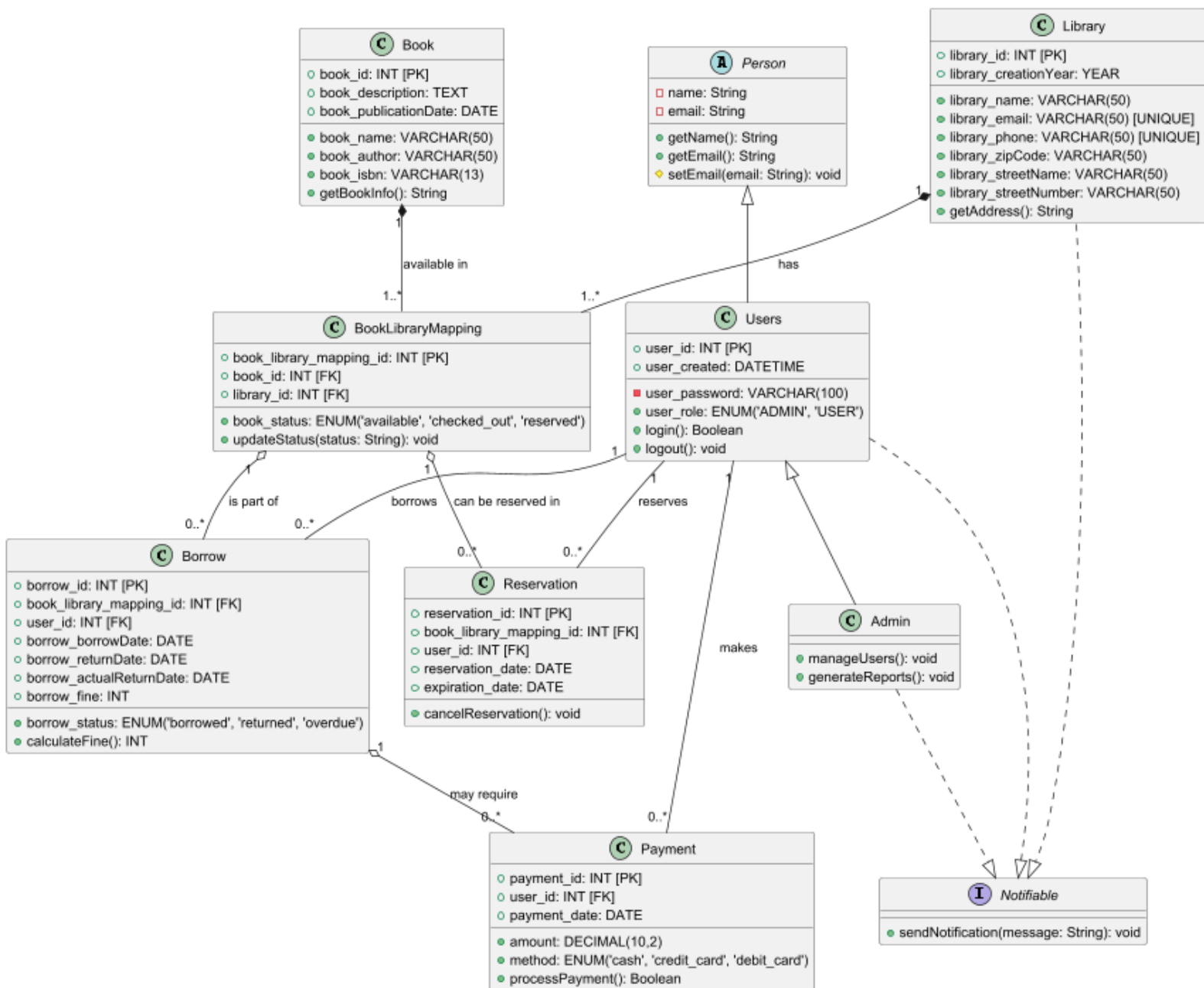
Core Classes:

- Book, Library, Users, Admin, and Payment are the primary classes, with BookLibraryMapping, Borrow, and Reservation linking their relationships and functionality.

Relationships:

- A Library holds many Books via BookLibraryMapping. The BookLibraryMapping tracks the status (available, borrowed, reserved) of each book in specific libraries.
- Users interact with books in two ways:
    - They can borrow a book through the Borrow class, which records dates (ongoing, finished, overdue) and calculates fines.
    - They can reserve a book using the Reservation class, which manages expiration dates and cancellations.
- Admin is a specialized role of Users, with additional responsibilities to manage users and generate reports.

Interactions and Processes:

- Borrowing or reserving a book may lead to a Payment, which handles amounts, payment methods, and processing.
- Notifications are managed by the Notifiable interface, enabling the system to send alerts to users (e.g., overdue books or reservation updates).

Inheritance and Methods:

- Admin extends Users with administrative functionalities.
- Shared methods such as getBookInfo() (in Book), processPayment() (in Payment), and updateStatus() (in BookLibraryMapping) ensure modular and reusable functionality.