BooknBorrow

Book-borrowing Management

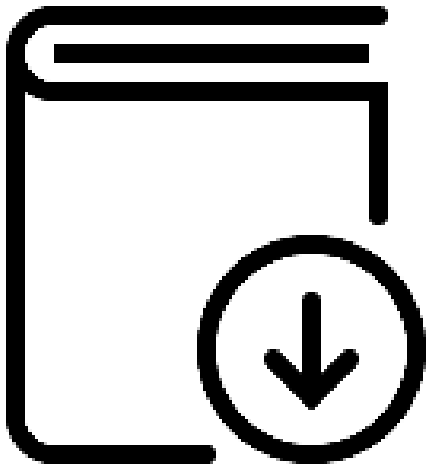


TABLE OF CONTENTS

Project presentation

BooknBorrow book-borrowing Management

BooknBorrow is a 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.

Database diagrams

Entity-relationship diagram

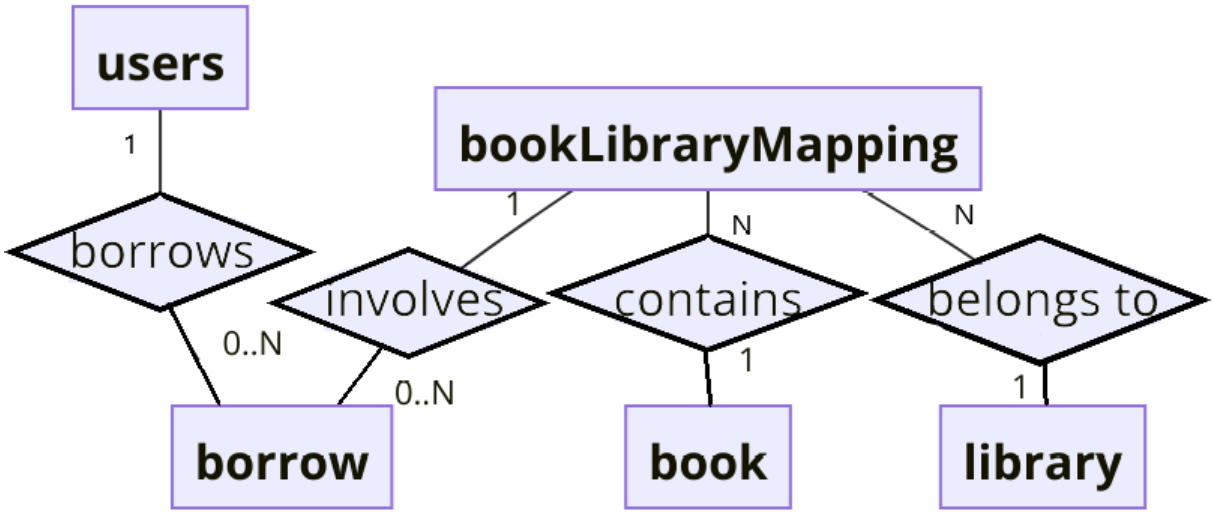
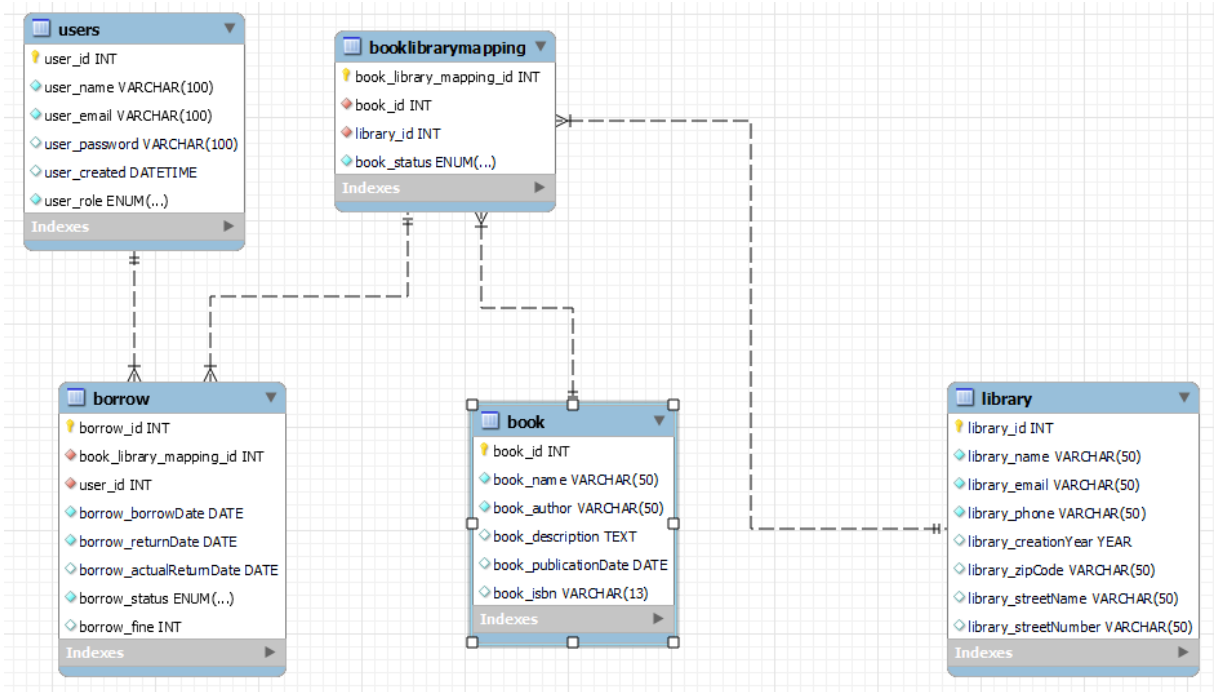
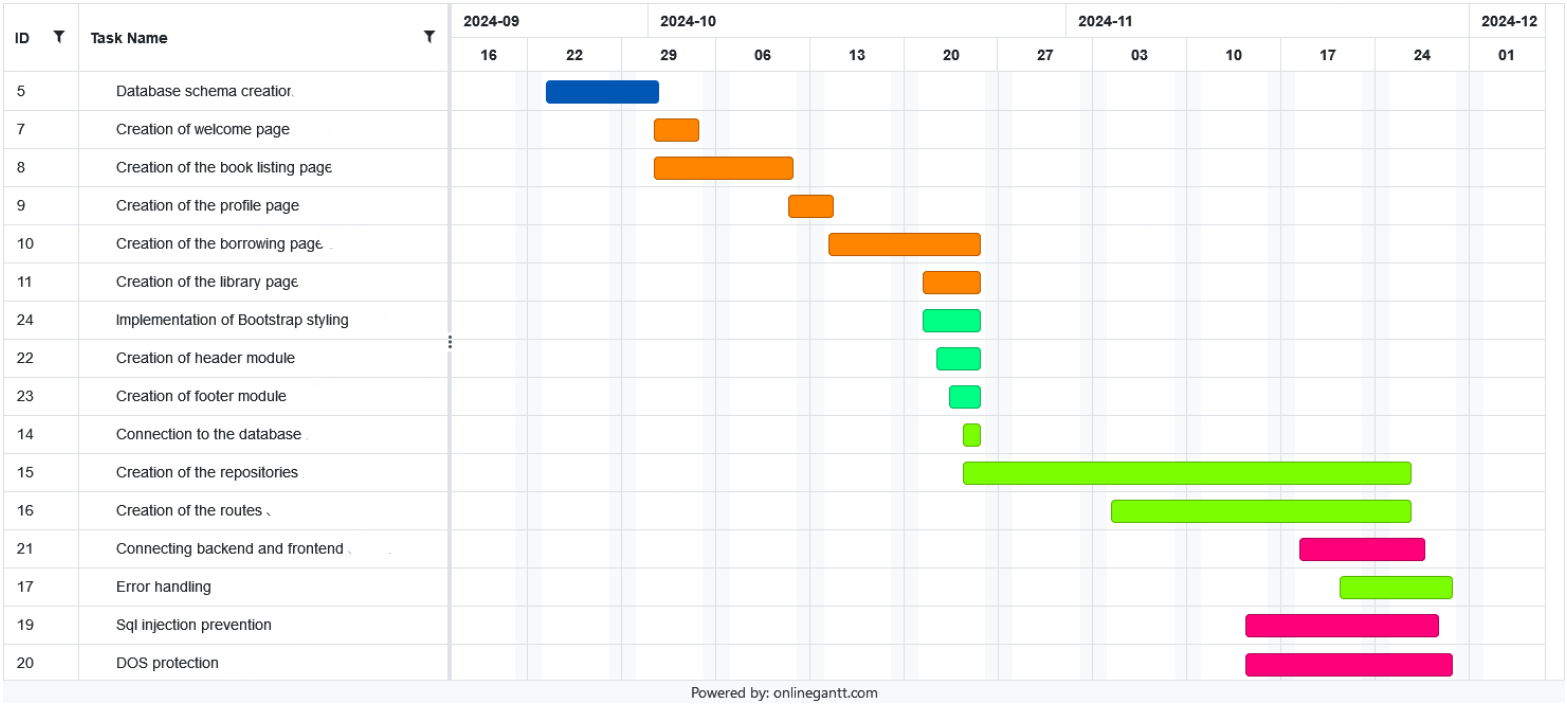


Table structure diagram



Gantt diagrams

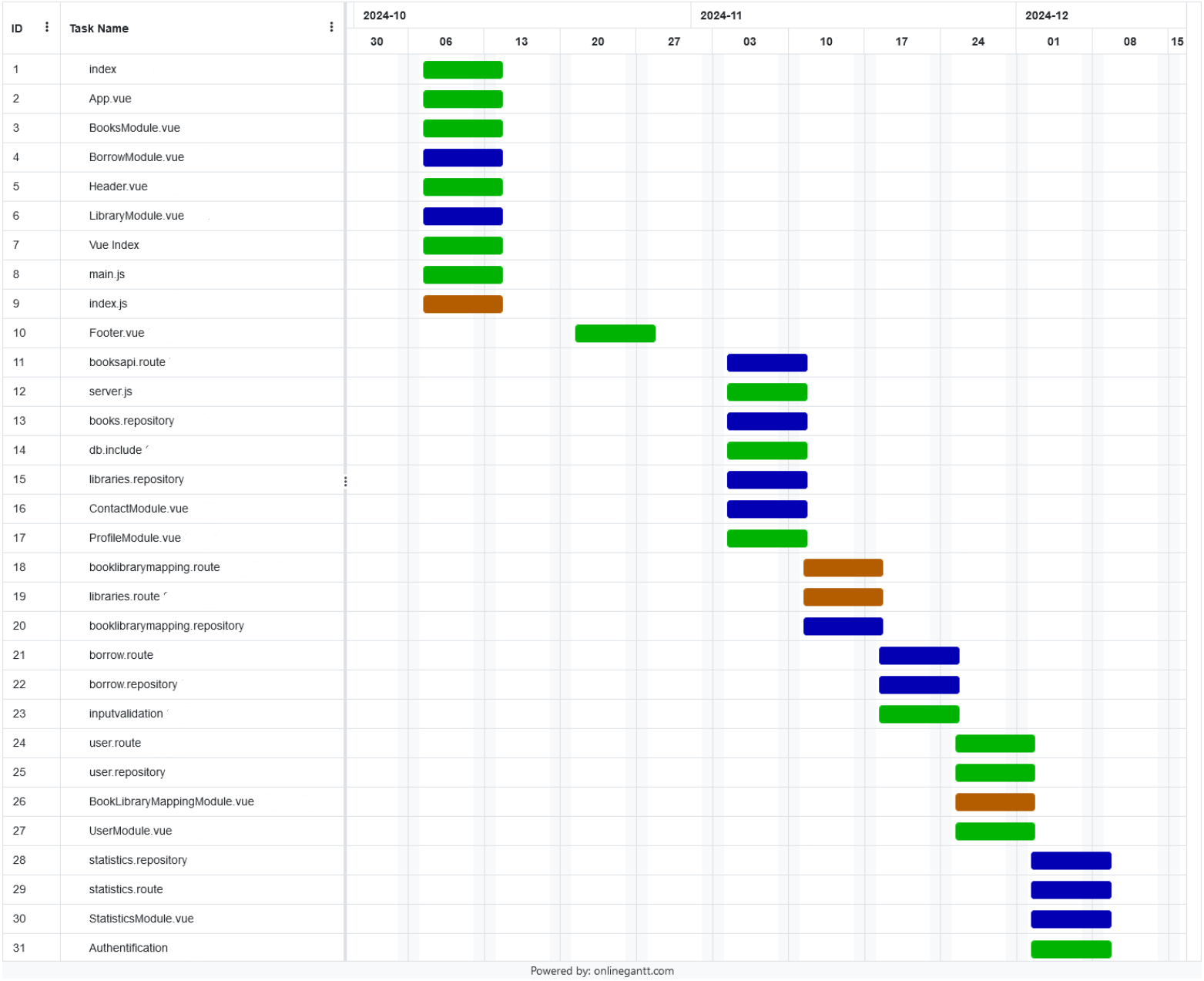
Planned gantt diagram



Legend:

	Clientside
	Styling
	Serverside
	Safety

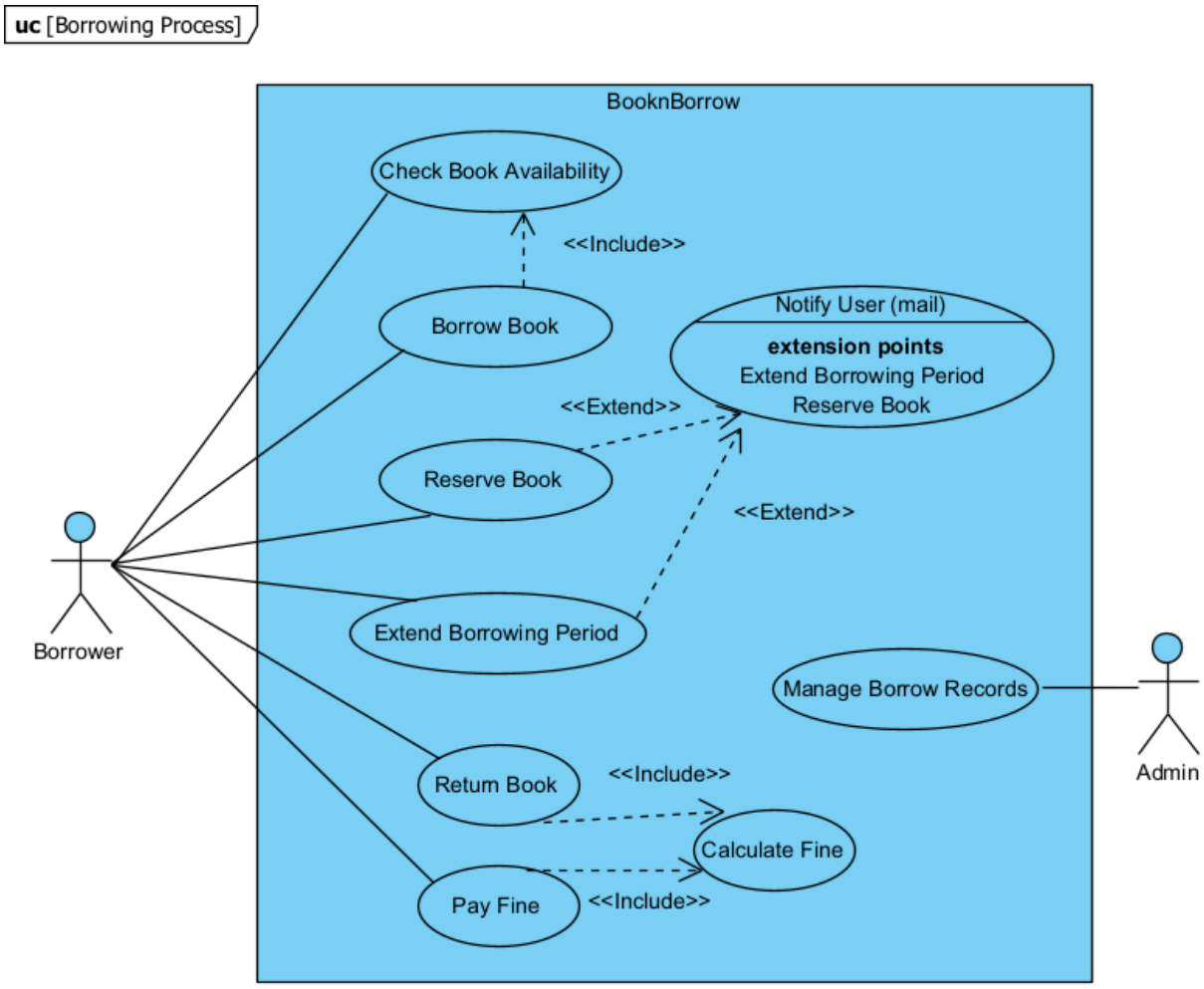
Actual gant diagram



	Hormone
	Both
	Eliot

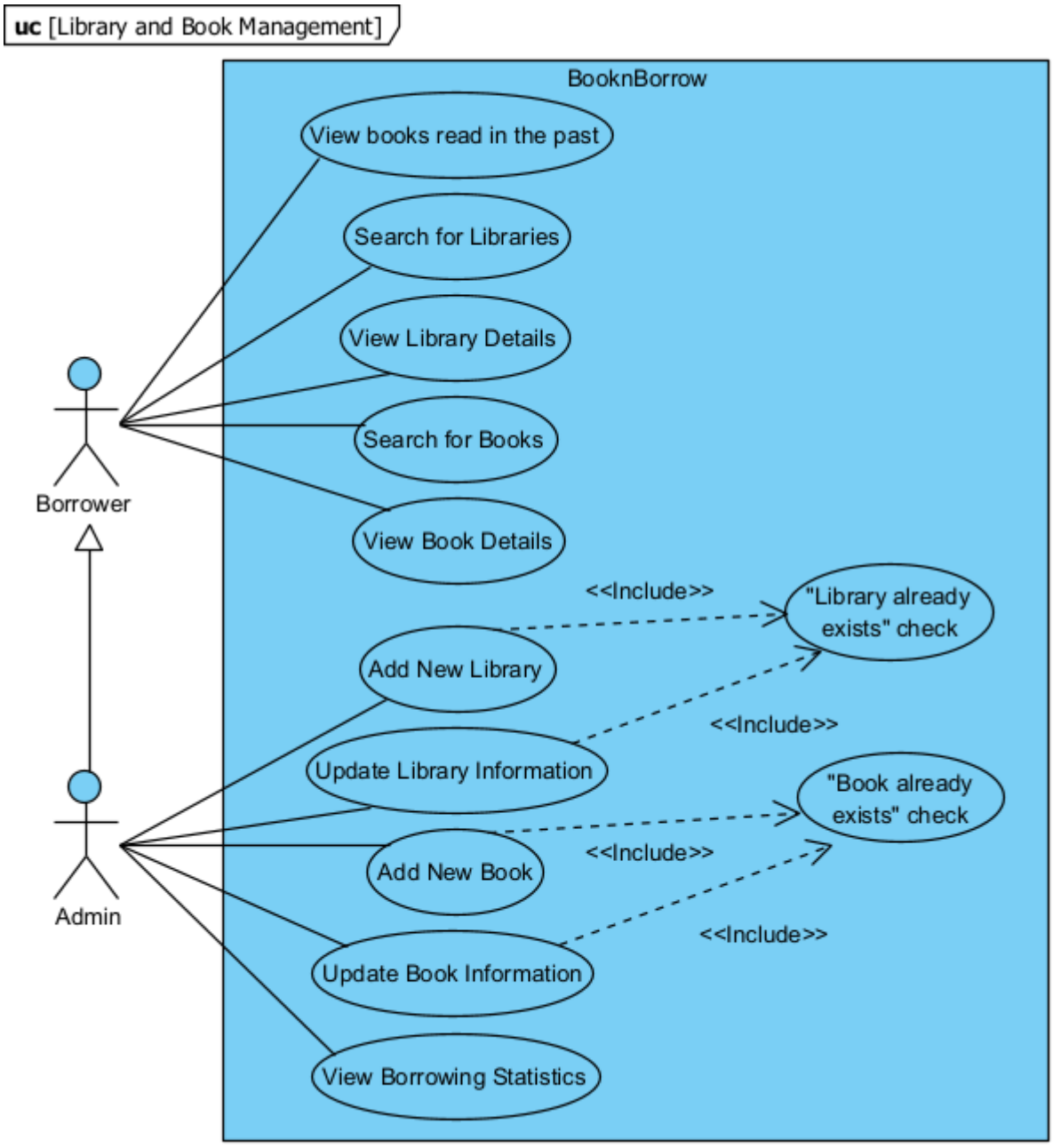
Use case diagrams

Borrowing Process Use Case Diagram



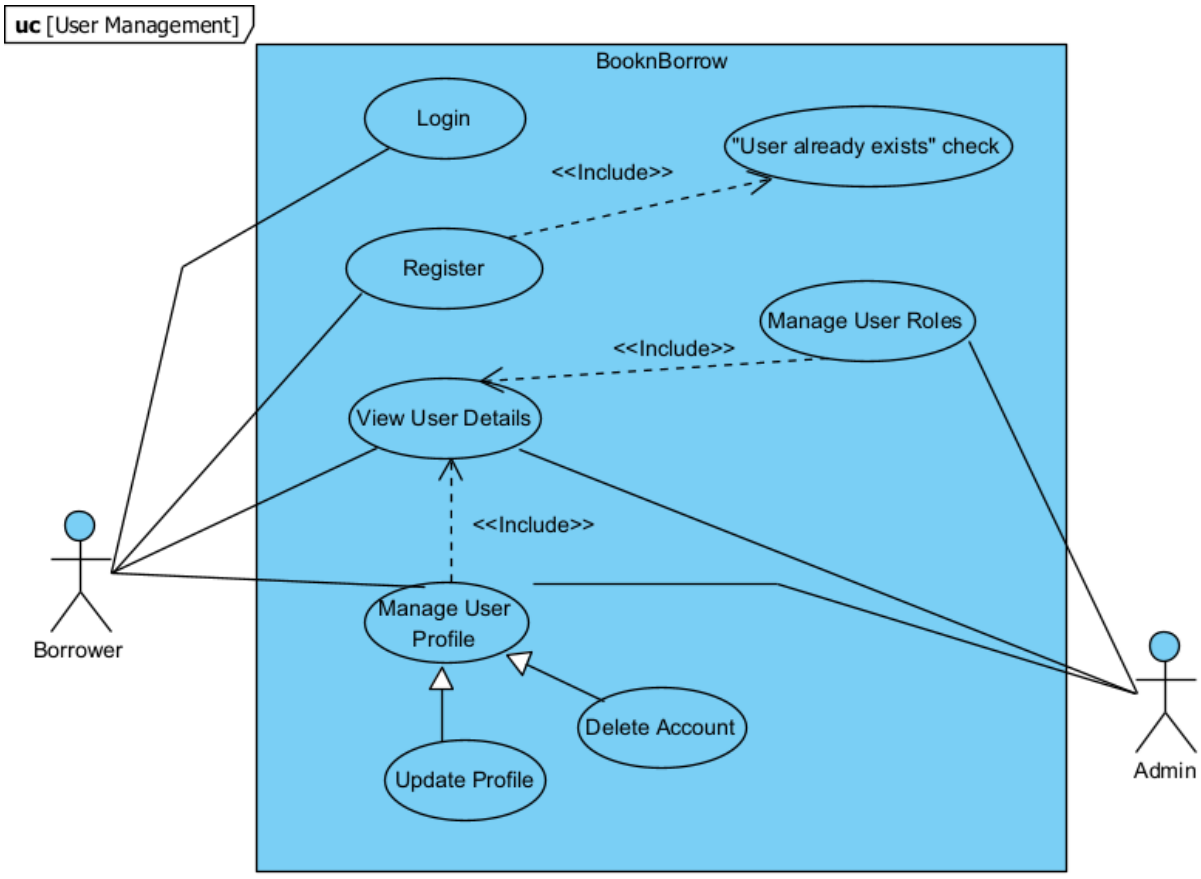
- **Actors**: User, Admin
- **Use Cases**: Borrow Book, Reserve Book, Return Book, Check Book Availability, Extend Borrowing Period, Pay Fine (User), Manage Borrow Records (Admin).

Library and Book Management Use Case Diagram



- **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).

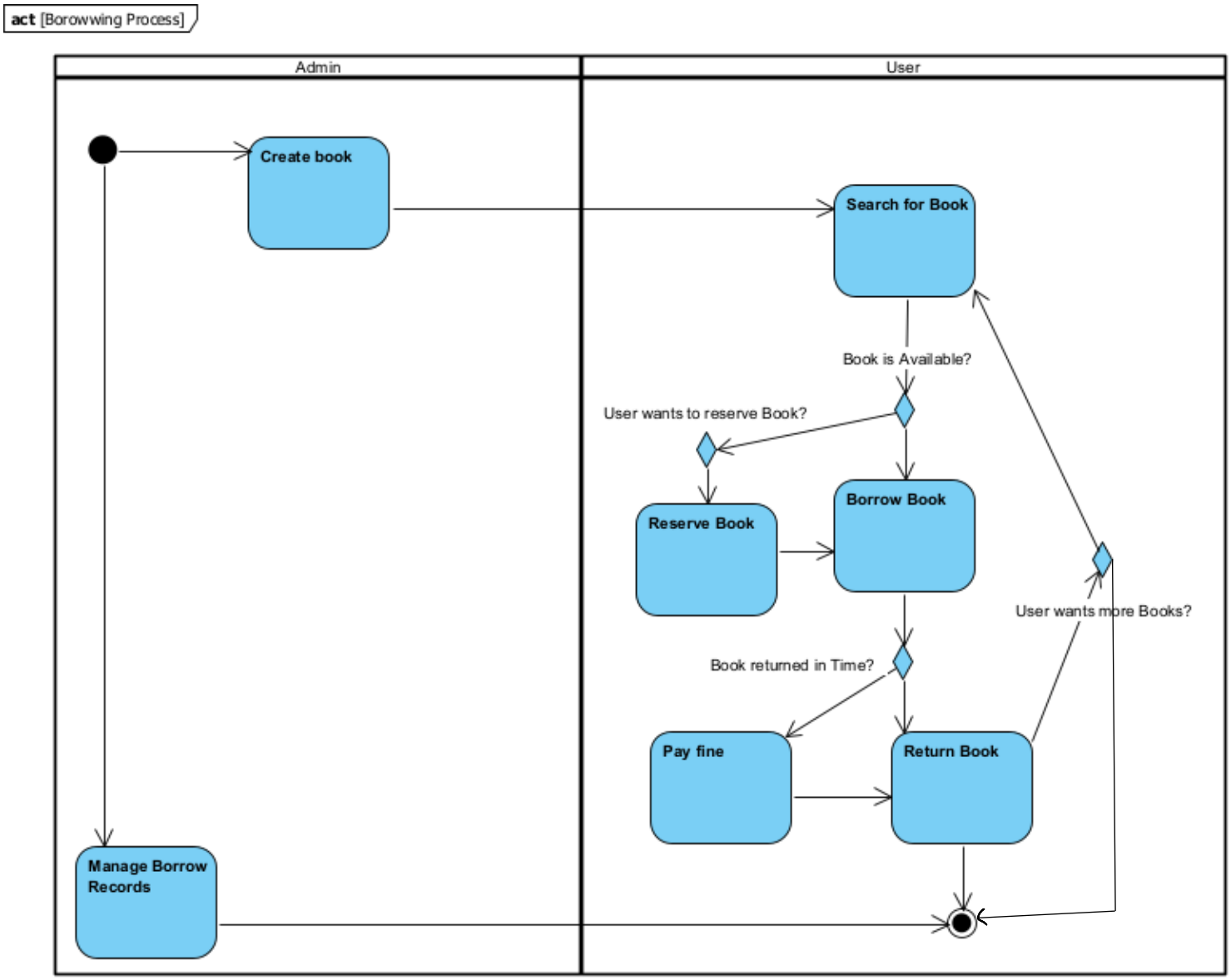
User Management Use Case Diagram:



- **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

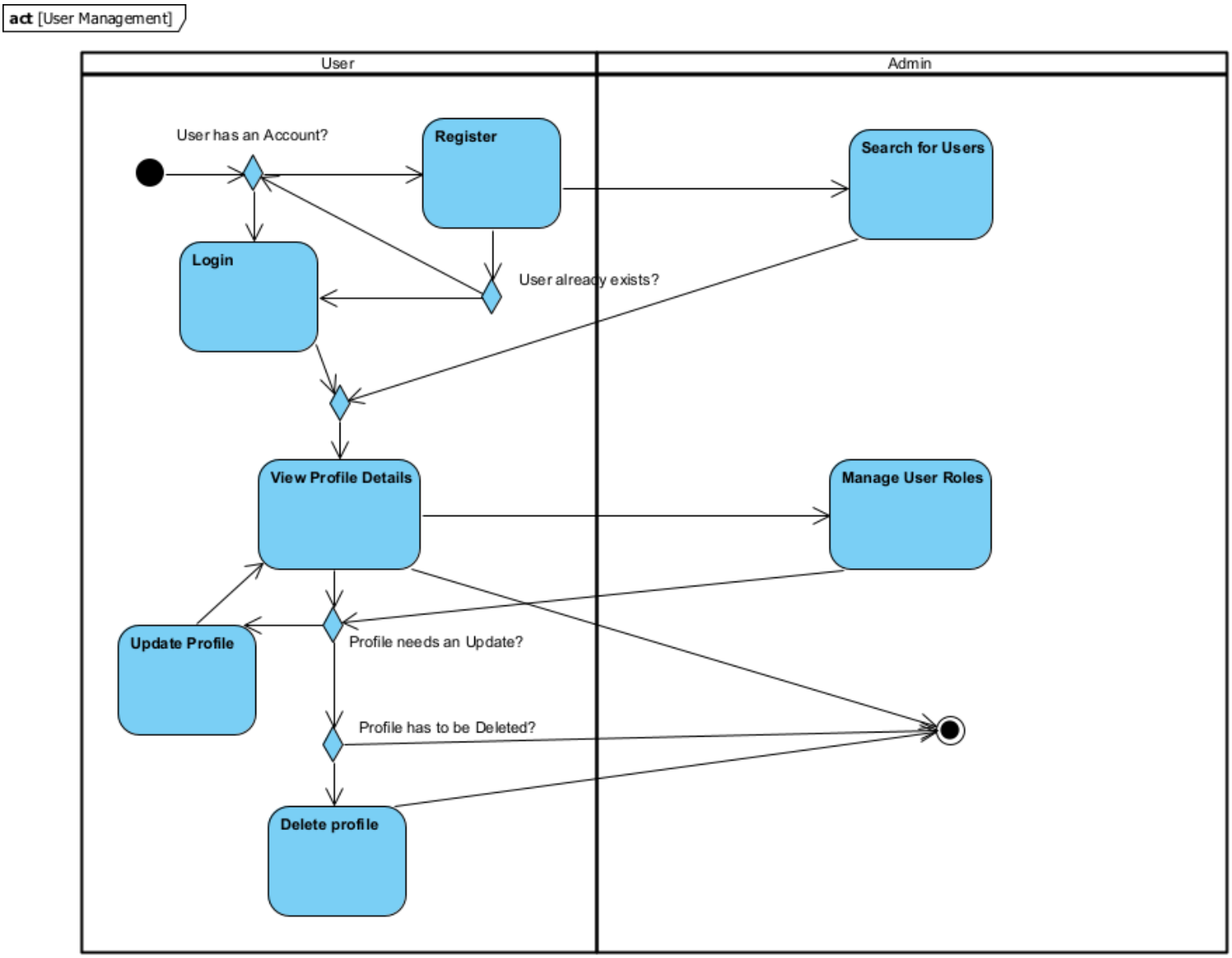
Borrowing Process



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



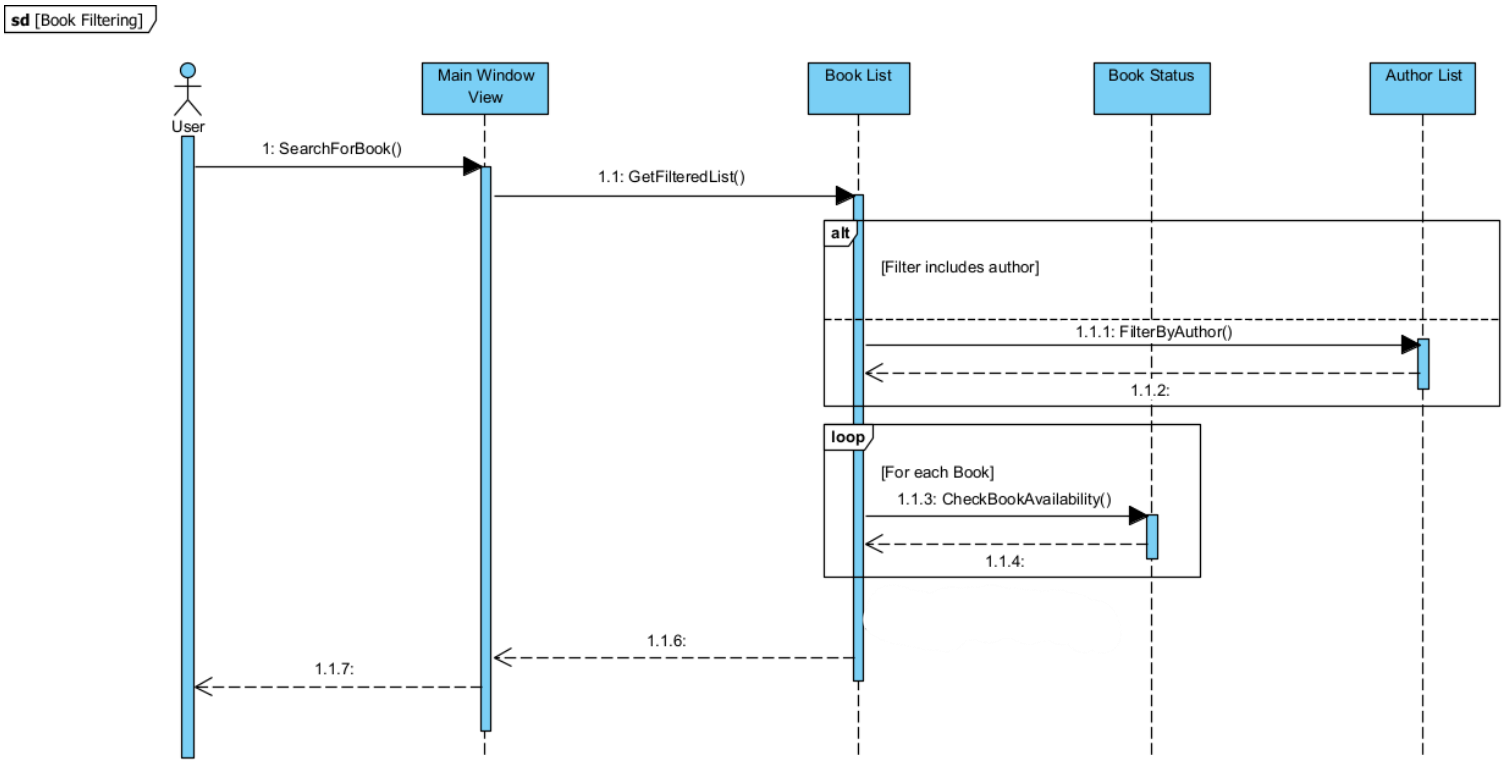
Actors: Admin, User

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.

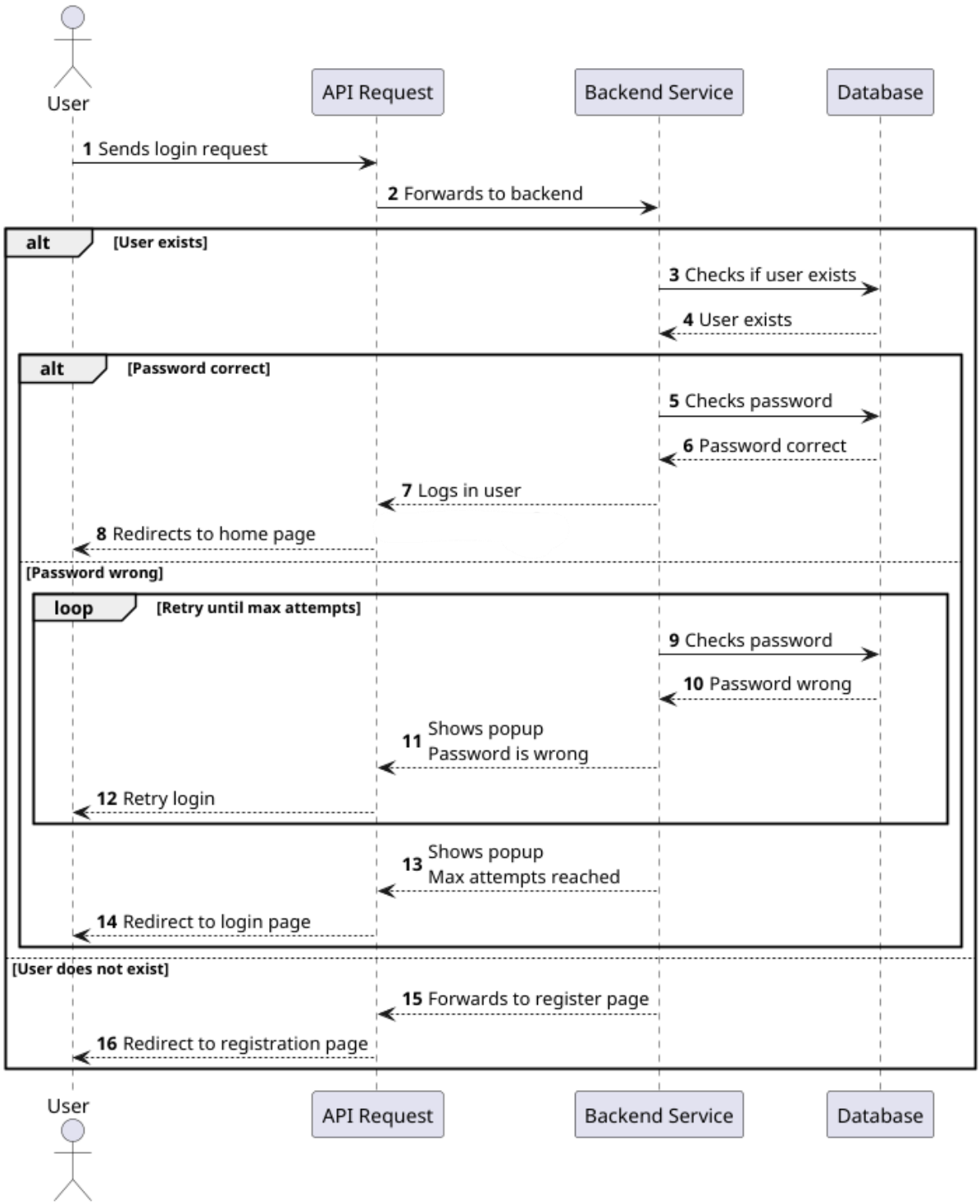
Sequence diagrams

Book Filering



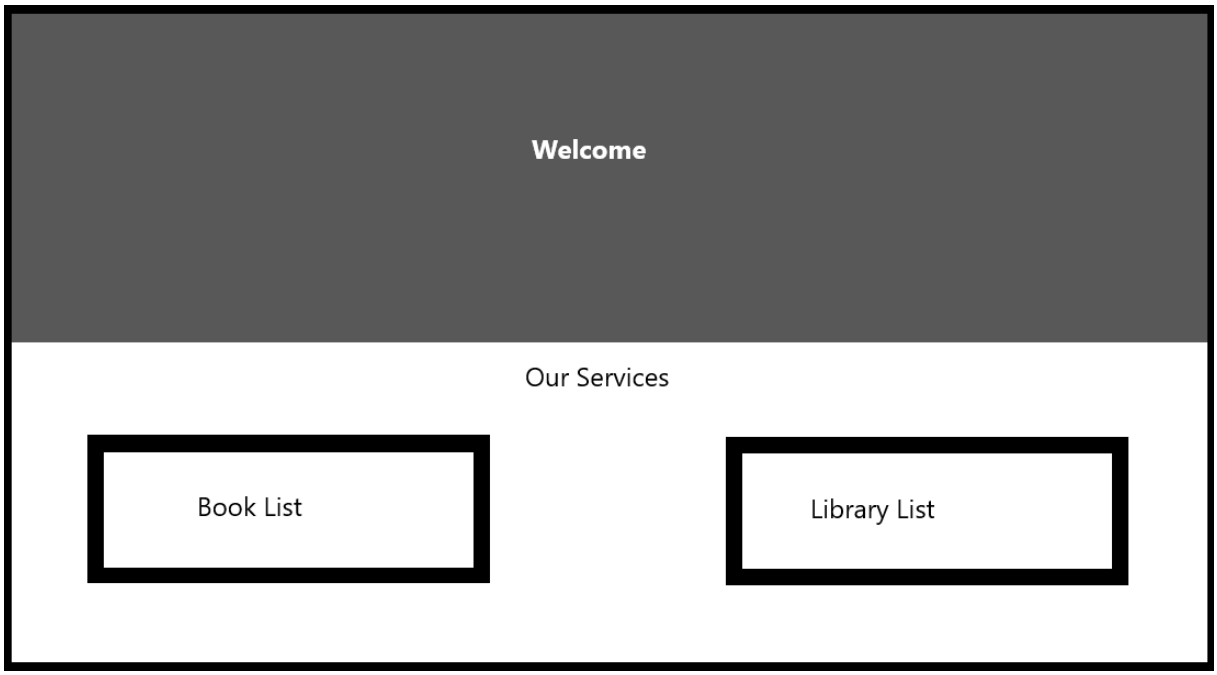
The Sequence begins with the User initiating a SearchForBook() request in the Main Window View. The Main Window View then sends a request to retrieve a filtered list of books (GetFilteredList()) to the Book List component. If the filter criteria include an author, the Book List interacts with the Author 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

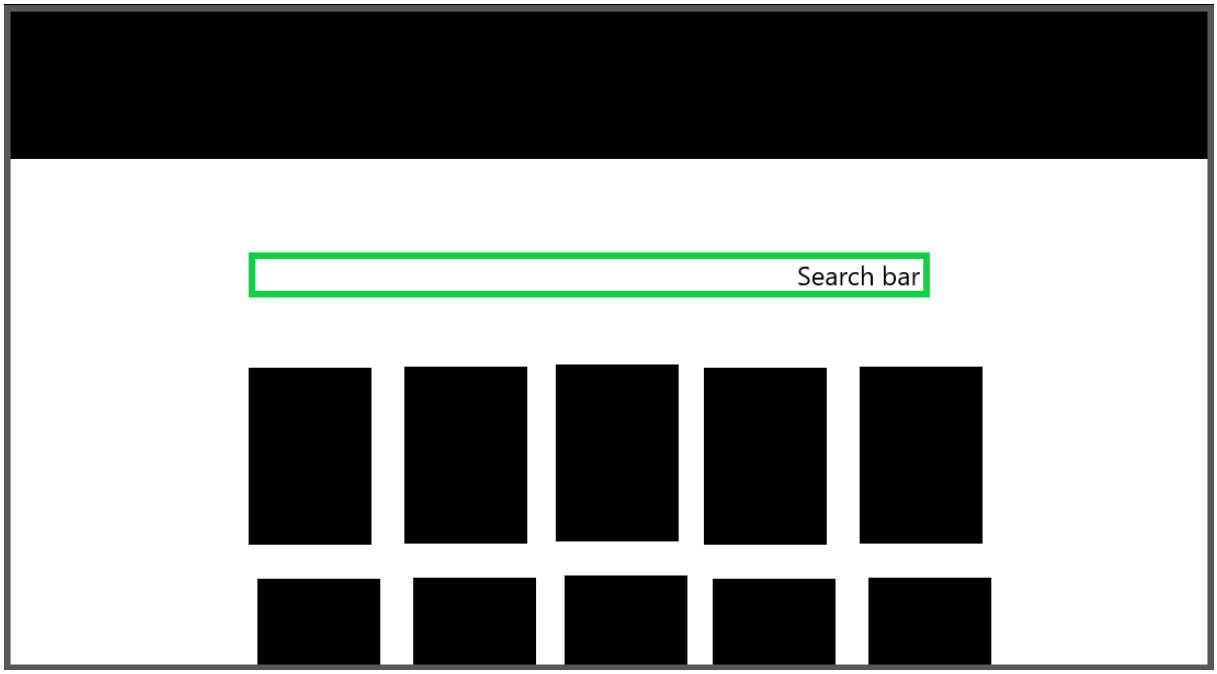

User
API Request
Backend Service
Database
1 Sends login request
2 Forwards to backend
alt
[User exists]
3 Checks if user exists
4 User exists
alt
[Password correct]
5 Checks password
6 Password correct
7 Logs in user
8 Redirects to home page
[Password wrong]
loop
[Retry until max attempts]
9 Checks password
10 Password wrong
11 Shows popup
Password is wrong
12 Retry login
13 Shows popup
Max attempts reached
14 Redirect to login page
[User does not exist]
15 Forwards to register page
16 Redirect to registration page
User
API Request
Backend Service
Database

Wireframe

Front page

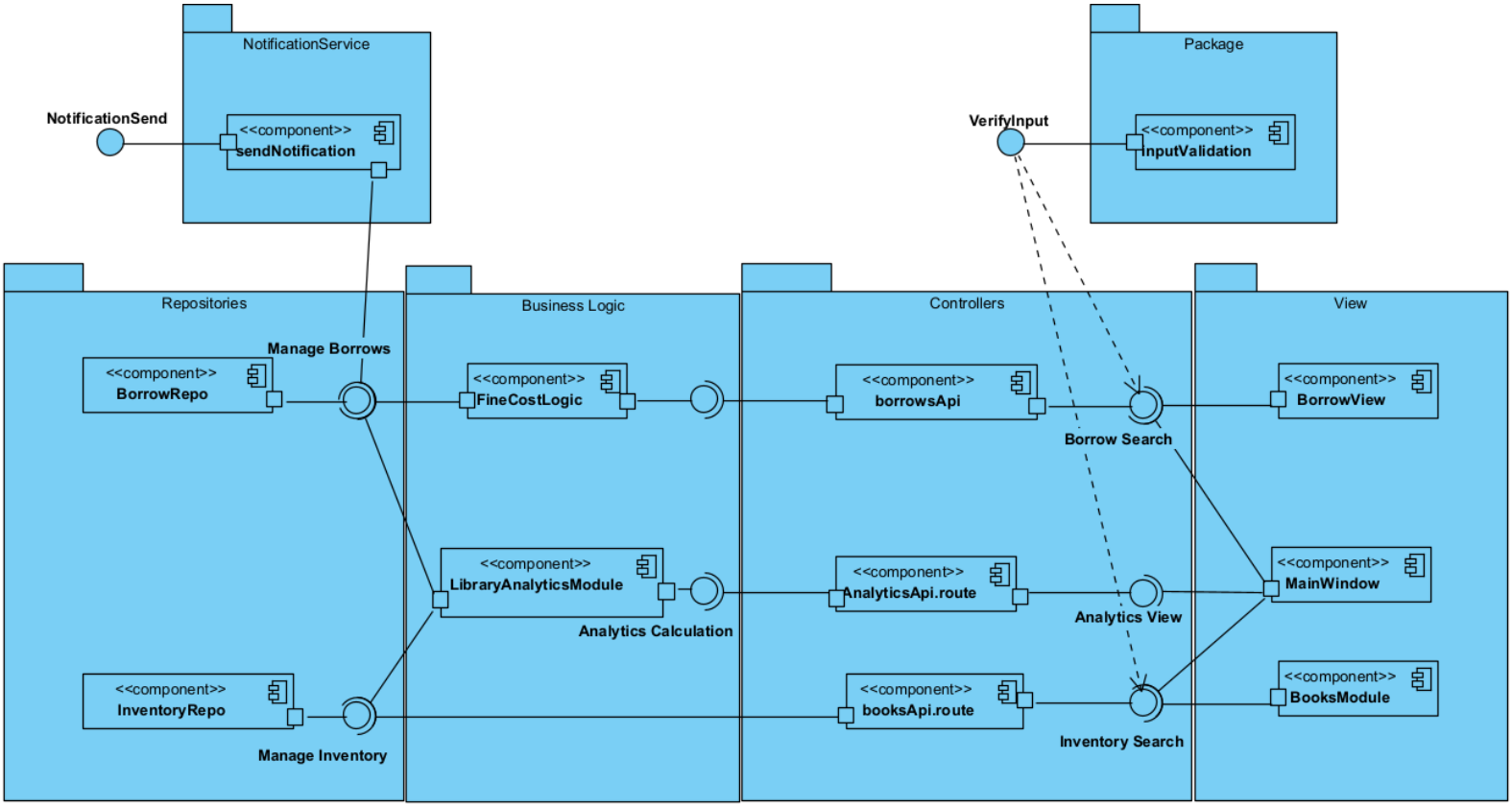


Book list page



Component diagrams

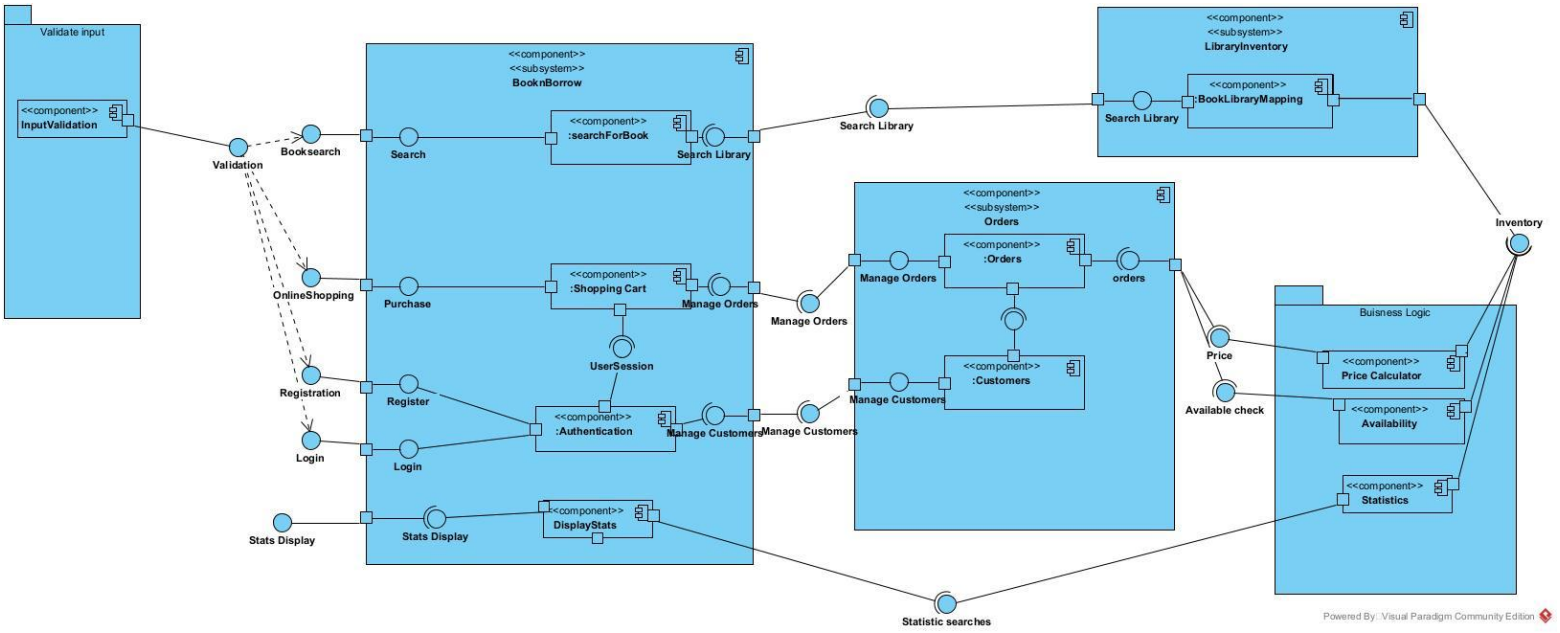
Layers Component Diagram



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

User Actions



- Input Validation
  - Validates all user inputs to ensure data integrity and proper functionality across the system.
- BooknBorrow Subsystem
  - SearchForBook: Enables users to search for books available in the library.
  - Shopping Cart: Facilitates book borrowing or purchasing activities.
  - Authentication: Manages user login functionality.
  - Register: Handles new user registrations.
  - DisplayStats: Provides statistical data for users and system administrators.
- Orders Subsystem
  - Orders: Manages customer orders, including order placement and processing.
  - Customers: Stores and manages customer-related data for personalization and order tracking.
- LibraryInventory Subsystem
  - BookLibraryMapping: Maps and retrieves book data to check library availability.
- Business Logic Subsystem
  - Price Calculator: Calculates costs for book purchases or borrowing.
  - Availability: Checks inventory to confirm book availability.
  - Statistics: Gathers and presents analytical data for the system.

Class diagram

Link to the diagram:

https://github.com/Hormone4/BooknBorrow-Library-Management/blob/main/diagrams/class/class-diagram.png

Frontend Components:

- **App Module**: Acts as the main orchestrator, connecting multiple frontend modules like BooksModule, BorrowModule, LibraryModule, etc. Handles the flow of information and state management between different components.
- Specific Frontend Modules:
  - **BooksModule**: Handles book-related operations like viewing (viewBooks()) and sending requests (sendRequest()).
  - **BorrowModule**: Manages borrowing operations like viewing (viewBorrow()) and returning borrowed items (returnBorrowed()).
  - **LibraryModule**: Manages libraries and associated books using methods like viewLibraryBooks().
  - **UserModule**: Focuses on user-related functionality like viewUserDetails() and user-related messaging (sendMessageToUser()).
  - **BookLibraryMappingModule**: Manages mappings of books to libraries with dedicated view and management functions.
  - **SharedModule**: Includes utility functionalities like toJSON() or data transformations.

Backend Controllers:

- **BooksController**: Manages books with methods like listBooks(), addBook(), and deleteBook().
- **BorrowController**: Handles borrowing operations such as borrow() and returnBorrowed() functions.
- **LibraryController**: Deals with library-specific logic like listLibraries() and addLibrary().
- **UserController**: Manages users and their details using listUsers() and updateUserDetails().
- **BookLibraryMappingController**: Handles linking books to specific libraries with methods like mapBookToLibrary() and listMappings().
- **StatisticsController**: Provides analytics functionalities such as generating monthly statistics.

Backend Repositories:

- **BooksRepository**: Interfaces with the database for book-related operations like getAllBooks() and addBook().
- **BorrowRepository**: Supports borrowing functionality with methods like getBorrows() and returnBorrowedItem().
- **LibraryRepository**: Fetches and updates library information via getAllLibraries() and updateLibrary().
- **UsersRepository**: Handles user data persistence with methods like getAllUsers() and updateUser().
- **BookLibraryMappingRepository**: Maintains mappings between books and libraries using methods like getBooksForLibrary() and addMapping().
- **StatisticsRepository**: Aggregates data for reports like getBorrowStats() or other analytics.

Domain Model:

- Classes:
    - Book: Represents a book entity with attributes like title and author.
    - Library: Encapsulates library data such as name and associated books.
    - User: Represents a user entity with attributes like name and email.
    - BookLibraryMapping: Links books to libraries.
    - Borrow: Tracks borrowed books, including details like borrowDate and returnDate.
    - FineCalculation: Manages fine computations for overdue borrowed books.
    - Statistics: Stores analytics-related data like borrow frequency.
- Relationships:
    - Libraries can have multiple books.
    - Books are linked to libraries through a mapping.
    - Users borrow books and return them, creating Borrow instances.
    - Borrow records are associated with users and books.