

## **Book Haven Library Management System Design**

The Book Haven library management system will allow users to perform actions like borrowing books, returning books, and searching for available books. The system will also manage books, users, and borrowing transactions.

### **Functional Requirements:**

- 1. User Management:**
  - Users can create an account or log in.
  - Users can view their borrowing history.
  - Users can modify their profile information.
- 2. Book Inventory Management:**
  - The system should store information on books such as title, author, genre, availability, and ISBN.
  - The system must allow administrators to add, update, or remove books.
- 3. Search Books:**
  - Users can search for books based on title, author, genre, and ISBN.
  - Search should provide relevant results based on the user's query.
- 4. Borrowing Books:**
  - Users can borrow books if they are available and not exceeding the borrowing limit.
  - The system should track the borrowed books with due dates.
  - Users should be able to view their borrowing status and return dates.
- 5. Returning Books:**
  - Users can return borrowed books once they have finished reading them.
  - The system should update the availability of returned books and keep track of late returns.
- 6. Notifications:**
  - Users should receive reminders for overdue books or approaching due dates.
  - System should notify users upon successful borrowing or returning.

### **Non-Functional Requirements:**

- 1. Scalability:**
  - The system should support a growing number of users and books.
  - Must handle spikes in activity (e.g., multiple users borrowing books simultaneously).
- 2. Performance:**
  - Quick response time for book searches, borrowing, and returning actions.
  - High availability for 24/7 access.
- 3. Security:**
  - Secure authentication and authorization for users and administrators.

- Protect sensitive user data such as login credentials.
- 4. **Usability:**
  - The UI should be intuitive and easy to navigate.
  - Simple and clear error messages for users.
- 5. **Maintainability:**
  - The system should be easy to update and maintain.
  - Clear code structure for future developers.

## **Subsystems Breakdown:**

1. **User Management:**
  - **Responsibilities:** Handle user login, registration, profile updates, and account deletion.
  - **Data:** User profiles, borrowing history, user credentials.
2. **Book Inventory:**
  - **Responsibilities:** Manage book data (title, author, ISBN, availability, genre, etc.).
  - **Data:** Books metadata, stock count, ISBN numbers.
3. **Borrowing System:**
  - **Responsibilities:** Manage book borrowing and returning, due dates, track overdue books.
  - **Data:** Borrowed books records, due dates, late fees.
4. **Notifications:**
  - **Responsibilities:** Send reminders for overdue books, borrowed status updates.
  - **Data:** Notification history for users, reminders, overdue status.

## **Parameters:**

1. **Maximum Books Borrowed per User:**
  - A user can borrow a maximum of 5 books at a time.
2. **Database Scalability Needs:**
  - Must support hundreds of thousands of books and users as the library grows.
  - Support for quick search and retrieval of book data even with high traffic.
3. **Performance Considerations:**
  - Fast response times for book searches (less than 2 seconds per search).
  - Borrowing and returning processes should be quick (less than 1 second for database update).

## Database Models Comparison: Relational vs NoSQL

### Relational Database (SQL)

- **Examples:** MySQL, PostgreSQL.
- **Strengths:**
  - Data integrity is ensured with ACID transactions.
  - Schema is well-defined, making it ideal for structured data.
  - Good for systems with complex relationships (e.g., users, books, borrow transactions).
- **Weaknesses:**
  - Less flexible when the schema needs to evolve (adding new fields might require schema migration).
  - Can struggle with scaling horizontally in distributed environments.

### NoSQL Database

- **Examples:** MongoDB, Cassandra, DynamoDB.
- **Strengths:**
  - Scales well horizontally for large amounts of data (easy to add more servers).
  - Flexible schema allows for easy changes.
  - Excellent for unstructured or semi-structured data.
- **Weaknesses:**
  - Lacks ACID guarantees (though some NoSQL systems offer eventual consistency).
  - Complex relationships (like joins) are harder to handle.

### Chosen Database Model: Relational (SQL)

Given the needs for structured data (book inventory, user profiles, transactions) and strong relationships between entities (users, books, borrowing transactions), a relational database is chosen. The benefits of ACID compliance and structured schema design make SQL a strong choice for maintaining consistency and data integrity.

### Why SQL:

- **Data Relationships:** The system needs to manage relationships (e.g., one-to-many between users and borrowed books). SQL databases are well-suited for these scenarios.
- **Scalability Needs:** With proper indexing and optimization, a relational database can scale vertically and handle the expected load.
- **Complex Queries:** SQL allows for complex queries like tracking overdue books, filtering by book attributes, and generating borrowing history.

## High-Level Prototype UI for Book Search and Borrowing

### 1. Home Page:

- **Search Bar:** Input field for searching by title, author, genre, or ISBN.
- **Filter Options:** Dropdowns for genre, availability, and sorting by popularity or due dates.
- **Search Results:** Display a list of available books with brief details (title, author, genre, availability).

### 2. Book Details Page:

- Display detailed information about the book (summary, author, genre, ISBN).
- Button for borrowing the book if available.
- Display the number of available copies.

### 3. Borrowing Page:

- **Borrow Book Button:** If the user is eligible (not exceeding max borrow limit).
- Show due date and a confirmation button for borrowing.

### 4. My Account Page:

- Show a list of borrowed books, due dates, and overdue books.
- Option to return books.
- Notifications for overdue books.

## Example UI Sketch:

### 1. Search Screen:

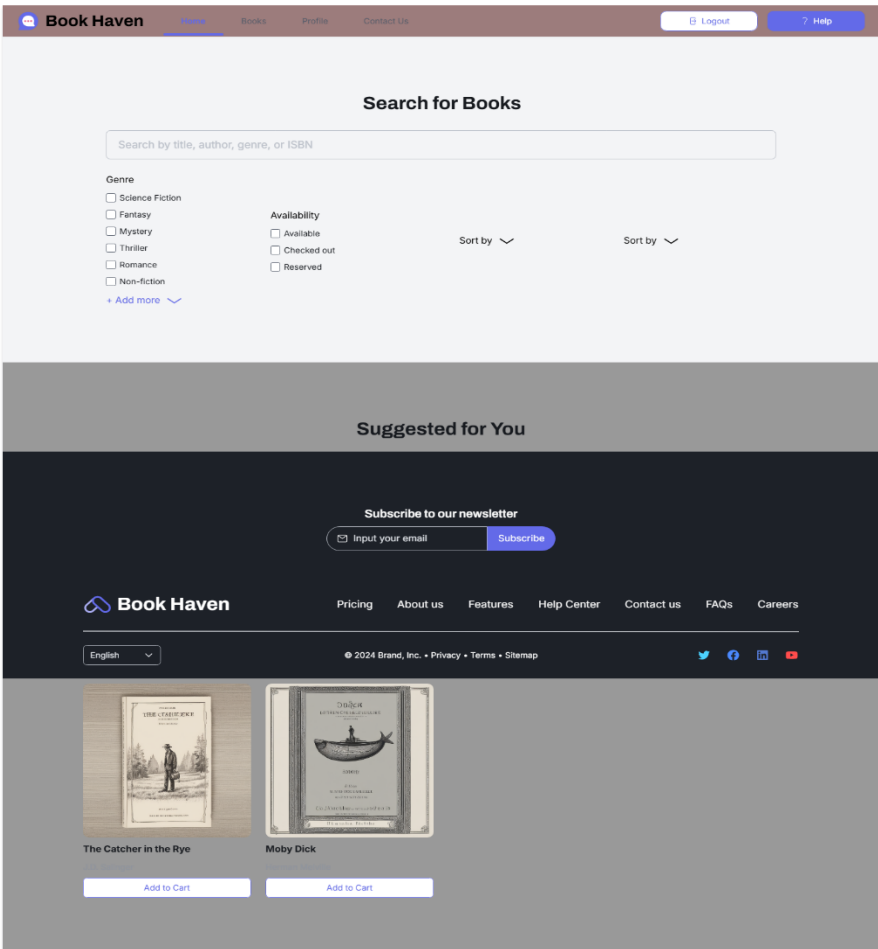
- Top search bar with "Search for books"
- Filters like "Genre", "Author", and "Availability"
- Results section showing book title, author, availability status, and borrow button.

### 2. Book Details Screen:

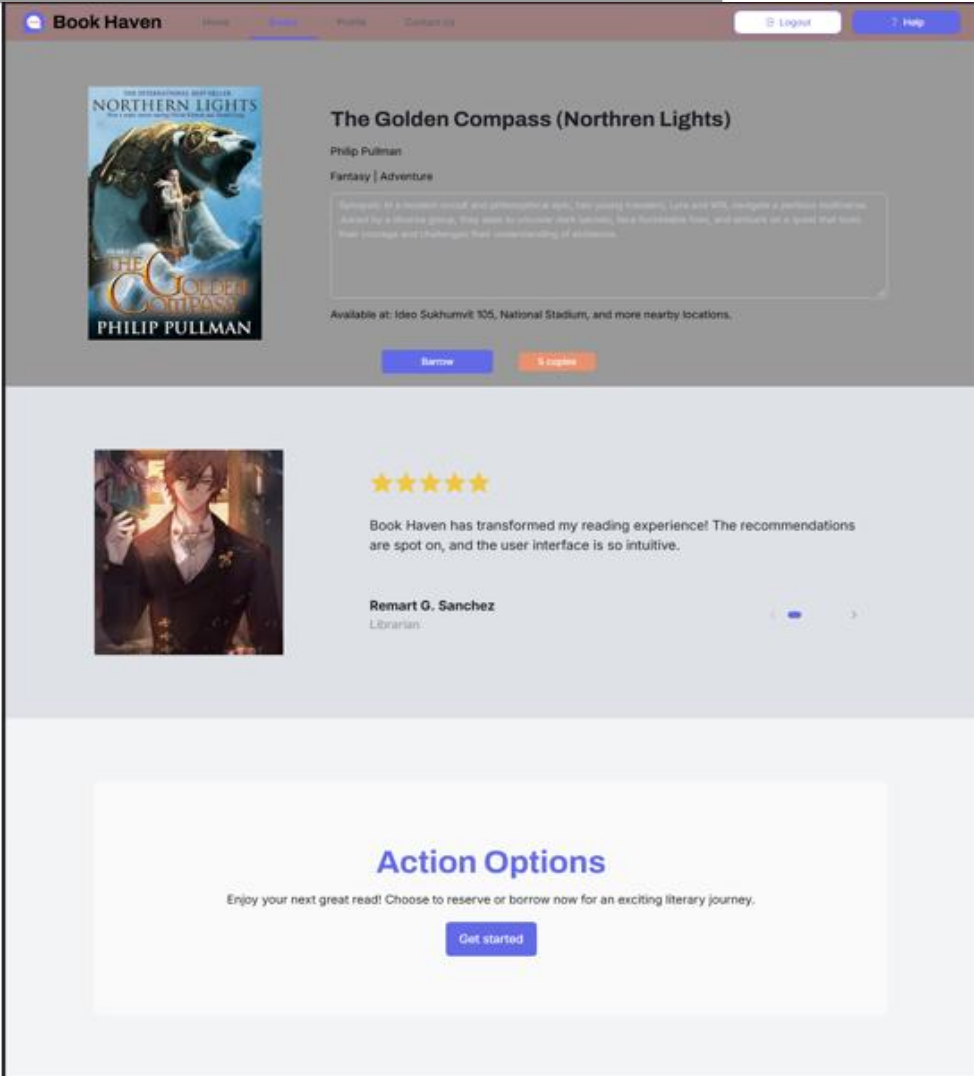
- Large book cover image, title, author
- Book details (genre, ISBN, summary)
- Borrow button (enabled if available)

### 3. Borrowing Screen:

- List of books being borrowed
- Due date displayed clearly
- Button to confirm borrowing.



Made with Visily



Book Haven

HomeBooksProfileContact Us

Janice P. Tabaco

Following

Message

Edit Profile

Borrowed Books

The Lord of the Rings : The Fellowship of the Ring

Author: J.R.R. Tolkien

Due Date: 2024-01-30

Return

Extend

1984

Author: George Orwell

Due Date: 2023-11-28

Return

Extend

The Time Machine

Author: H.G. Wells

Due Date: 2024-05-01

Return

Extend

Reservation History

Title	Reservation Date	Status
The Psychology of Money	02/11/2023	Returned
The Book Thief	09/10/2023	Returned
Educated	18/09/2023	Returned
Atomic Habits	23/08/2023	Returned

Personal Preferences

Email Notification

Choose your limits: Current library loan limits, etc.


Save changes

Logout

Help

Get in Touch

Fill in the form to get in touch.



First name

Input text

Last name

Input text

Email

Input text

Company name

Input text

Company size

Please select

Which topic best fit your needs?

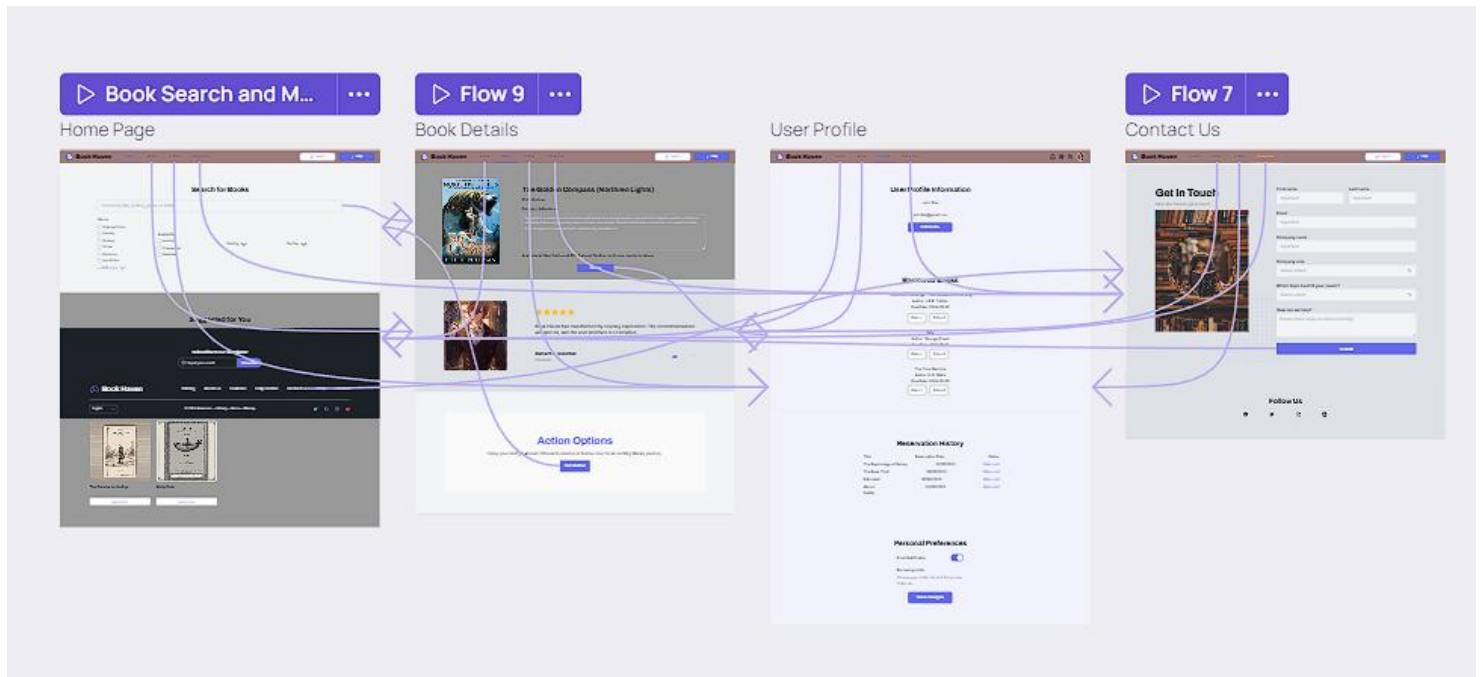
Please select

How can we help?

Please share what you want us to help

Submit

Follow Us



## Use Case Diagram (Identify Key Actors and Interactions)

A **Use Case Diagram** shows the key actors (users) interacting with the system and the functionalities (use cases) they can perform. In this LMS, the key actors and their use cases would be:

### Key Actors:

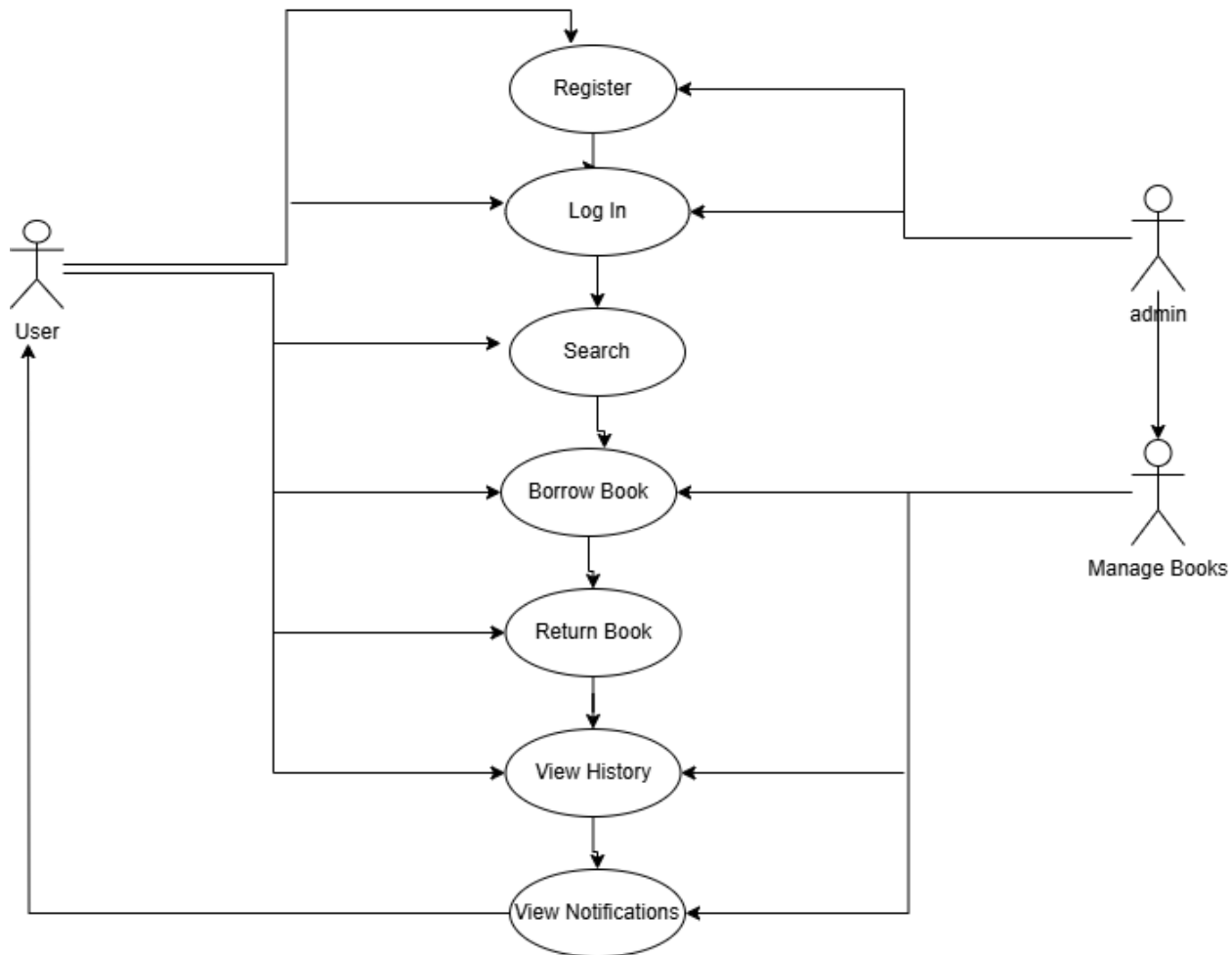
- **User:** The person who registers, logs in, borrows, and returns books.
- **Admin:** The person responsible for managing books (adding, updating, deleting).
- **System:** Manages notifications, user management, and data processing.

### Use Cases:

- **User:**
  - Register
  - Log in
  - Search for Books
  - View Borrow History
  - Borrow Books
  - Return Books
  - View Notifications
- **Admin:**
  - Add, Update, or Delete Books

- View All Borrowed Books and History

### Use Case Diagram:



### Class Diagram (Define Classes: User, Book, Loan)

The **Class Diagram** defines the structure of the system in terms of objects (classes), their attributes, and methods.

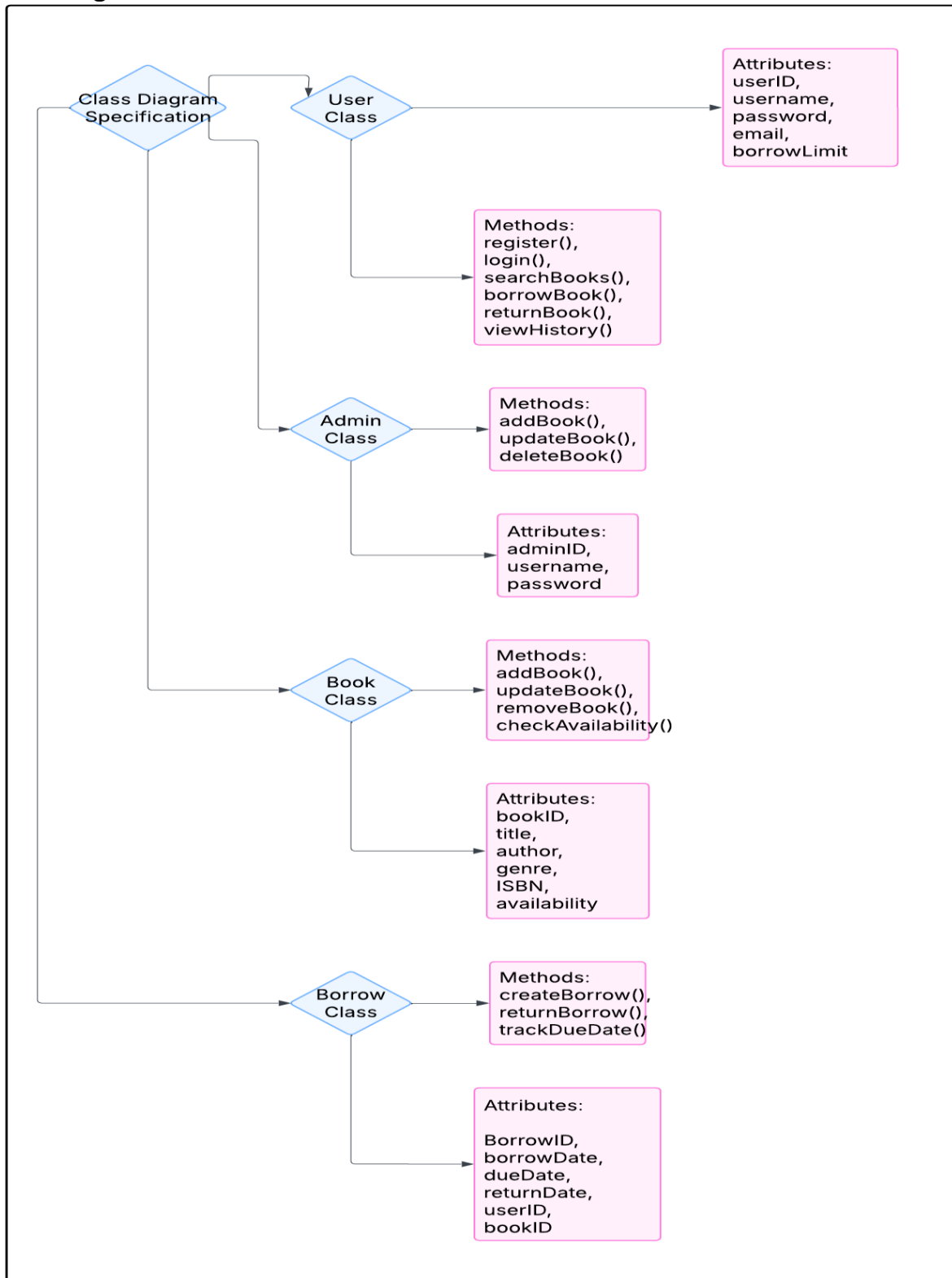
#### Classes:

- **User**
  - Attributes: userID, username, password, email, borrowLimit
  - Methods: register(), login(), searchBooks(), borrowBook(), returnBook(), viewHistory()
- **Book**
  - Attributes: bookID, title, author, genre, ISBN, availability



- Methods: addBook(), updateBook(), removeBook(), checkAvailability()
- **Borrow**
  - Attributes: BorrowID, borrowDate, dueDate, returnDate, userID, bookID
  - Methods: createBorrow(), returnBorrow(), trackDueDate()
- **Admin**
  - Attributes: adminID, username, password
  - Methods: addBook(), updateBook(), deleteBook()

### Class Diagram:



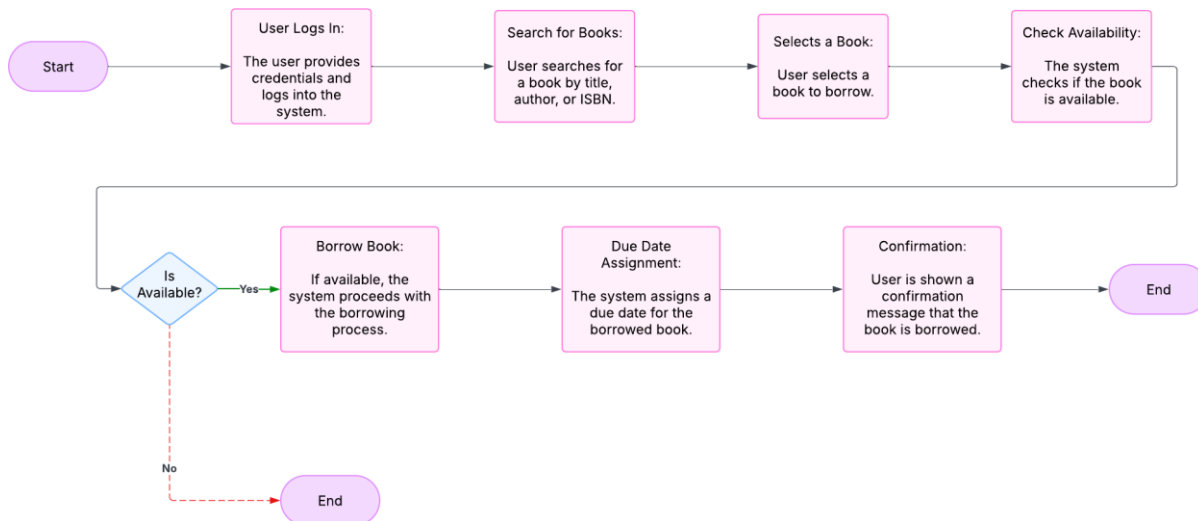
## Activity Diagram (Book Borrowing Process)

The **Activity Diagram** shows the flow of actions involved in borrowing a book.

### Steps in the Borrowing Process:

1. **User Logs In:** The user provides credentials and logs into the system.
2. **Searches for Books:** User searches for a book by title, author, or ISBN.
3. **Selects a Book:** User selects a book to borrow.
4. **Check Availability:** The system checks if the book is available.
5. **Borrow Book:** If available, the system proceeds with the borrowing process.
6. **Due Date Assignment:** The system assigns a due date for the borrowed book.
7. **Confirmation:** User is shown a confirmation message that the book is borrowed.

Activity Diagram:



## Component Diagram (System Components and Their Interactions)

A **Component Diagram** shows the system's components and their interactions.

### Components:

- **User Interface:** Manages user interactions such as logging in, searching for books, and viewing the borrow history.
- **Authentication Service:** Handles login, user registration, and authentication.

- **Book Management Service:** Handles adding, updating, deleting books, and checking availability.
- **Loan Management Service:** Handles book borrowing and returning processes.
- **Notification Service:** Sends reminders to users for overdue books.

### Component Diagram:

