# INTRODUCTION TO PROJECT

‘Rento-Book application’ aims to streamline library operations, enhance user experience, and promote efficient access to resources for patrons. Automating the whole process for a physical library is our main aim which will solve all the problems on maintaining records physically on to the book. Additionally, our application provide functionality to view events along with the details of the events taking place.

A Rento-Book Application is a software application that simplifies and automates the operations of libraries. It is a complete system for managing library duties such as purchases, member management, monitoring, storing, and circulation. The primary objective of an Application is to properly organize and manage the resources available in a library, making it easier for librarians to conduct everyday operations and create a user-friendly experience for users**.**

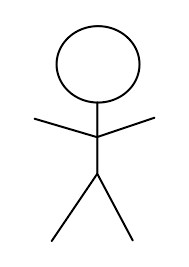
Conventional libraries are having difficulty integrating various formats, including multimedia and e-resources, because of outdated management systems. Inefficient cataloguing, resource tracking bottlenecks, and a lack of analytics tools hinder librarians from optimizing collections and improving user experiences. To close the gap, libraries require a modern library management system with an intuitive interface, effective cataloguing, and analytics capabilities to resurrect libraries as vibrant centres of knowledge and community involvement in the digital era.

# Solution:

To solve the traditional issue we are building a **Web development**project of library management system using **Html**,**Bootstrap** , **Java**and **MYSQL**in which we will be providing User-friendly interface for easy navigation , Efficient book search functionality , seamless book issuance and return policy , automated tracking of library activities, Regular maintenance of book availability records and Secure login and access control managed by the admin.

**2.REQUIREMENTS**

**2.1 FUNCTIONAL REQUIREMENTS**

****

USER

**2.1 User Account**

The customers, who will henceforth be called the ‘user’, will be presented with 3 choices by the Rento-Book system, as the first step in the interaction between them.

This feature used by the user to login into system. They are required to enter user id and password before they are allowed to enter the system. The user id and password will be verified and if invalid id is there user is allowed to not enter the system**.**

**Functional requirements** –

user id is provided when they register -The system must only allow user with valid id and password to enter the system -The system performs authorization process which decides what user level can access to. -The user must be able to logout after they finished using system.

**2.2 Registration and creation of user profile**

**Description of feature:**

This feature can be performed by all users to register new user to create account.

**Functional requirements –**

System must be able to verify information -System must be able to delete information if information is wrong ****

* 1. **Quick Search**

Here we provided Quick Search facility for any user to search Books schedule without login into account .This will provide user an option for searching Books and comparing their prices of all categories.

After a user logs into their account in a library management system like Rento-Book, several actions and functionalities typically become available.

* **Post-Login Actions:**

1. **User Dashboard Access:**
   * The user is redirected to their personalized dashboard or home page. This dashboard often displays relevant information such as current borrowed books, upcoming due dates, reserved books, and account details.
2. **Account Overview:**
   * Users can view and update their profile information, including personal details, contact information, and password settings.
3. **Book Search and Management:**
   * **Search Functionality:** Users can perform searches for books in the library catalog based on various criteria such as title, author, genre, or ISBN.
   * **Advanced Search:** If available, users can use advanced search options to refine their queries, such as filtering by availability, publication date, or format (e.g., hardcover, ebook).
4. **Borrowing and Returning Books:**
   * **Check Availability:** Users can check the availability of books in real-time. If a book is available, they can proceed to borrow it.
   * **Borrow Books:** Users can select books to borrow, review borrowing terms, and confirm their selection. The system updates the library records to reflect the new status of the borrowed books.
   * **Return Books:** Users can return borrowed books through the system. The system will update the status of the books and adjust the user's borrowing history accordingly.
5. **Reservations and Requests:**
   * **Reserve Books:** Users can reserve books that are currently checked out. Once the book is returned to the library, the user will be notified and can pick it up.
   * **Request New Books:** If the system supports it, users can request the acquisition of new books or suggest books for addition to the library’s collection.
6. **View and Manage Bookings:**
   * Users can view their current and past bookings, including information about the books they have borrowed or reserved. This section often includes details like due dates, overdue fines, and the status of reservations.
7. **Event Viewing:**
   * **Upcoming Events:** Users can view details about library events, workshops, or reading programs. This may include event descriptions, dates, times, and registration options if applicable.
8. **Notifications and Alerts:**
   * **Due Date Reminders:** The system may send notifications about upcoming due dates for borrowed books.
   * **Overdue Notices:** Users will receive alerts if any of their borrowed books are overdue, including information on any fines incurred.
9. **Account and Transaction History:**
   * Users can access a history of their transactions, including borrowing history, returns, and reservations. This section helps users keep track of their interactions with the library.
10. **Admin Features (for Admin Users):**
    * **Manage Library Inventory:** Admins can add, update, or delete book records and manage book categories.
    * **Manage User Accounts:** Admins can view and manage user profiles, including resetting passwords or deactivating accounts if necessary.
    * **Generate Reports:** Admins can generate reports on library usage, book circulation, and user activities to analyze the performance and needs of the library.



Admin Features

* **Non-Functional Requirements for Rento-Book Library Management System**

2.2.1 Interface

* User Interfaces:
  + The user interfaces for Rento-Book should be intuitive and user-friendly, designed with modern web standards. The application should be responsive and accessible across various devices including desktops, tablets, and smartphones.
  + Interfaces should include:
    - Homepage: Displaying library highlights, search options, and quick links to various features.
    - Login/Registration Page: Secure and straightforward forms for user authentication and account creation.
    - Search and Browse: Functionalities for users to search for books, view categories, and browse library collections.
    - Borrow/Return Books: Interfaces for users to borrow and return books, including confirmation and status updates.
    - User Dashboard: Personalized user profile, borrowing history, and current reservations.
    - Admin Panel: For managing users, books, and system settings, including add, update, and delete functionalities.

2.2.2 Performance

* Number of Concurrent Users:
  + Rento-Book should be capable of handling a minimum of 500 concurrent users. This is based on expected usage patterns, taking into account the peak times and overall system load.
* System Response Time:
  + The system should ensure that most user actions (such as searching for books, borrowing, and returning) are completed within 2 seconds under normal operating conditions.
* Scalability:
  + The system should be designed to scale horizontally to accommodate increased loads, allowing additional servers or resources to be added as needed.

2.2.3 Constraints

* Scalability:
  + The system must handle up to 1000 transactions/inquiries per second to accommodate peak usage and ensure smooth operation during high demand.
* Database Constraints:
  + The MySQL database should be optimized for performance, with indexing and query optimization to handle large volumes of data and complex queries efficiently.

2.2.4 Other Requirements

* Hardware Interfaces:
  + Server Specifications:
    - Minimum: Intel i5 or equivalent processor, 8 GB RAM, 500 GB SSD.
    - Recommended: Intel i7 or higher processor, 16 GB RAM, 1 TB SSD for better performance and scalability.
* Software Interfaces:
  + Operating Systems: The application should be compatible with modern operating systems including Windows 10/11, macOS, and popular Linux distributions.
  + Web Server: Apache Tomcat 9.0 or later for running the Java-based application.
  + Database: MySQL 8.0 or later, ensuring compatibility with the data model and supporting efficient data retrieval and storage.
  + Browsers: The application should be compatible with the latest versions of major web browsers including Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari.
  + Libraries and Frameworks:
    - Java Spring Boot for the backend development.
    - Bootstrap for responsive design.
    - HTML5/CSS3 for structuring and styling web pages.

These non-functional requirements will help ensure that the Rento-Book library management system is robust, scalable, and user-friendly, providing a seamless experience for both users and administrators.

* **3. Design**

**3.1 Database Design**

The database design for the Rento-Book library management system involves several tables to manage users, books, transactions, and other related data. Below is the detailed structure for each table:

**-- Create User table with Role column**

**CREATE TABLE User (**

**UserID INT AUTO\_INCREMENT PRIMARY KEY,**

**Username VARCHAR(255) UNIQUE NOT NULL,**

**Password VARCHAR(255) NOT NULL,**

**Name VARCHAR(255) NOT NULL,**

**Email VARCHAR(255) UNIQUE NOT NULL,**

**PhoneNumber VARCHAR(20),**

**Address VARCHAR(255),**

**Role ENUM('User', 'Admin') NOT NULL DEFAULT 'User'**

**);**

**-- Create Book table**

**CREATE TABLE Book ( BookID INT AUTO\_INCREMENT PRIMARY KEY, Title VARCHAR(255) NOT NULL,**

**AuthorID INT, GenreID INT,**

**ISBN VARCHAR(20) UNIQUE,**

**PublishedDate DATE,**

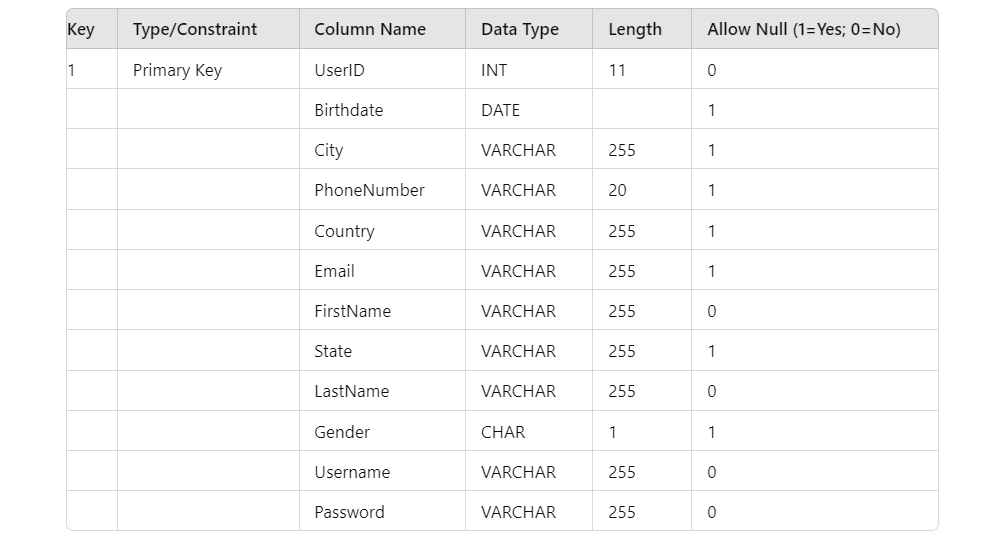
**AvailableCopies INT NOT NULL,**

**TotalCopies INT NOT NULL,**

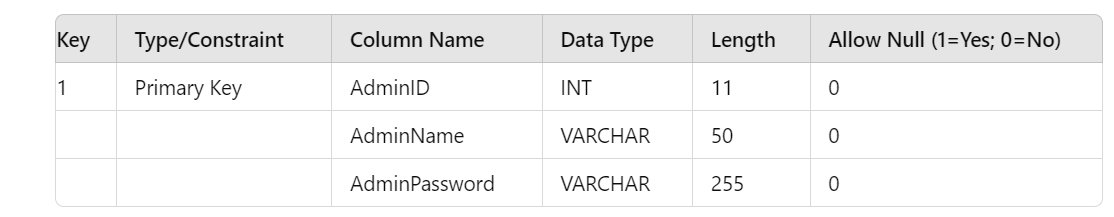
**FOREIGN KEY (AuthorID) REFERENCES Author(AuthorID),**

**FOREIGN KEY (GenreID) REFERENCES Genre(GenreID) );**

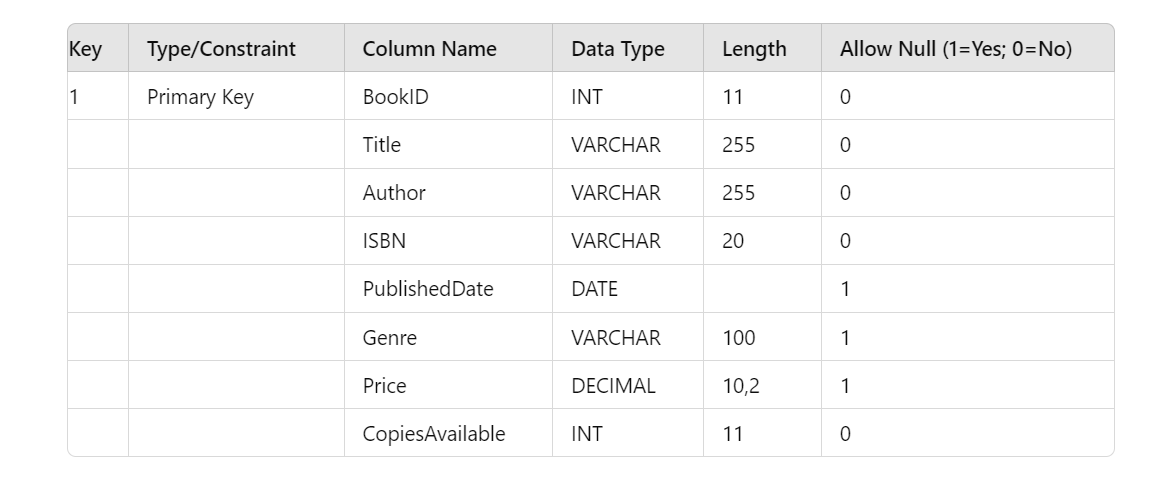
**Table 1: User-Info**



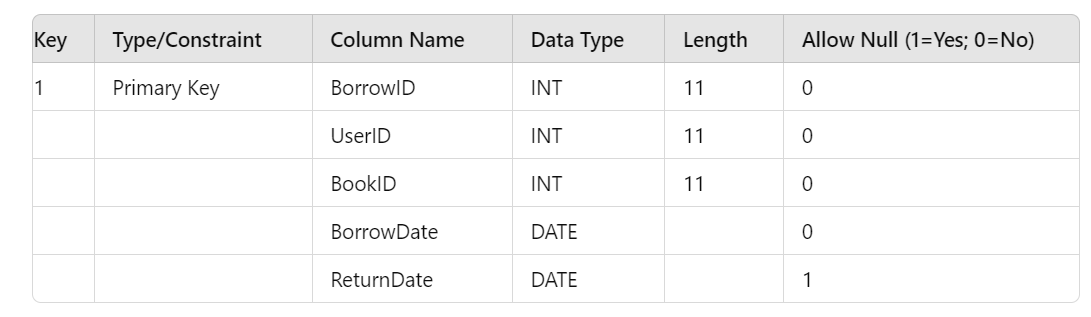
**Table 2: Administrator-Login**



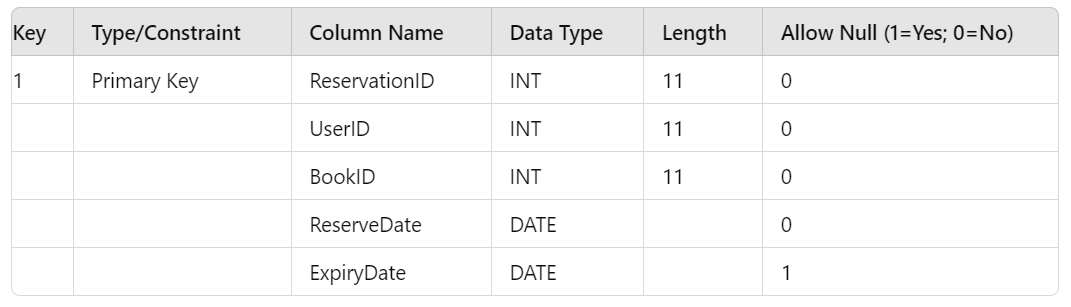
**Table 3: Book**



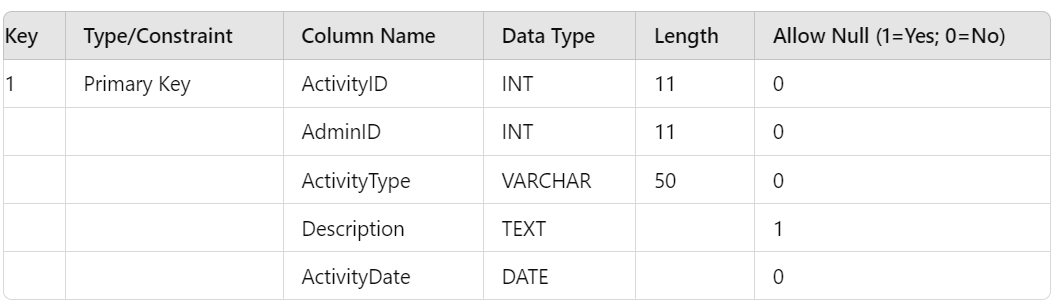
**Table 4: Borrowing**



**Table 5: Reservation**

****

**Table 6: Admin-Activity**



**E-R Diagram,Dataflow diagram and Class Diagram:**

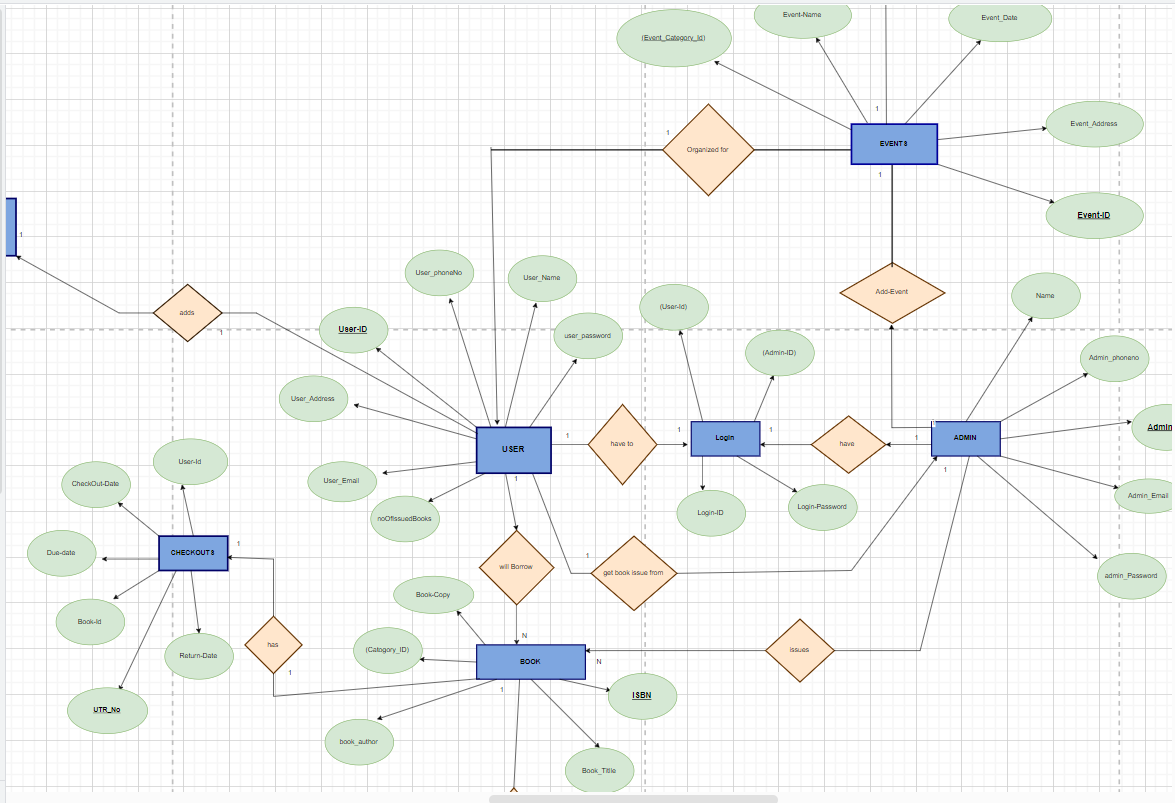


Fig: ER-Diagram

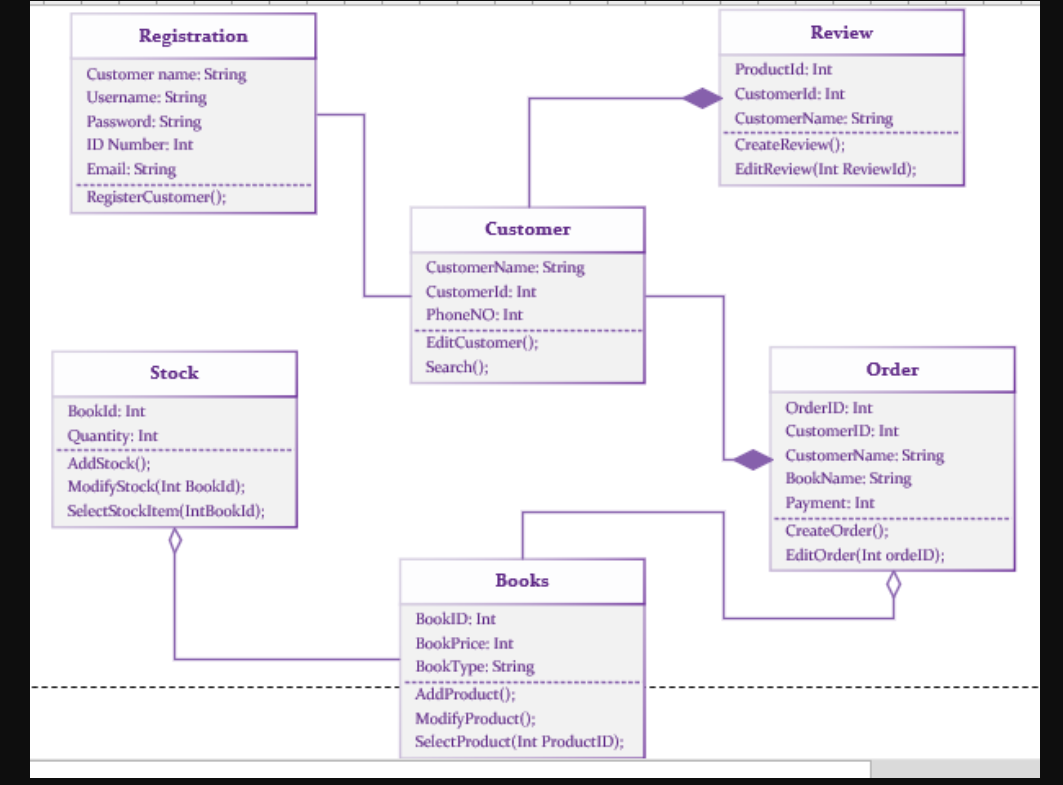


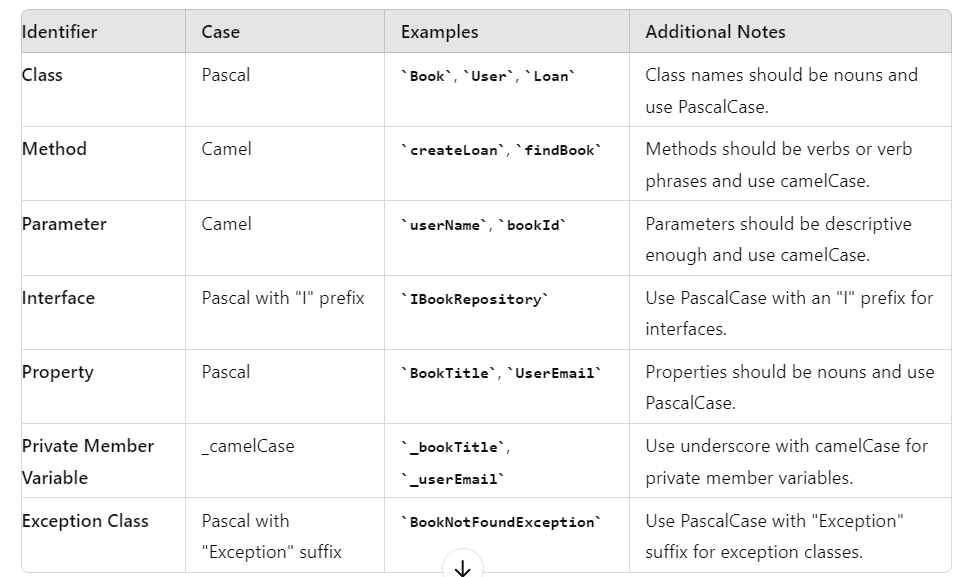
Fig: Class-Diagram

**4. CODING STANDARDS IMPLEMENTED**

### Naming and Capitalization

Below summarizes the naming recommendations for identifiers in Pascal casing is used mainly (i.e. capitalize first letter of each word) with camel casing (capitalize each word except for the first one) being used in certain circumstances.

**Below summarizes the naming recommendations for identifiers:**

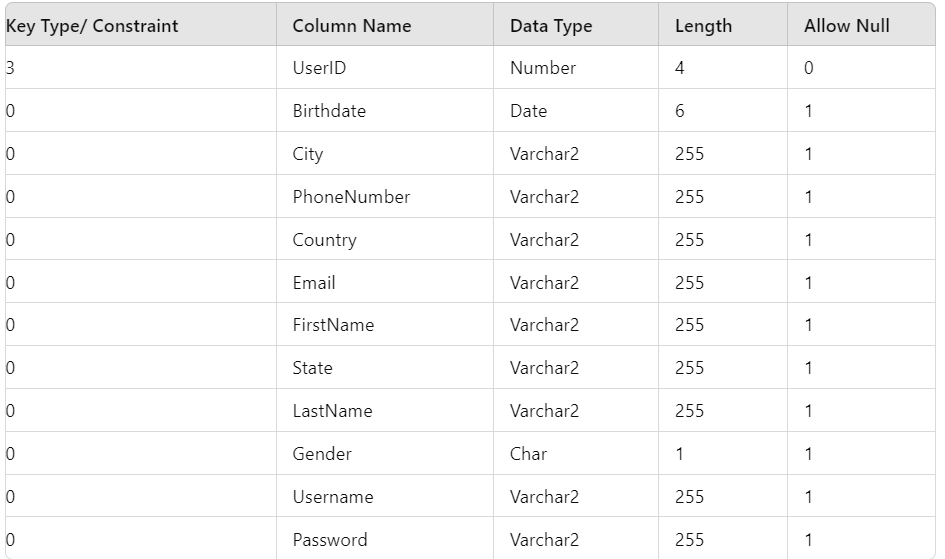
****

**Comments**

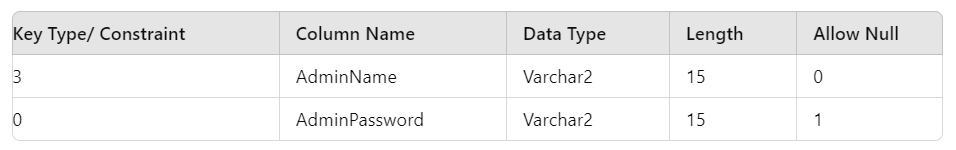
* **Comment each type, each non-public type member, and each region declaration.**
* **Use end-line comments only on variable declaration lines.**
* **Separate comments from comment delimiters with one space.**
* **Begin the comment text with an uppercase letter.**
* **End the comment with a period.**
* **Explain the code; do not repeat it.**
* **3. DATABASE DESIGN**

**Table Structures**

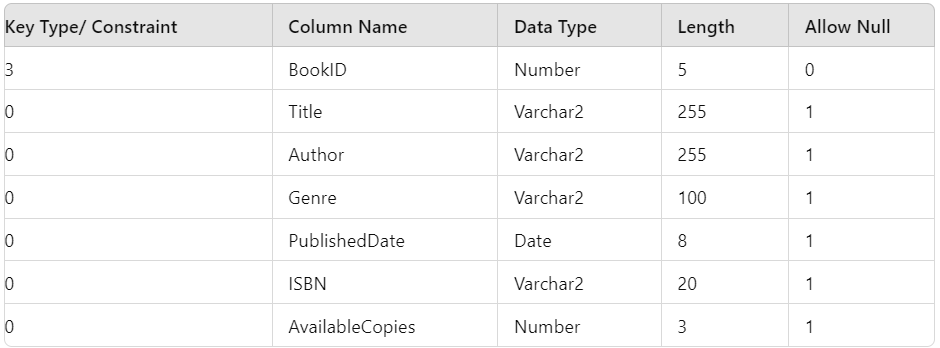
1. **User\_Info**

****

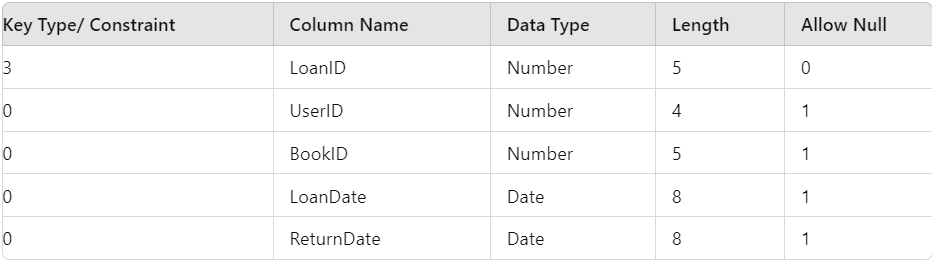
1. **Administrator\_Login**

****

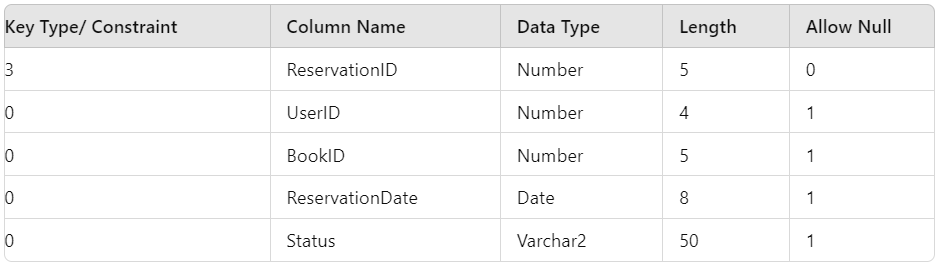
1. **BookMaster**



1. **Loan**

****

1. **Reservation**

****

**5. TEST REPORT**

**Another group called Linux did the testing and the report of the testing is given hereunder.**

**GENERAL TESTING:**



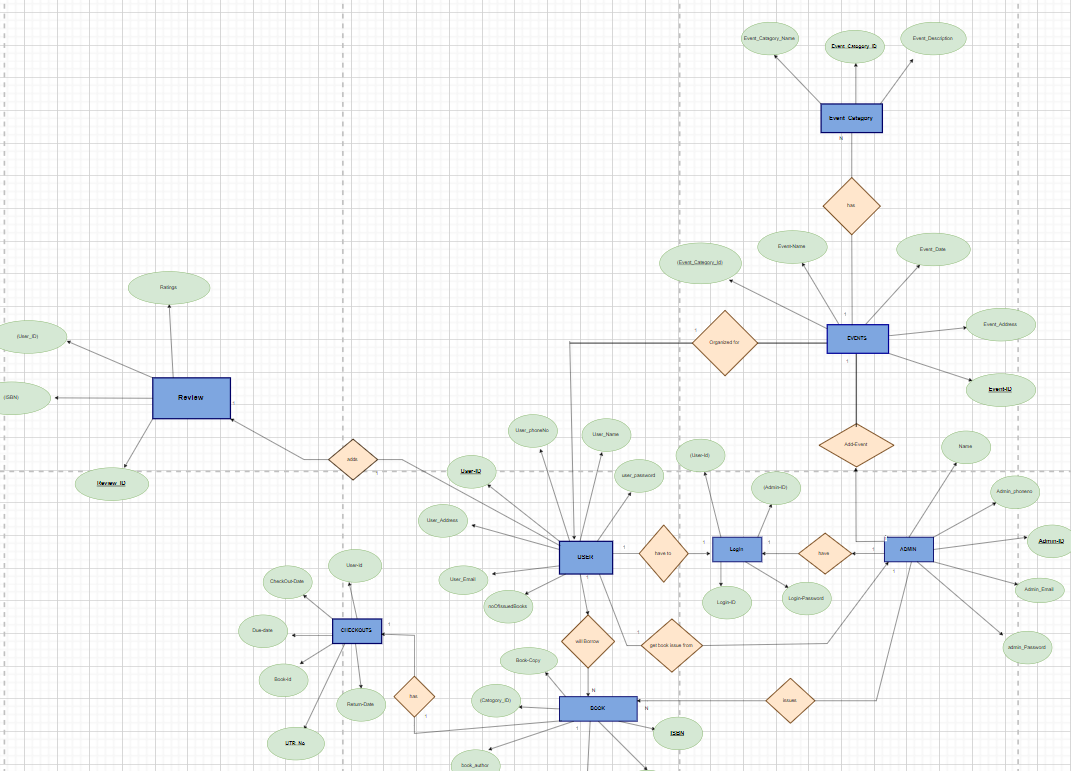
**STATIC TESTING:**

****

1. **6. PROJECT MANAGEMENT RELATED STATISTICS**
2. ****

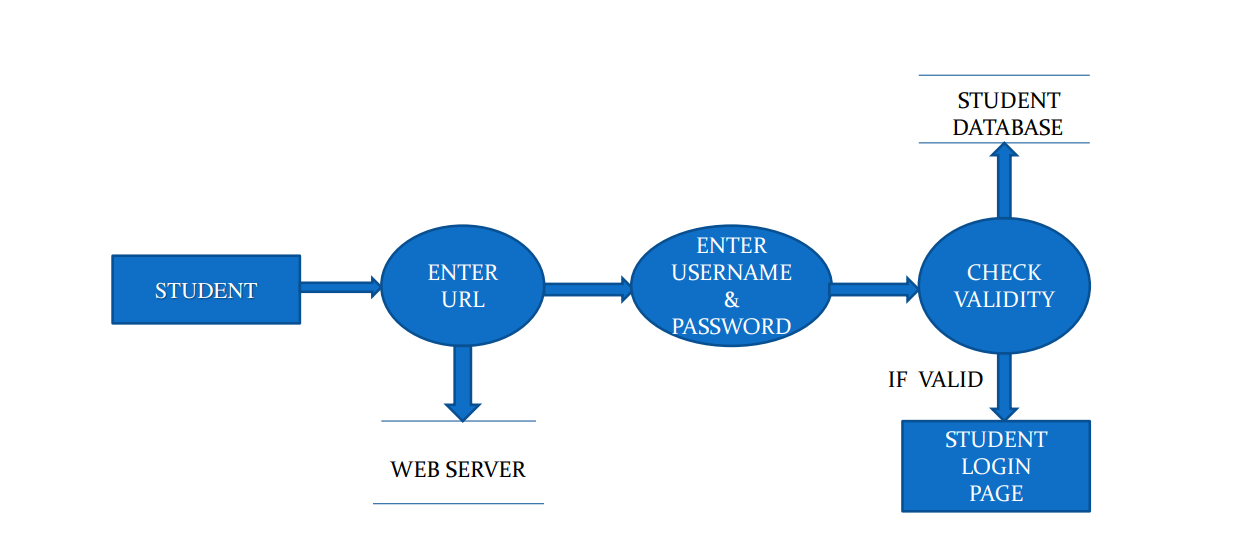
Appendix A

Entity Relationship Diagram



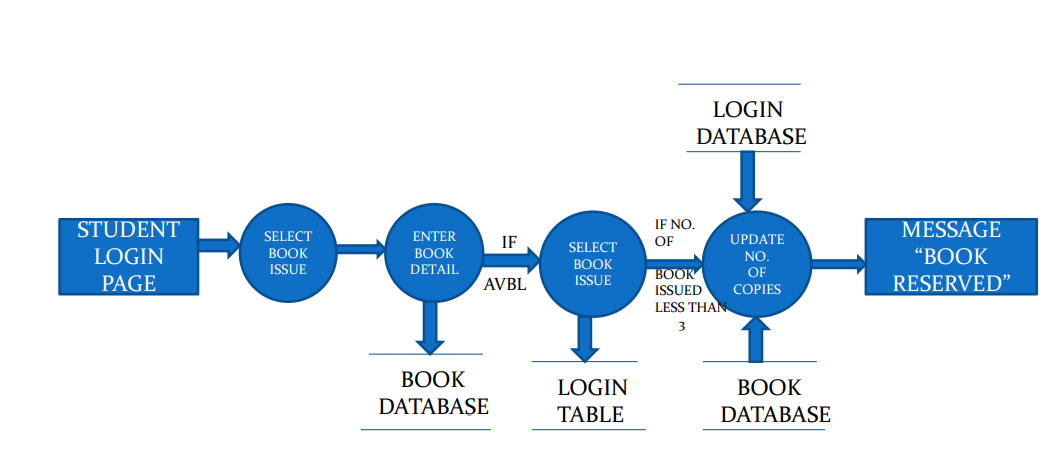
**Data Flow Diagram:**

*Data flow For Student:*

**

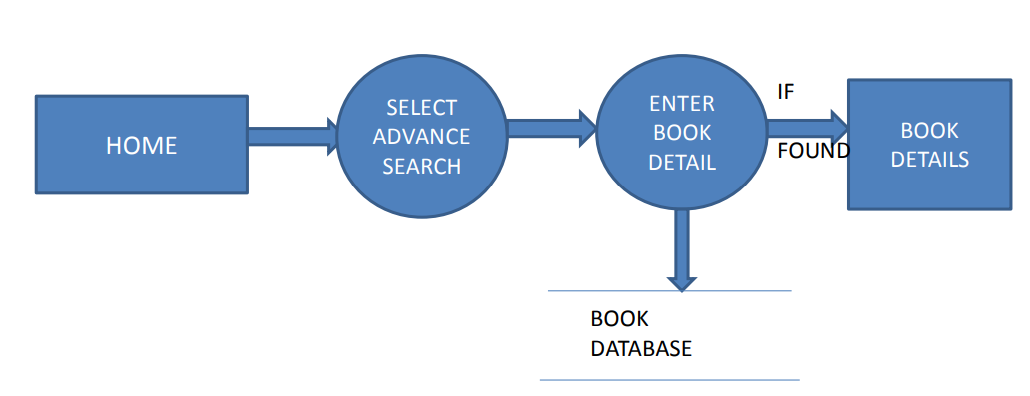
After entering to the home page of the website , student can choose the STUDENT LOGIN option where they are asked to enter username & password , and if he/she is a valid user then a student login page will be displayed.

*For Book:*

****

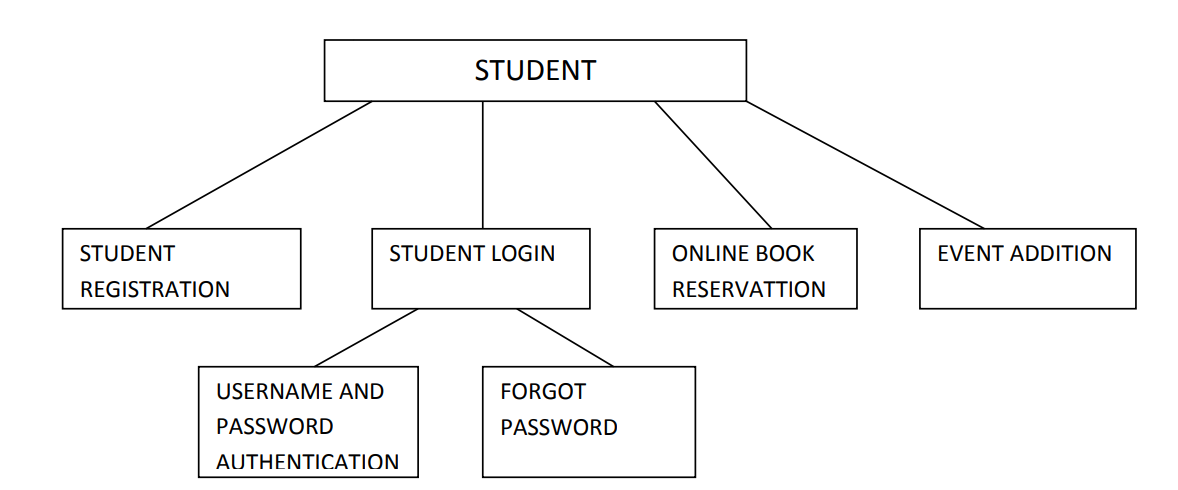
It is a 2nd level Data Flow Diagram where after entering STUDENT LOGIN page he/she can select a book issue option where after entering the book detail, he/she can select the book issue option and if the maximum no of books issued limit is not crossed then a request will be sent to the librarian who will approve the book issue.

*For bookSearch:*

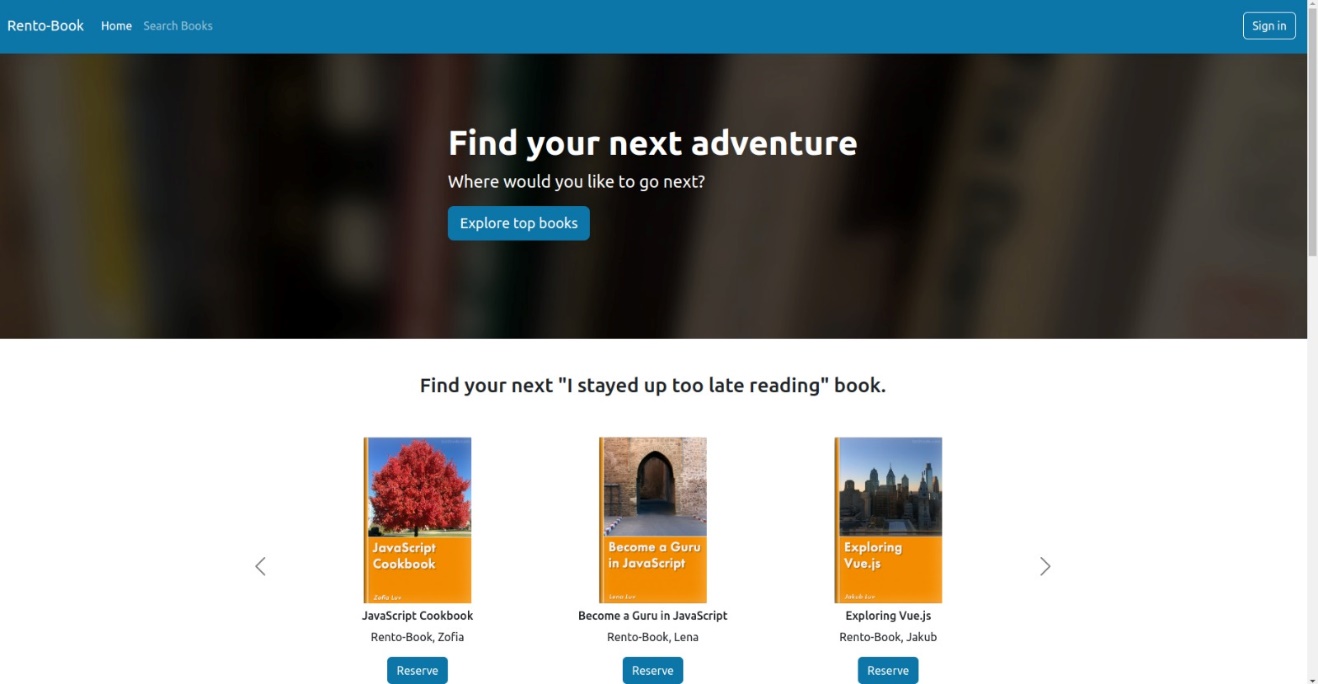
******

After the home page login there will be an option of the book search where after entering book detail like author name, publication, book name etc book details will be displayed.

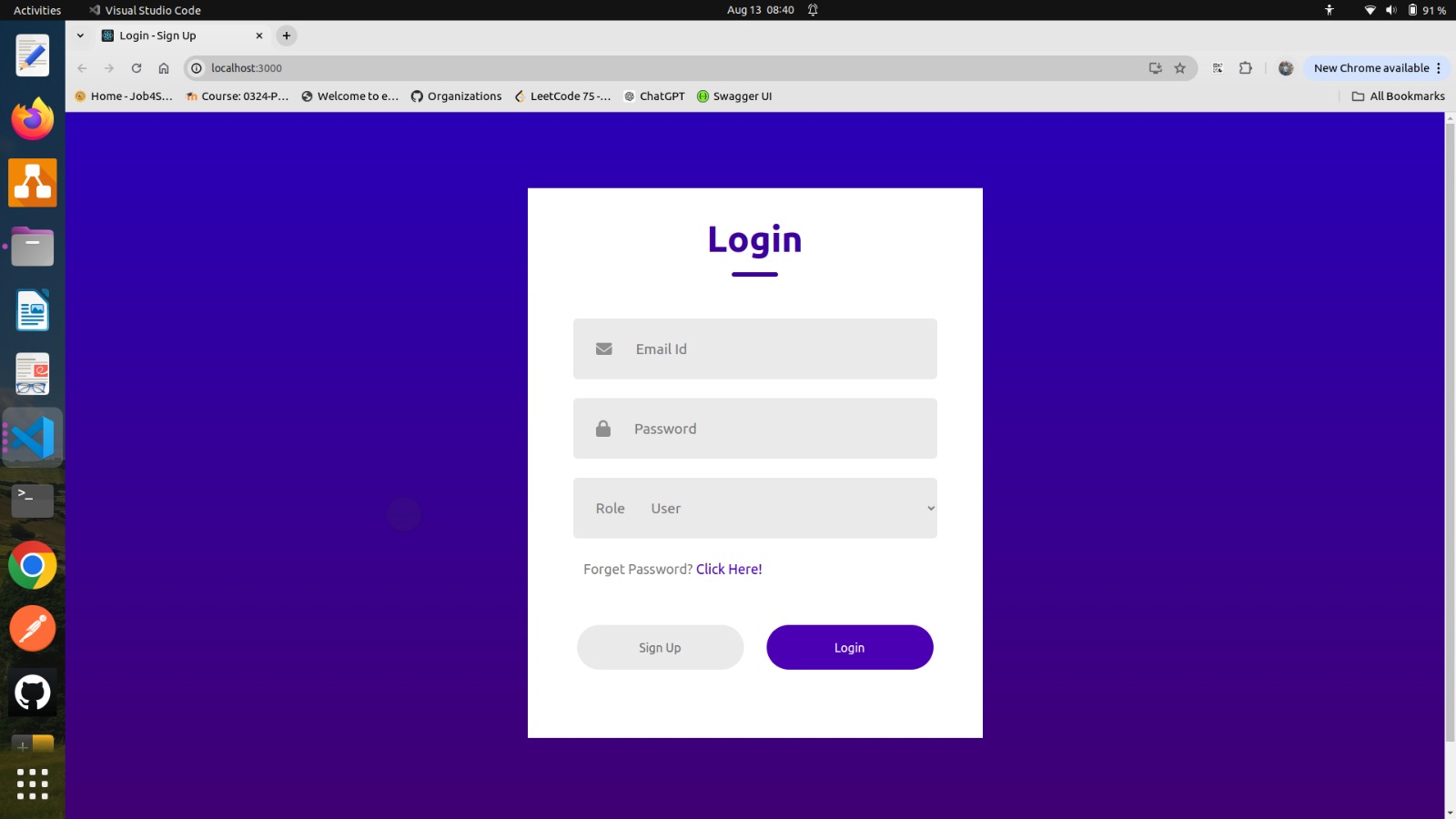
**Class Diagram**



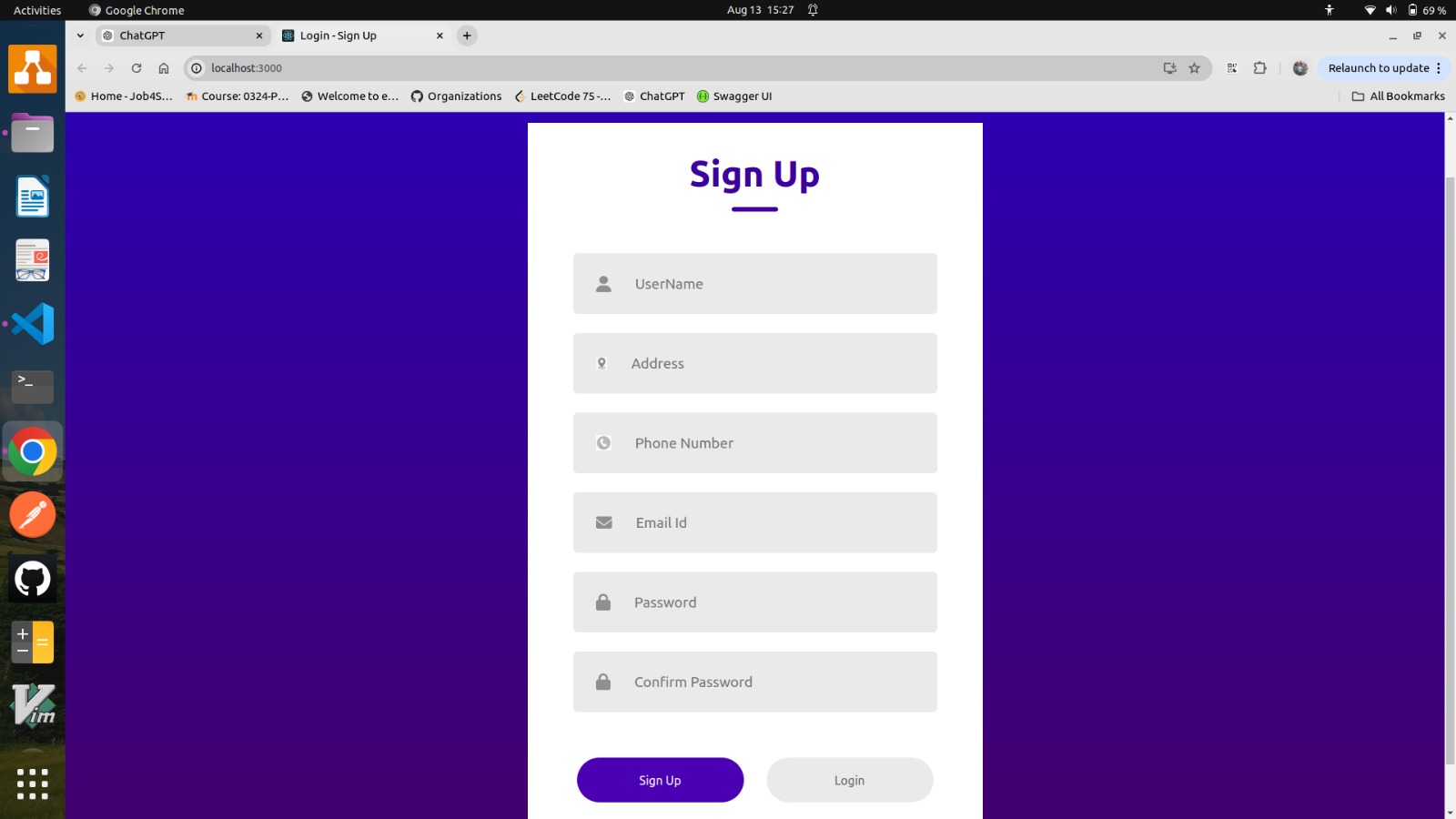
Appendix B

**HomePage:**

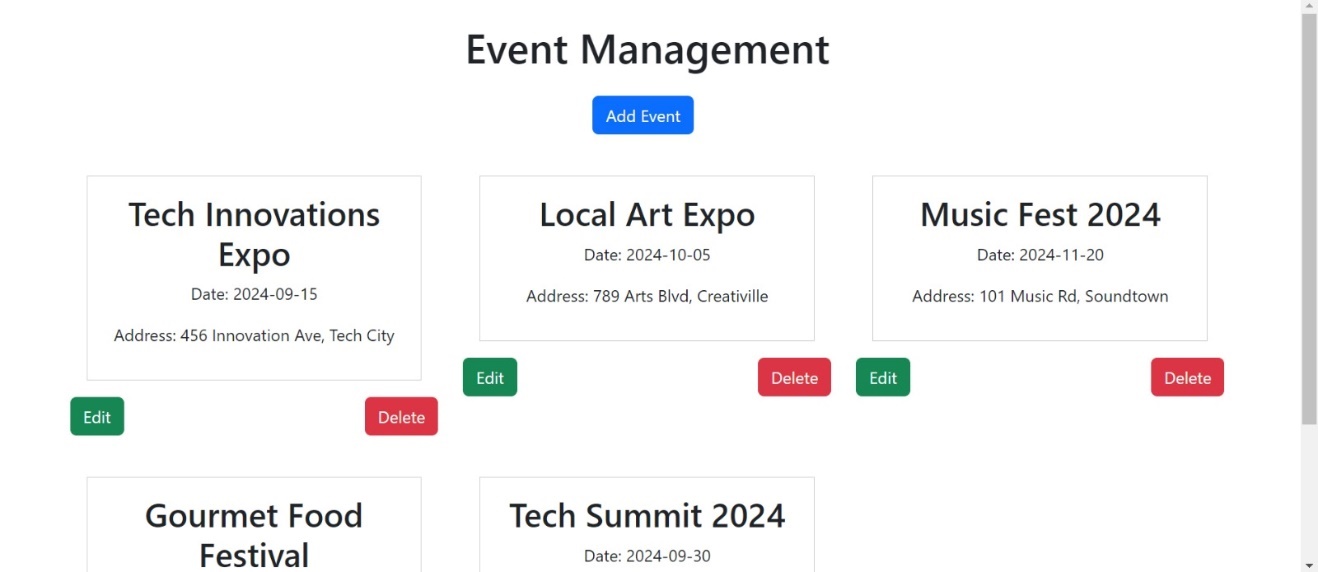
LoginPopup:



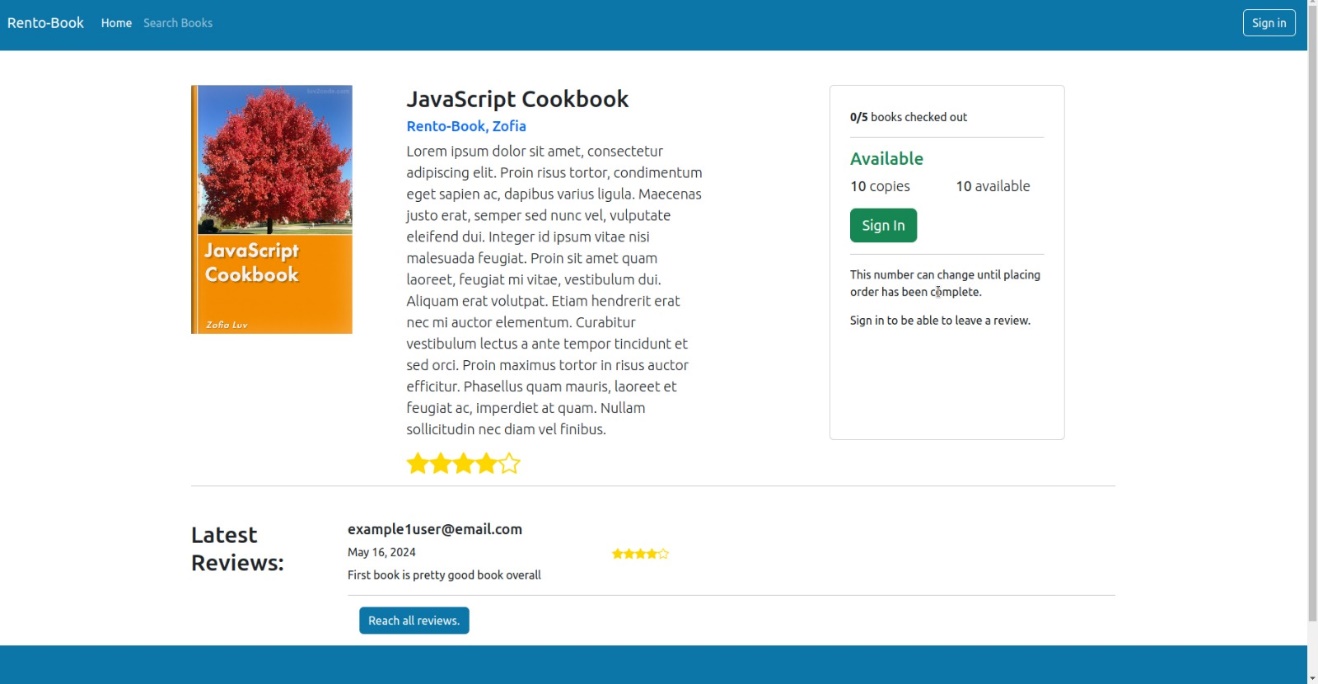
User/ Administrators SignUp:



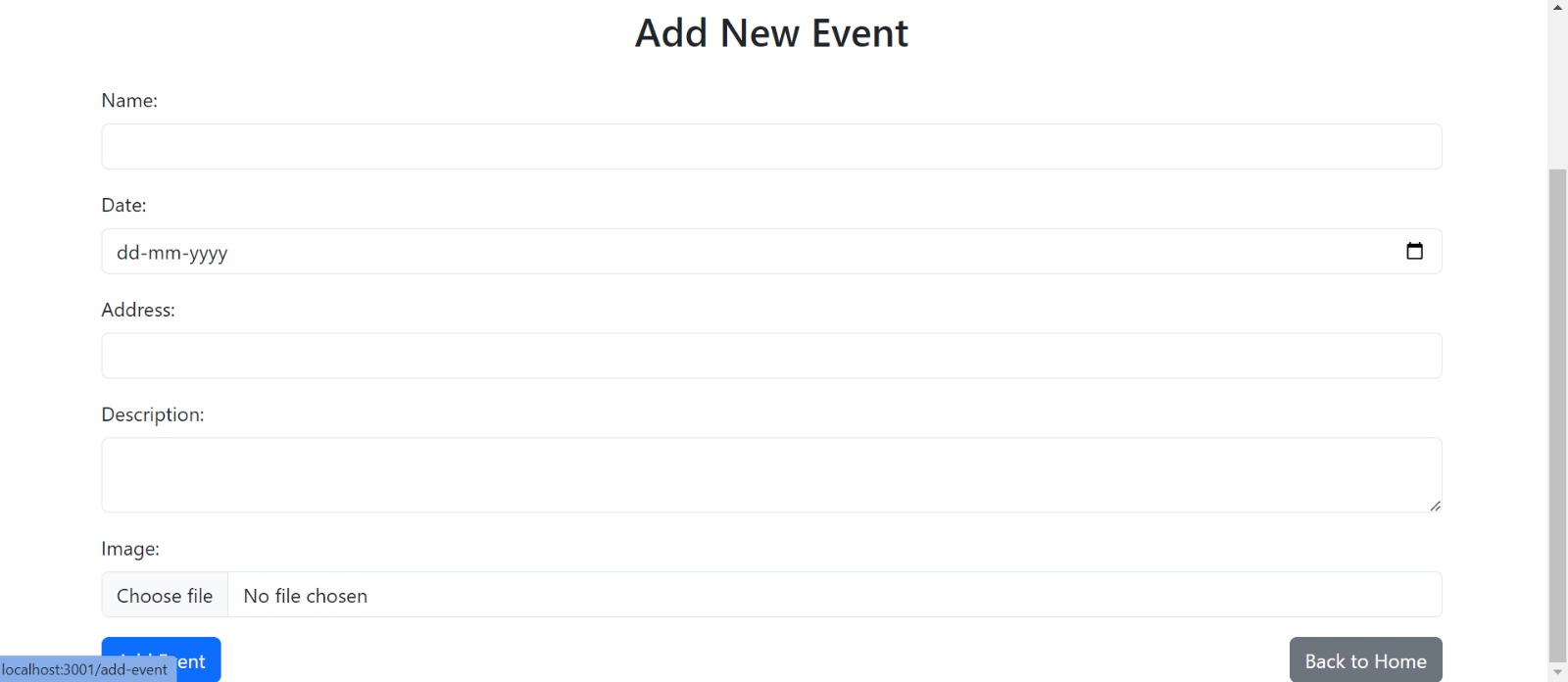
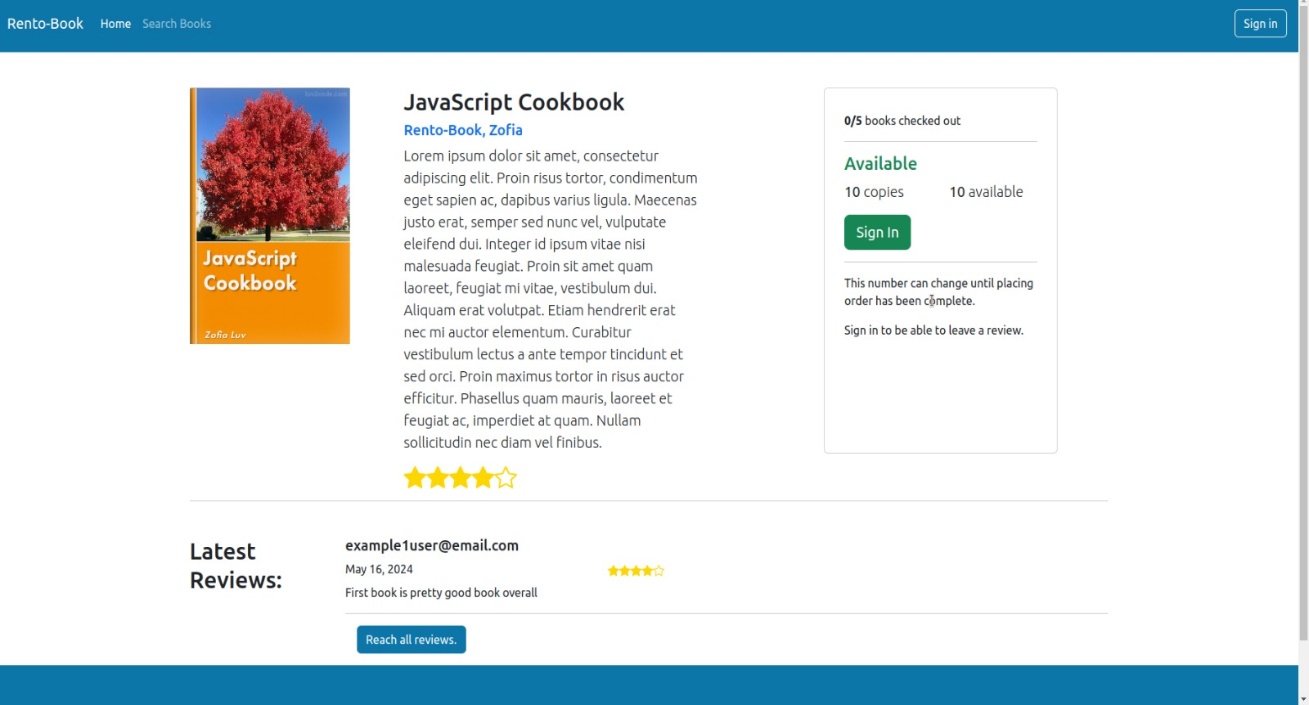
**Add Event:**



**View Details:**

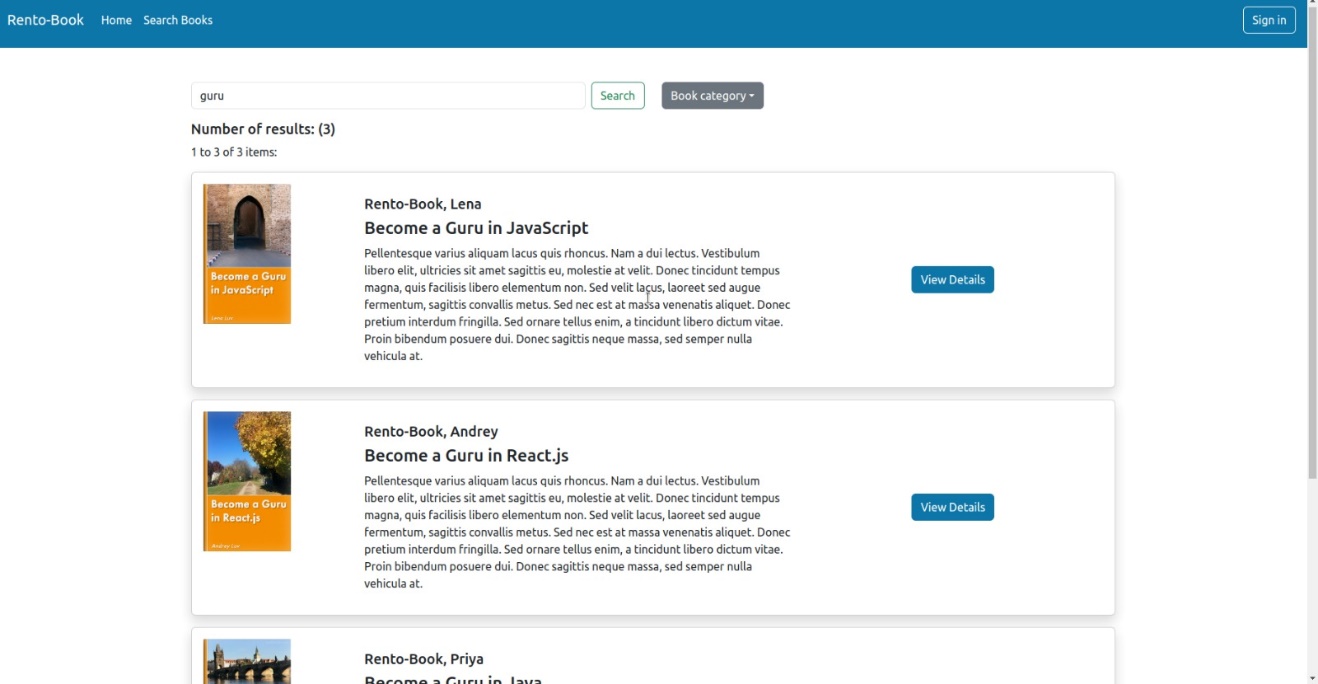
****

View Book:

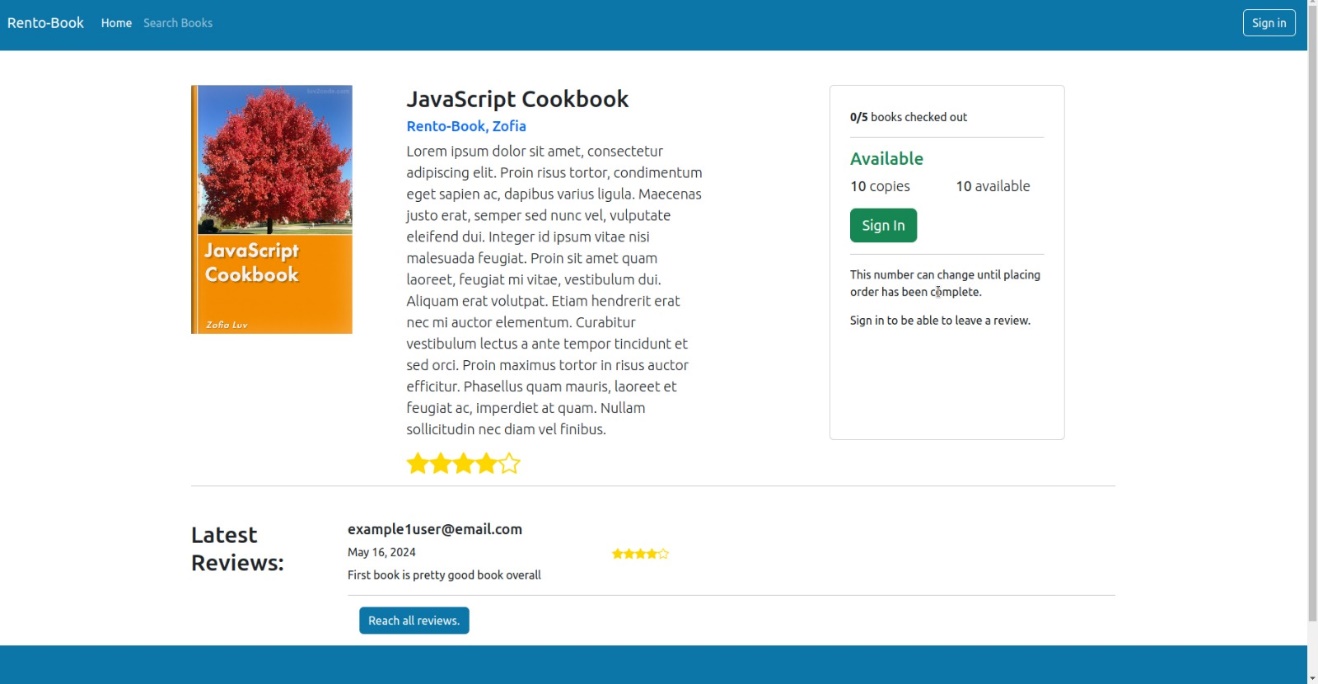


**Add New Event:**

**Search Book:**



**Ratings:**



**7.REFERENCES:**

<https://docs.spring.io/spring-boot/index.html>

<https://legacy.reactjs.org/docs/getting-started.html>

<https://docs.oracle.com/javase/8/docs/api/>

<https://en.wikipedia.org/wiki/Library_management>

<https://github.com/topics/library-management-system>

<https://www.geeksforgeeks.org/library-management-system/>

<https://www.cambridge.org/core/books/abs/librarianship/library-management-systems/D6462625F4FB3171037F1C69EBDBE4EB>

<https://www.javatpoint.com/library-management-system-in-java-swing>