*UE21CS352B - Object Oriented Analysis & Design using Java*

## Mini Project Report

## "Library Management system"

*Submitted by:*

| | |
|---|---|
| **K P Sandeep Ram** | **PES1UG21CS286** |
| **K Prajwal Rai** | **PES1UG21CS254** |
| **Kriti Sujeeth** | **PES1UG21CS290** |
| **Jyotiraditya J** | **PES1UG21CS252** |

*6th Semester E  Section*

**Prof. Bhargavi Mokaskhi**
Designation

**January - May 2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
FACULTY OF ENGINEERING
**PES UNIVERSITY**
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India
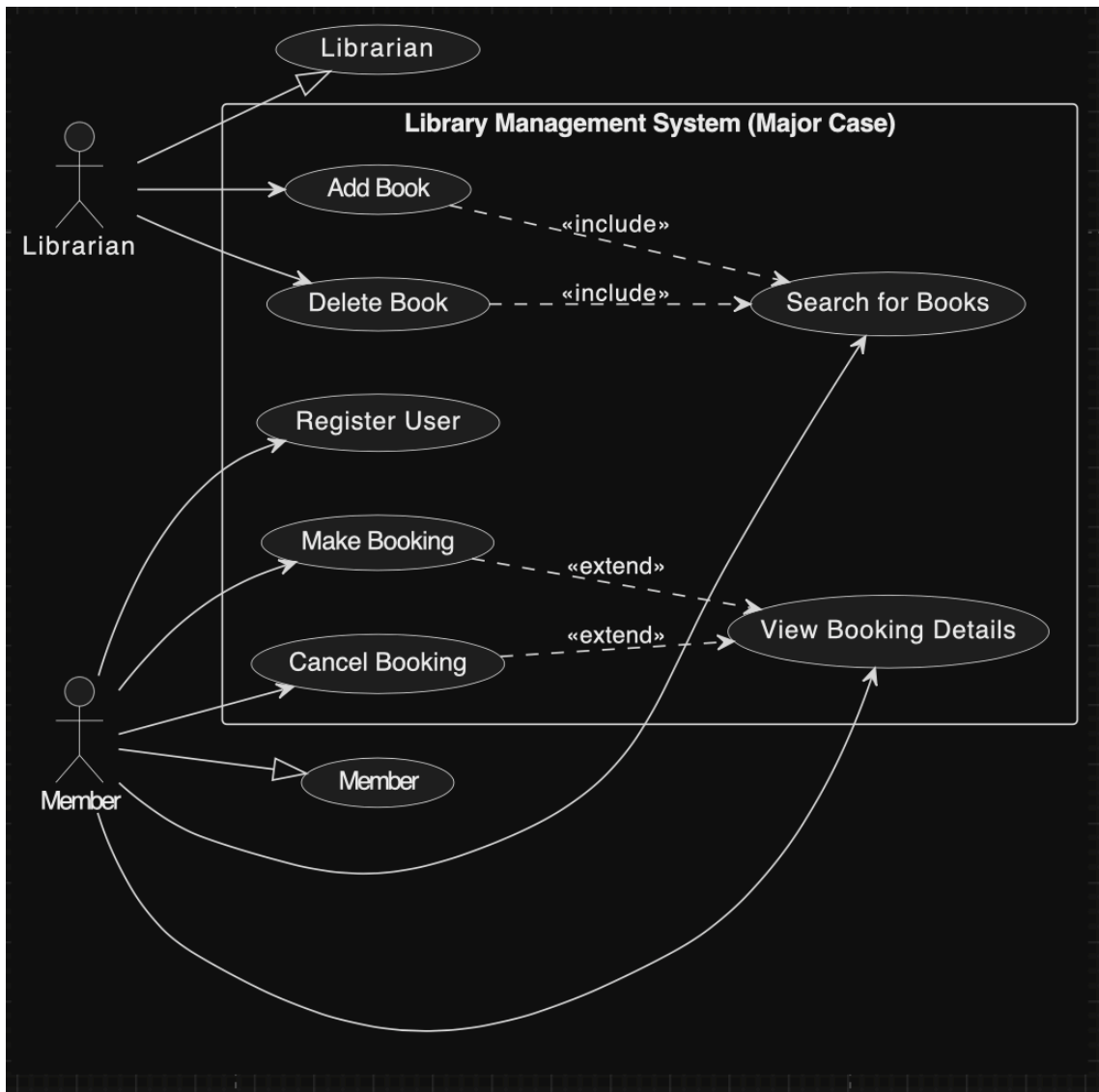
# 1. Problem Statement (synopsis)

This library management system aims to streamline the operations of a library by providing an efficient and user-friendly platform for librarians, members, and non-members (guests) to access and manage library resources.
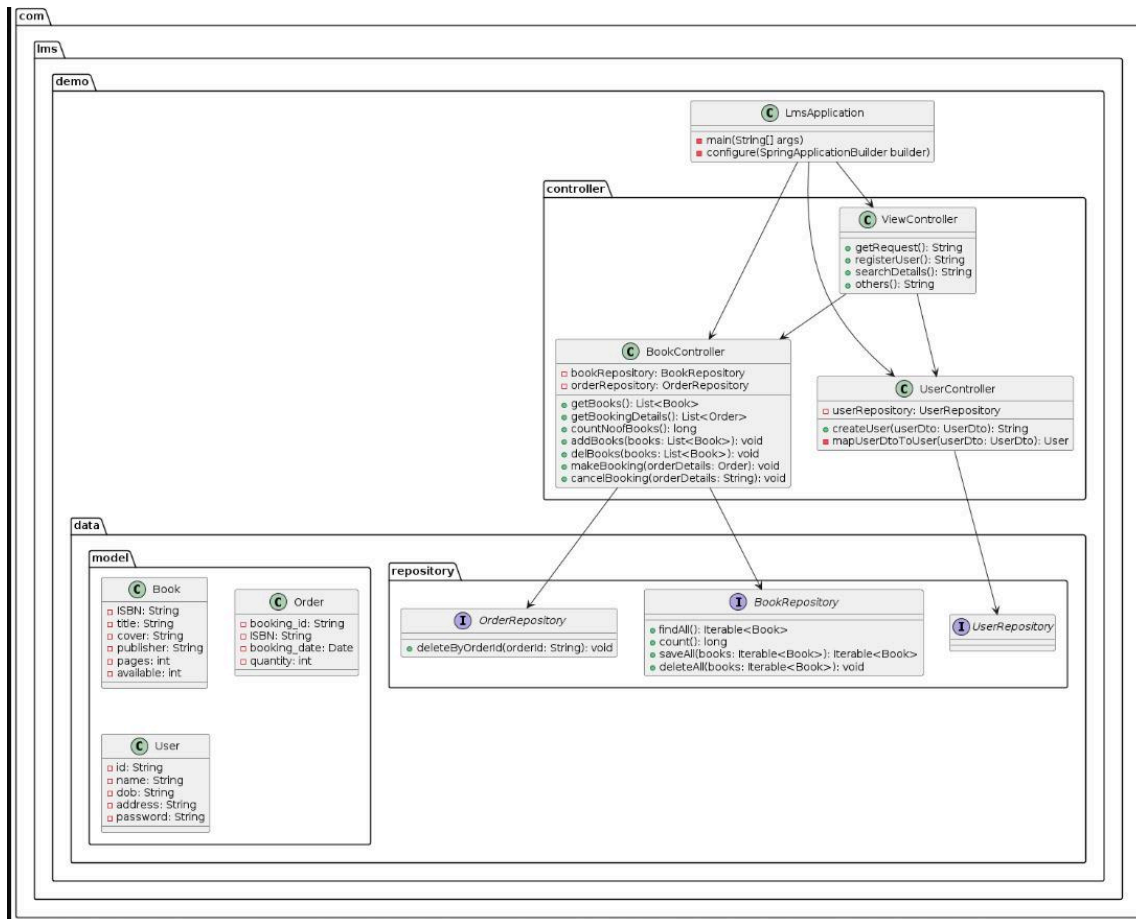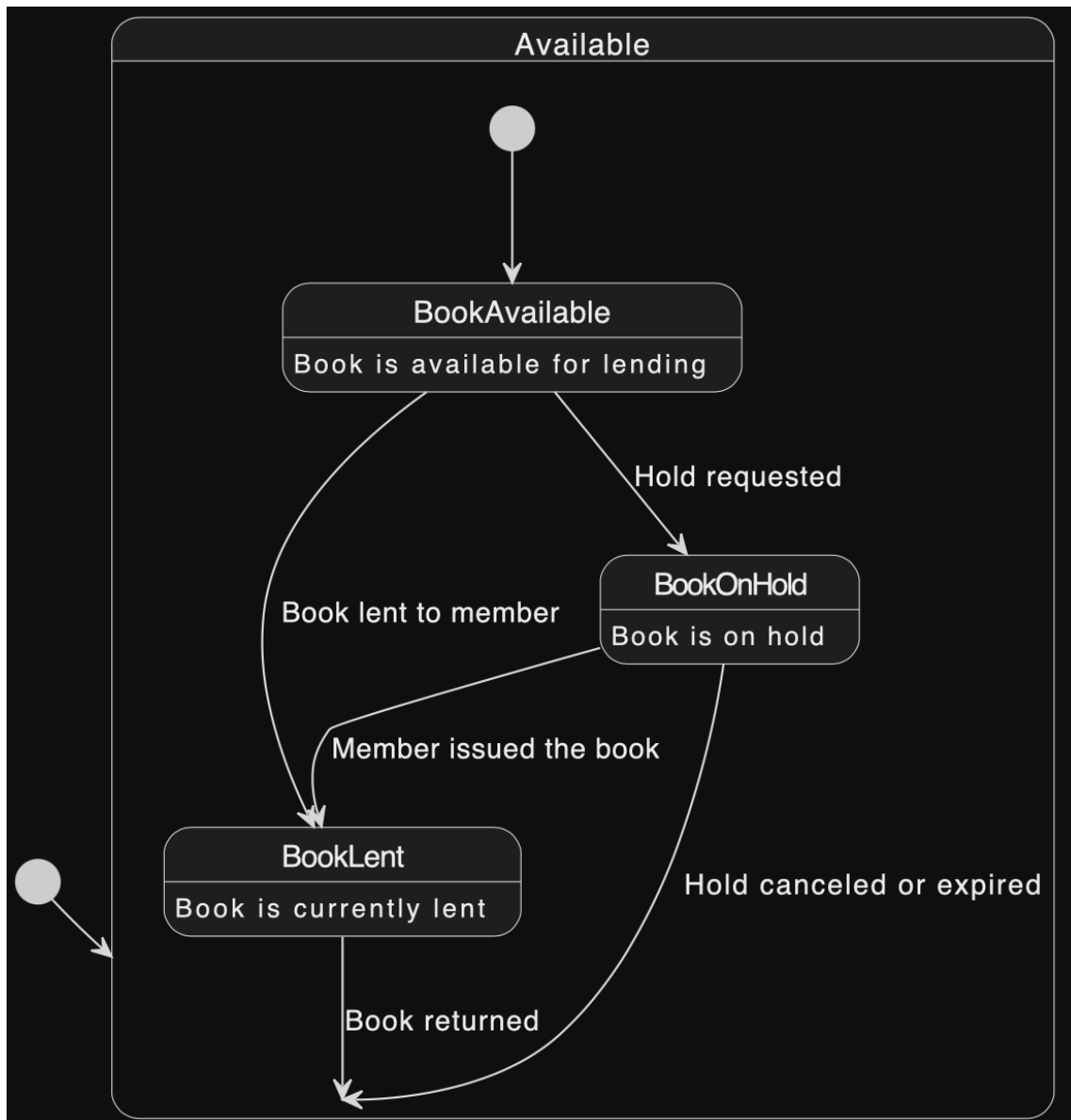
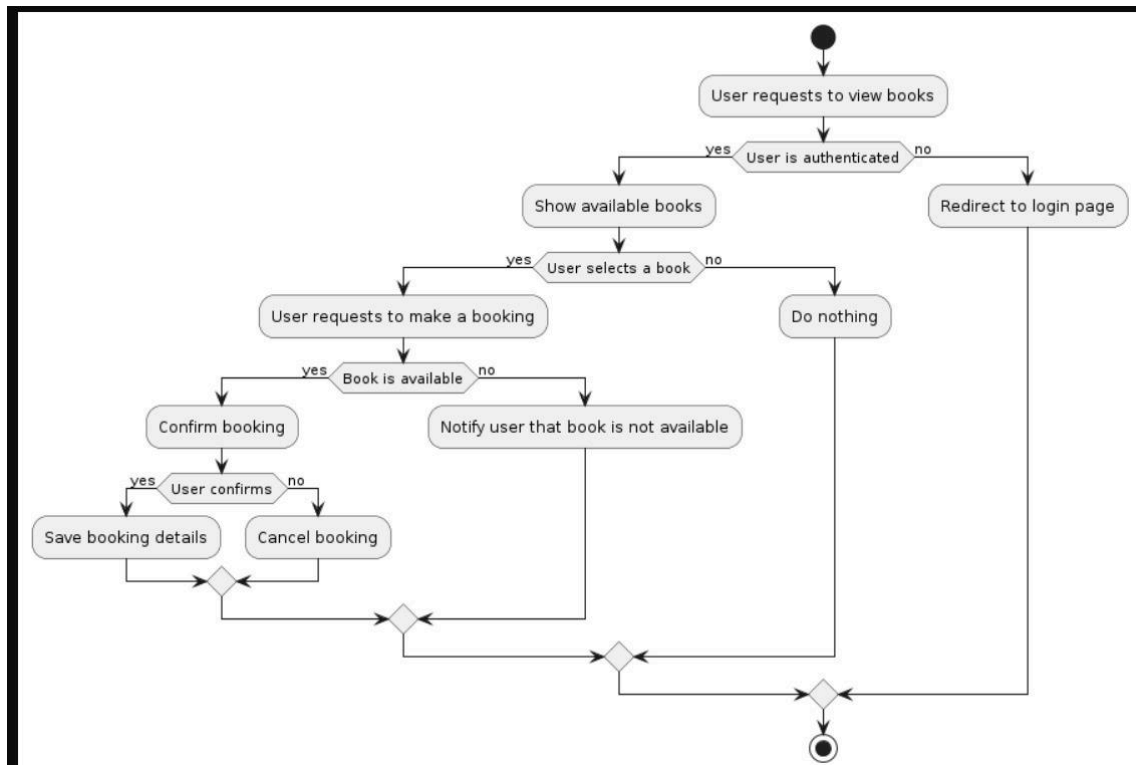# 2. Models

**Minor - Use case**

**Major- Use case**
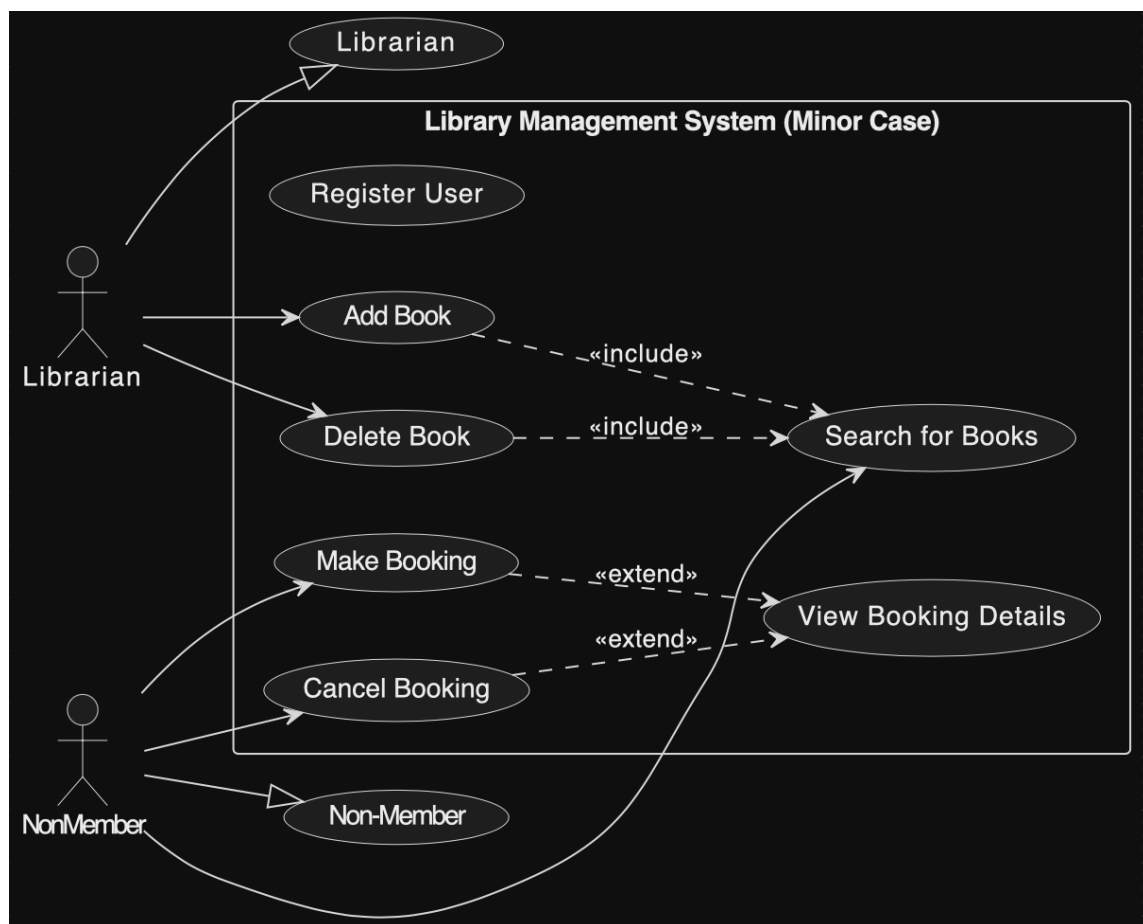


Library Management System (Major Case)

## Class Models

**State Diagram**

**Activity Diagram**

# 3. Architecture and design patterns used

**1. Single Responsibility Principle (SRP):** The Single Responsibility Principle (SRP) states that a class should have only one reason to change, meaning it should have only one job or responsibility. This principle helps in keeping classes focused and manageable, reducing complexity and making the codebase easier to understand, maintain, and extend.

 Each class in our codebase has a specific task:

- Controllers (e.g., `BookController`, `UserController`, `ViewController`) handle HTTP requests and views.
- Model classes (`Book`, `Order`, `User`) manage data and behavior for specific entities.
- Repository interfaces (`BookRepository`, `OrderRepository`, `UserRepository`) deal with data access operations.
- The `UserDto` class acts as a data transfer object between layers.
- We've organized our code to ensure each class focuses on its designated role, keeping things clear and manageable

**2. Dependency Injection (DI):** Dependency Injection (DI) is a design pattern used to remove hard-coded dependencies between classes by injecting dependencies from external sources. This promotes loose coupling and facilitates easier testing, scalability, and flexibility in the application.

- In the `UserController` class, we inject the `UserRepository` dependency via constructor injection.
- This allows us to easily swap dependencies or mock them for testing purposes, enhancing the flexibility and testability of our code.

    .

**3. Open/Closed Principle (OCP):** The Open/Closed Principle (OCP) states that software entities (classes, modules, functions, etc.) should be open for extension but closed for

modification. This means that existing code should be easily extensible without requiring changes to its source code.

Classes are structured to allow for extending functionality without modifying existing code.

This promotes modularity and reusability, as new features can be added without risking unintended side effects in existing code.

**4. Model-View-Controller (MVC) Pattern:** The Model-View-Controller (MVC) pattern is a widely used architectural pattern for designing user interfaces. It divides an application into three interconnected components: Model (data and business logic), View (presentation layer), and Controller (handles user input and updates the model and view).

Models (`Book`, `Order`, `User`) handle data and business logic.

Views are managed by the `ViewController` class, handling view-related logic.

Controllers (`BookController`, `UserController`) manage HTTP requests and interact with models and views.

**5. Repository Pattern:** The Repository pattern is a design pattern that separates the logic that retrieves data from a data source (such as a database) from the business logic in the application. It provides a layer of abstraction for data access operations, making it easier to switch between different data sources or databases.

Repository interfaces (`BookRepository`, `OrderRepository`, `UserRepository`) abstract database operations.

They define CRUD operations for entities, decoupling application logic from data storage mechanisms.

Spring Data JPA simplifies repository implementation, reducing boilerplate and enhancing maintainability.

**6. Data Access Object (DAO) Pattern:** The Data Access Object (DAO) pattern is a design pattern that separates the data access code from the business logic in an application. It provides an abstract interface to access data stored in a database or other data sources.

Model classes (`Book`, `Order`, `User`) represent data entities.

Repository interfaces (`BookRepository`, `OrderRepository`, `UserRepository`) encapsulate data access logic.

This separation ensures clear responsibilities and promotes maintainability and testability.

7. **Data Transfer Object (DTO) Pattern:** The Data Transfer Object (DTO) pattern is a design pattern used to transfer data between software application subsystems or layers. It typically consists of simple data structures with no business logic.

We employ the DTO pattern for efficient data transfer between layers:

- The `UserDto` class carries user-related data between different parts of the application.
- DTOs like this simplify communication and reduce the amount of data transferred.
- Using Lombok's `@Builder` annotation streamlines DTO creation, improving readability and reducing boilerplate code.

# 4. Github Repository Link:

https://github.com/SandeepRam31/OOAD
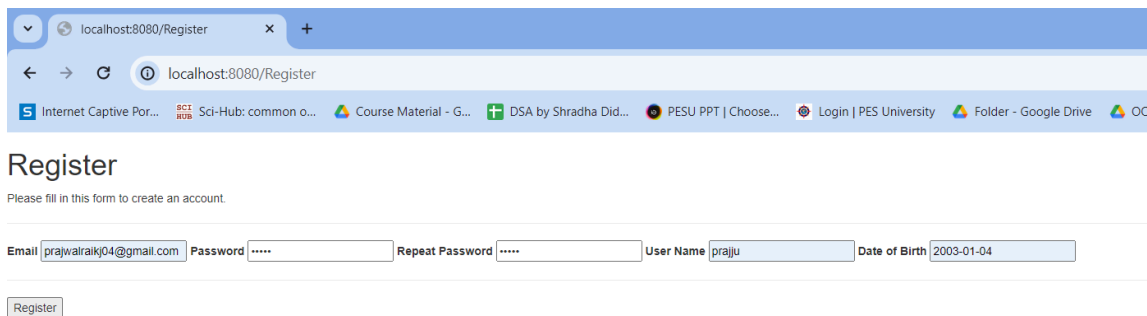
# 5. Individual Contributions:

● **Prajwal Rai**: Designed the overall design patterns and principles to be used in the project, and developed the register, login, and search pages for the application. Ensuring connectivity to the backend and implementing 'Search all book details' and 'Number of books' functionalities.

● **Sandeep Ram**: Implemented Spring book frame work and set up the apis to link the front end to the back end. Implemented the logic for Add New Book, Bookings and Cancel Booking.

● **Kriti Sujeeth**: Implemented the front end design as well as database schema. Created the use case (minor and major) and State diagrams relevant for this project.

● **Jyotiraditya**: Created the Class and activity diagrams, developed the makebooking and delete booking functionalities along with database.

# 6. <u>Screenshots with input values:</u>

● **Register page:**
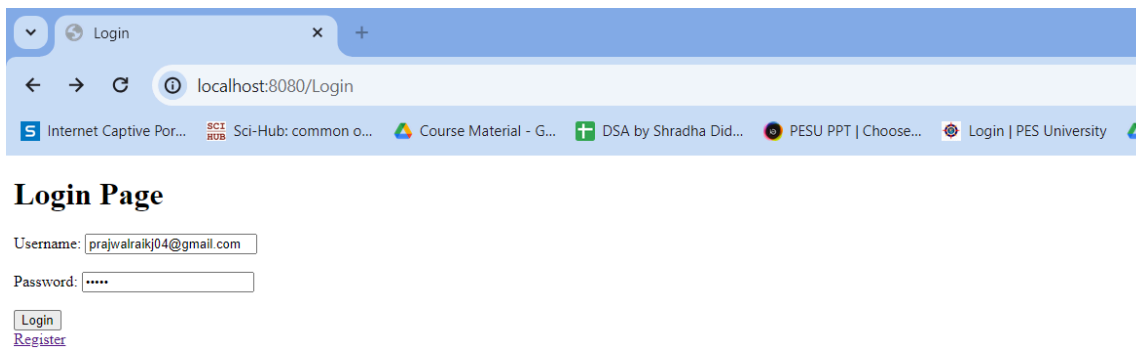


Output: User info entered in the database

- **Login page:**



- **Home page:**

- **Search Book Function:** The user can search for the book of his/her liking from the collection of books available:



- **Add Book function:** The Librarian can add a book to the database.

- **Delete Book function:** The librarian can delete a book from the database using this function:

**Before deleting:**

**After deleting:** Select the book to removed



Click on Delete Books button

```
mysql> select * from book where isbn="664";
Empty set (0.00 sec)
```

- **Booking function:** Allows the user to order a book.



```
mysql> select * from order_details;
+----------------------------------+------+--------------+----------+
| booking_id                       | ISBN | booking_date | quantity |
+----------------------------------+------+--------------+----------+
| 402880388f0b7c0d018f0b80019b0000 | 666  | 2024-04-23   | 666      |
| 402880388f1e15ce018f1e1bffba0000 | 666  | 2024-04-27   | 666      |
| ff80808162bf9a0d0162bf9dfc8a0008 | 663  | 2018-04-13   | 663      |
+----------------------------------+------+--------------+----------+
```

Clicking on the "Buy" button confirms the booking

- **Cancel Booking:** The user can cancel the book that he had previously ordered using this functionality.

# Welcome to Online Library

Pick Your Search Criteria

Search all Book Details ○   Add New Book ○   Delete Existing Book ○   Booking ○   Cancel Booking ●

Search

Cancel Your Booking

| Booking Code:663 | Booking Code:666 | Booking Code:666 | Booking Code:663 |
|---|---|---|---|
| Booking date is:2024-04-22 | Booking date is:2024-04-23 | Booking date is:2024-04-27 | Booking date is:2018-04-13 |
| Refund Amount:663 | Refund Amount:666 | Refund Amount:666 | Refund Amount:663 |
| Cancel Order | Cancel Order | Cancel Order | Cancel Order |

First   Previous   1   Next   Last

```
mysql> select * from order_details;
+--------------------------------+------+--------------+----------+
| booking_id                     | ISBN | booking_date | quantity |
+--------------------------------+------+--------------+----------+
| 402880388f0b7c0d018f0b80019b0000 | 666  | 2024-04-23   | 666      |
| 402880388f1e15ce018f1e1bffba0000 | 666  | 2024-04-27   | 666      |
| 402880388f1e15ce018f1e1fdab10001 | 672  | 2024-04-27   | 672      |
| ff80808162bf9a0d0162bf9dfc8a0008 | 663  | 2018-04-13   | 663      |
+--------------------------------+------+--------------+----------+
4 rows in set (0.00 sec)
```

On clicking the cancel option, the order is cancelled.

Search Page     ×     +

← → C  ⓘ  localhost:8080/search

S Internet Captive Por...   SCI Sci-Hub: common o...   ▲ Course Material - G...   🟦 DSA by Shradha Did...   ⬤ PESU PPT | Choose...   ⬤ Login | PES University   ▲ Folder - Google Drive

# Welcome to Online Library

**Pick Your Search Criteria**

Search all Book Details ○     Add New Book ○     Delete Existing Book ○     Booking ○     Cancel Booking ⦿
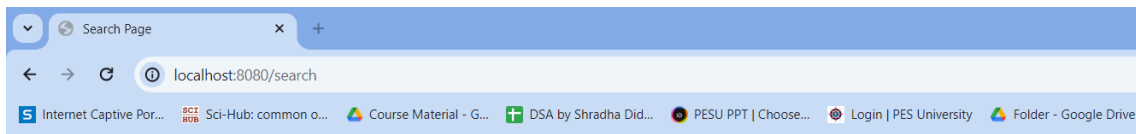
Search

Cancel Your Booking

| **Booking Code:666** | **Booking Code:666** | **Booking Code:663** |
| **Booking date is:2024-04-23** | **Booking date is:2024-04-27** | **Booking date is:2018-04-13** |
| **Refund Amount:666** | **Refund Amount:666** | **Refund Amount:663** |
| Cancel Order | Cancel Order | Cancel Order |

First  Previous  1  Next  Last

Cancelled Sucessfully

```
mysql> select * from order_details;
+----------------------------------+------+--------------+----------+
| booking_id                       | ISBN | booking_date | quantity |
+----------------------------------+------+--------------+----------+
| 402880388f0b7c0d018f0b80019b0000 | 666  | 2024-04-23   | 666      |
| 402880388f1e15ce018f1e1bffba0000 | 666  | 2024-04-27   | 666      |
| ff80808162bf9a0d0162bf9dfc8a0008 | 663  | 2018-04-13   | 663      |
+----------------------------------+------+--------------+----------+
```