**ONLINE BOOK STORE SYSTEM**

#### CS2333 – Object Oriented Programming Using JAVA Project Report

*Submitted by*

**GOWTHAM PL-231001049**

**ALFRED DHANAM P-231001011**

*Of*

**BACHELOR OF TECHNOLOGY**

*In*

**INFORMATION TECHNOLOGY**

#### RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM

(An Autonomous Institution)



**RAJALAKSHMI ENGINEERING COLLEGE**

**NOVEMBER-2024**

**BONAFIDE CERTIFICATE**

|  |
| --- |
| Certified that this project titled “Online Book Store System” is the bonafide work of **GOWTHAM PL(231001049),ALFRED DHANAM P(231001011)** who carried out the project work under my supervision. |

**SIGNATURE**  **SIGNATURE**

**Dr.P.Valarmathie Mrs.Usha S**

**HEAD OF THE DEPARTMENT** **COURSE INCHARGE**

**Assistant Professor(S.G)**

Information Technology Information Technology

Rajalakshmi Engineering College Rajalakshmi Engineering College

This project is submitted for CS23333 – Object Oriented Programming Using JAVA held on \_\_\_\_\_\_\_\_\_

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| 1 | **1.1 ABSTRACT** | **5** |
|  | 1.2 INTRODUCTION | **5** |
|  | 1.3 PURPOSE | **5** |
|  | 1.4 SCOPE OF PROJECT | **6** |
|  | 1.5 SOFTWARE REQUIREMENT SPECIFICATION | **6** |
| **2** | **SYSTEM FLOW DIAGRAM** | **12** |
|  | 2.1 USE CASE DIAGRAM | **12** |
|  | 2.ENTITY RELATIONSHIP DIAGRAM | **13** |
|  | 2.3 DATA FLOW DIAGRAM | **14** |

|  |  |  |
| --- | --- | --- |
| **3** | **MODULE DESCRIPTION** | **15** |
| **4** | **IMPLEMENTATION** |  |
|  | 4.1 DESIGN | **16** |
|  | 4.2 DATABASE DESIGN | **19** |
|  | 4.3 CODE | **21** |
| **5** | **CONCLUSION** | **26** |
| **6** | **REFERENCE** | **26** |

**1.1Abstract:**

The primary objective of the JDBC-powered Online Book Store System in Java is to provide a reliable and efficient platform for users to seamlessly browse, select books, and buy books, all while leveraging the power of Java's database connectivity capabilities.

## 1.2 Introduction:

The Online Bookstore System is a modern software application aimed at transforming the traditional methods of purchasing and managing books. This advanced system introduces automation to improve efficiency, minimizing manual effort and operational challenges. Secured by robust authentication, it ensures that only authorized users can manage essential functions like book inventory, user accounts, and order details. By leveraging Java and MySQL, the system offers a seamless and user-friendly interface for book browsing and purchasing. With its focus on reliability and scalability, this system provides an innovative and efficient alternative to conventional bookstore practices, meeting the demands of today’s digital age.

1.3 Purpose:

The purpose of the Online Bookstore System is to create an efficient and user-friendly platform for browsing and purchasing books online. The system aims to streamline the process of book selection, purchase, and order management while providing a secure and seamless user experience. By utilizing Java, MySQL, and JDBC, the project ensures smooth integration between the frontend and backend, making it an effective solution for modern e-commerce in the book retail industry. By integrating modern technologies, the Online Bookstore System provides a reliable solution for the growing demands of online book retail while laying the groundwork for future scalability and feature enhancements.

## 1.4 Scope of the Project:

The scope of the Online Bookstore System includes the development of a fully functional e-commerce platform for purchasing books online. The system will provide users with features such as browsing a catalog of books, searching for specific titles, viewing detailed book information, and making secure purchases. The backend, powered by MySQL, will handle user authentication, book inventory management, and order processing. JDBC will be used to ensure seamless communication between the Java frontend and the MySQL database.

## 1.5 Software Requirement Specification:

Introduction:

The Online Bookstore System is a platform that allows users to browse and purchase books online. It uses Java for the frontend, MySQL for the backend, and JDBC for database connectivity, ensuring a smooth and secure user experience.

Document Purpose:

This Software Requirements Specification (SRS) document provides a detailed outline of the Online book store System’s requirements. It includes design considerations, architectural structure, and technical specifications, ensuring smooth implementation and continued software maintenance.

Product Scope:

The product scope of the Online Bookstore System includes a platform where users can browse and search for books, view detailed information, and make secure purchases. It features user account management, order processing, and a MySQL database backend for storing book and user data.

MRS - Movie Reservation System

SRS - Software Requirements Specification.

References and Acknowledgement:

* <https://www.javatpoint.com/java-awt>
* <https://www.javatpoint.com/java-swing>
* <https://www.w3schools.com/sql/>
* https://www.geeksforgeeks.org/introduction-to-jdbc/

The Online book store system enables authorized personnel to efficiently manage book records, offering a streamlined reservation process for book store globally. This system simplifies daily operations across various educational venues.

Product Perspective:

The Online Bookstore System is designed as a standalone web application that integrates a frontend built with Java and a backend powered by MySQL. It is intended to serve as an e-commerce solution for book retailers, providing a user-friendly interface for customers to browse and purchase books. The system leverages JDBC to enable smooth communication between the frontend and the database. It fits within the broader context of online retail solutions, offering a secure, efficient, and scalable platform for the sale of books.

Product Functionality:

* Admin Register: Allows new administrators to register on the platform.
* Admin Login: Provides secure login functionality for existing administrators.
* Add Book: Enables administrators to add details of new books.
* View book: Facilitates viewing and updating of existing book information.
* Delete book: Permits removal of books from the system database.
* Add to cart: Supports the creation of new cart for users.
* Update orders: Allows users to view and modify their existing orders.
* Remove Admin: Enables deletion of specific administrator accounts.

User and Characteristics:

Qualification: Users should have at least basic educational qualifications, such as matriculation, and be comfortable with English.

Experience: Familiarity with the university registration process is advantageous. Technical Experience: Users are expected to have elementary knowledge of computers for optimal system interaction.

Operating Environment:

Hardware Requirements:

* Processor: Any Processor over i3
* Operating System: Windows 8, 10, 11
* Processor Speed: 2.0 GHz
* RAM: 4GB
* Hard Disk: 500GB

Software Requirements:

* Database: MySQL
* Frontend: Java (SWING, AWT)
* Technology: Java (JDBC)

Constraints:

* System access limited to administrators.
* Delete operation restricted to administrators without additional checks for simplicity.
* Administrators must exercise caution during deletion to maintain data consistency.

Assumptions and Dependencies:

* System administrators create and confidentially communicate login IDs and passwords to users. Specific Requirements:

User Interface:

The Movie Reservation System provides user-friendly, simplified interfaces for:

1. Admin Register: Registering new administrators.
2. Admin Login: Logging in existing administrators.
3. View book: Viewing and updating existing book information.
4. Delete book: Deleting existing book.
5. Add to cart: Creating new cart for users.
6. Update orders: Viewing and modifying existing orders.
7. Remove Admin: Deleting specific administrator accounts.

Hardware Interface:

* Screen resolution of at least 640 x 480 or above.
* Compatible with any version of Windows 8, 10, 11.

Software Interface:

1. MS-Windows Operating System
2. Java AWT and SWING for designing the front end
3. MySQL for the backend
4. Platform: Java Language
5. Integrated Development Environment (IDE): Netbeans

Functional Requirements:

1. Log in Module (LM):
   * Users (admins) access the Login Module.
   * LM supports user login with a username and password.
   * Passwords are masked for security.
   * Successful login verification by the database administrator is required for access.

1. Registered Users Module (RUM):
   * After successful login, users (admins) can navigate through the application. - Users can view detailed information about movies, showtimes, and reservations.
   * Users can update and maintain book details, including modifying name and price.

1. Administrator Module (AM)
   * Upon successful login, the system displays administrative functions.
   * Functions include adding and updating book details.
   * The "Add" function allows administrators to input new book details and remove unused entries.
   * The "Update" function enables administrators to modify existing movie details in the database.
   * All add, update, or delete requests trigger the AM module to communicate with the Server Module (SM) for necessary database changes.

1. Server Module (SM):
   * SM acts as an intermediary between various modules and the database (DB).
   * Receives requests from different modules and formats pages for display.
   * Validates and executes requests received from other modules.
   * Handles communication with the database, ensuring data consistency and integrity, especially regarding book details, stock, and price.

Non-functional Requirements:

Performance:

* + The system must handle real-time ordering requests efficiently, ensuring a response time of less than 2 seconds for book selection and confirmation. - Safety-critical failures, such as payment processing errors, must be addressed instantly to ensure a smooth user experience.

Reliability:

* + The system is safety-critical; in case of abnormal operation or downtime, immediate measures should be taken to resolve the issue and restore normal functionality.

Availability:

* + Under normal operating conditions, user requests for book orders, including book selection and payment, should be processed within 2 seconds to maintain a seamless booking experience.
  + Immediate feedback on reservation status and confirmation should be communicated to users to enhance the overall booking process.

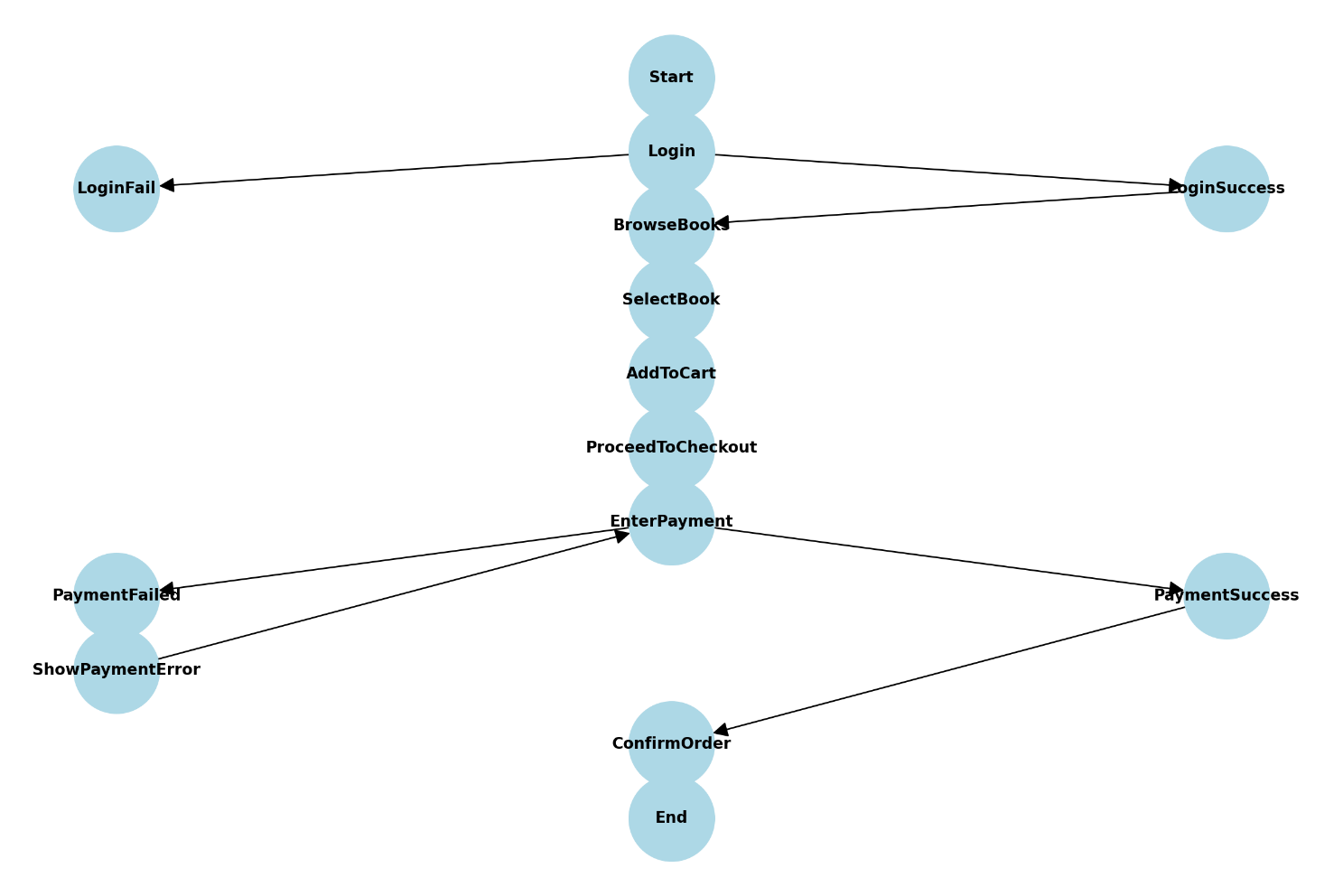
Security:

* + A robust security mechanism must be in place on the server side to prevent unauthorized access, safeguard user payment information, and ensure the integrity of the reservation system.
  + User privacy, including personal details, must be securely stored and managed to maintain confidentiality.

Maintainability:

* + Design documents outlining software and database maintenance procedures must be available to facilitate regular updates and modifications to the movie reservation system.
  + Administrative access should be provided for proper maintenance at both the front end and back end, ensuring the system's long-term functionality and adaptability. end

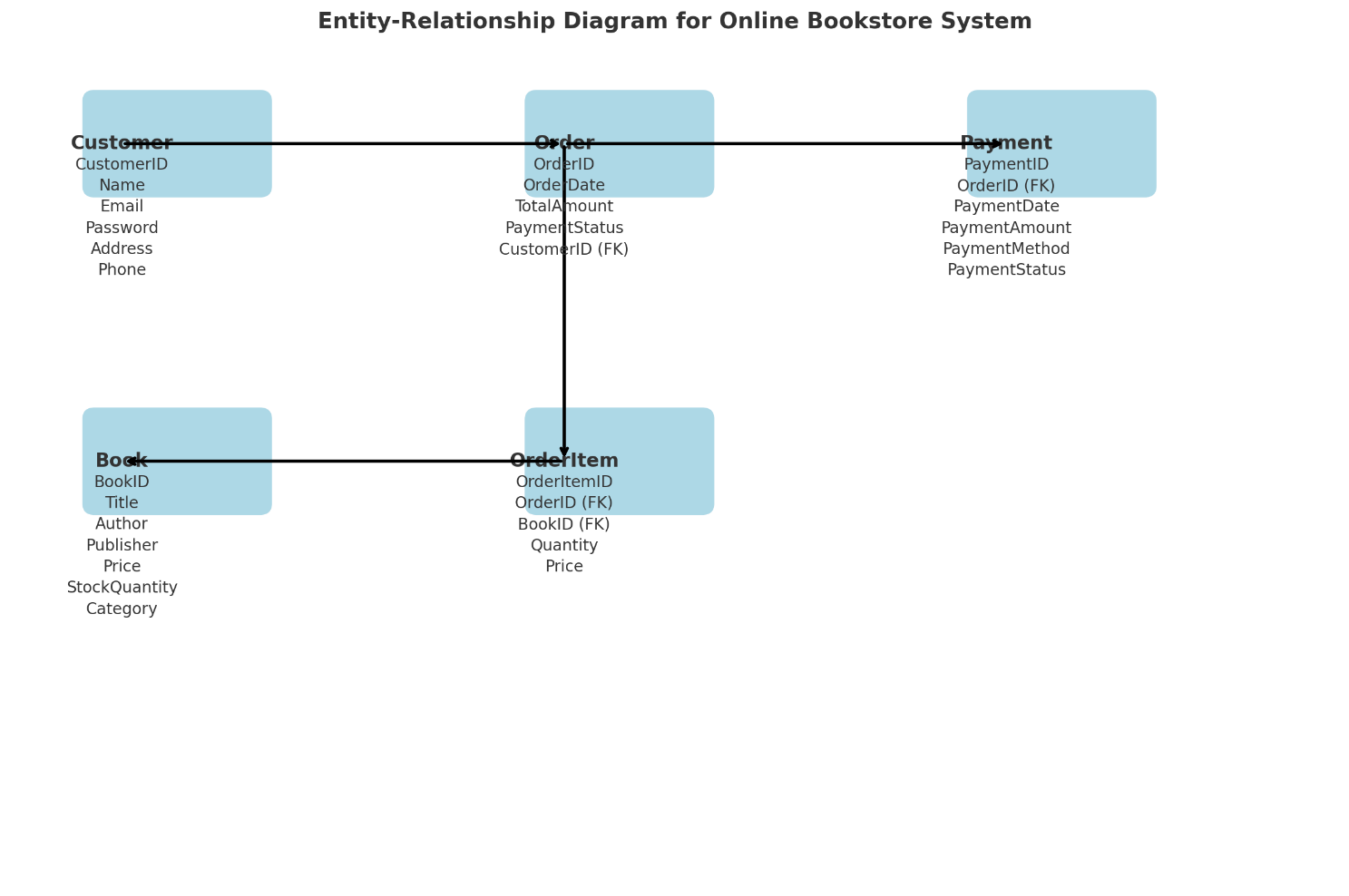
# 2.System Flow Diagrams:

****

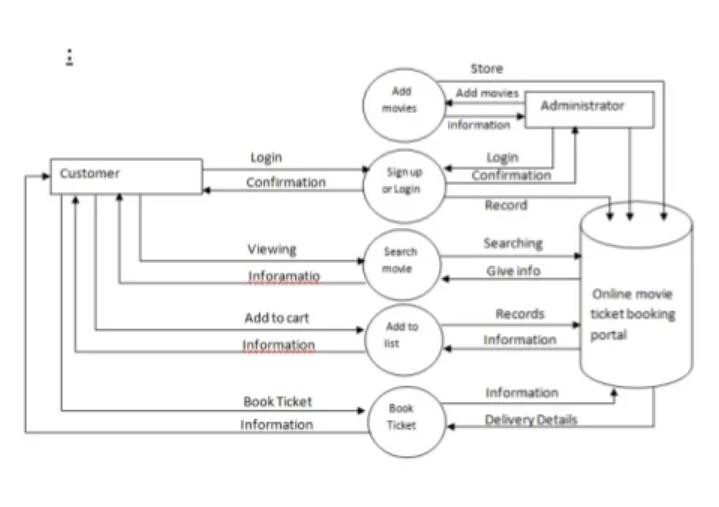
**FIGURE 2.1 USE CASE DIAGRAM**

## 2.2 Entity-relationship diagram:

E-R (Entity-Relationship) Diagram is used to represents the relationship between entities in the table.



## FIGURE 2.2 Entity-relationship diagram

 **FIGURE 2.3 Data-flow diagram**

# 3. Module description:

Register:

Admin can register with the username and password for the registration

Login:

Admin can log in with their username and password.

After Login:

Add book:

In this section, admin can add details about new books,including the title, stock, release date, and other relevant information.

View book:

Admin can view and update book details such as the title, no of seats and release date.

Delete book:

Admin can delete book details, removing books from the system.

Add Reservation:

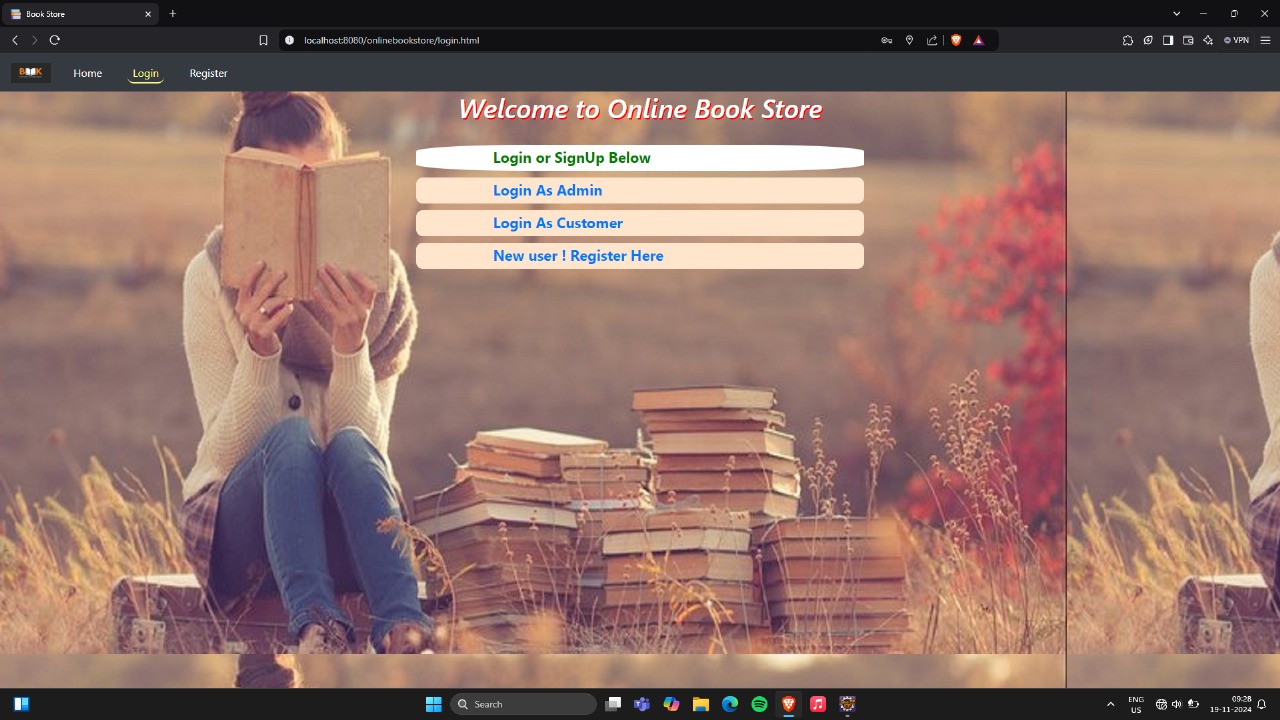
Admin can add reservation details, including the book, date, time, and number of books.

Update Reservation:

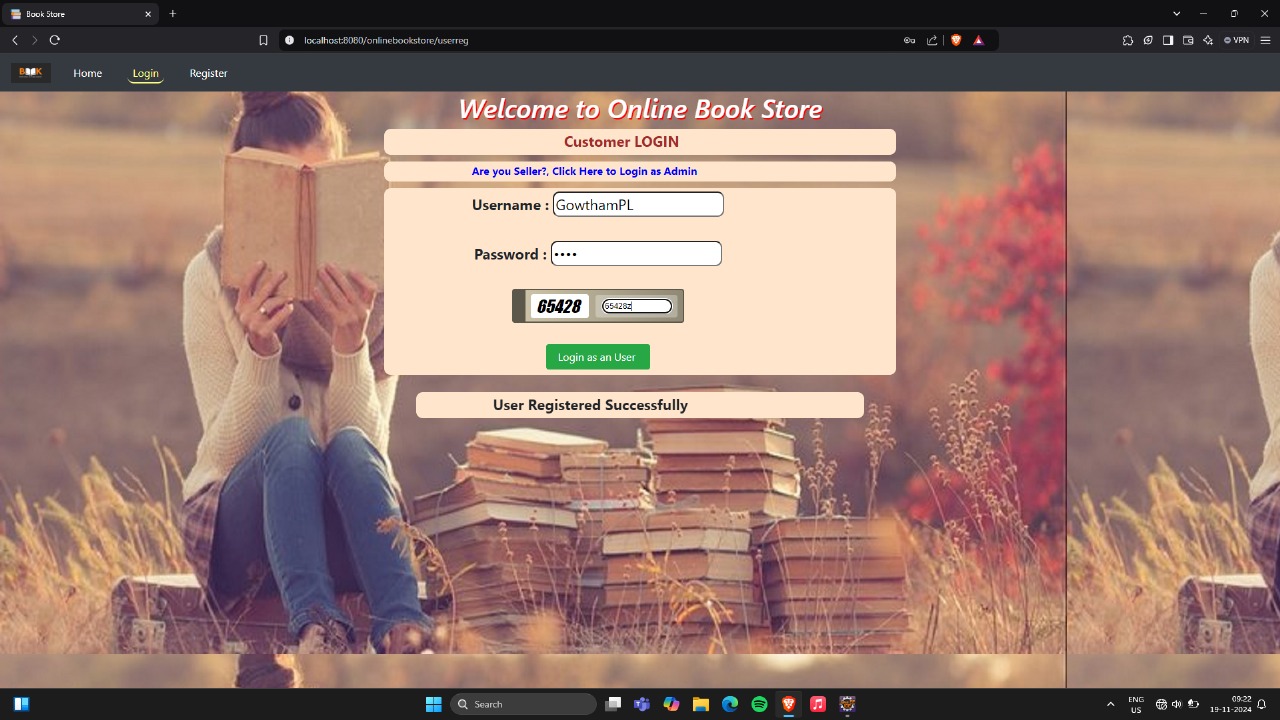
Admin can update reservation details, such as changing the date, time, or the number of reserved books.

# 4.Implementation:

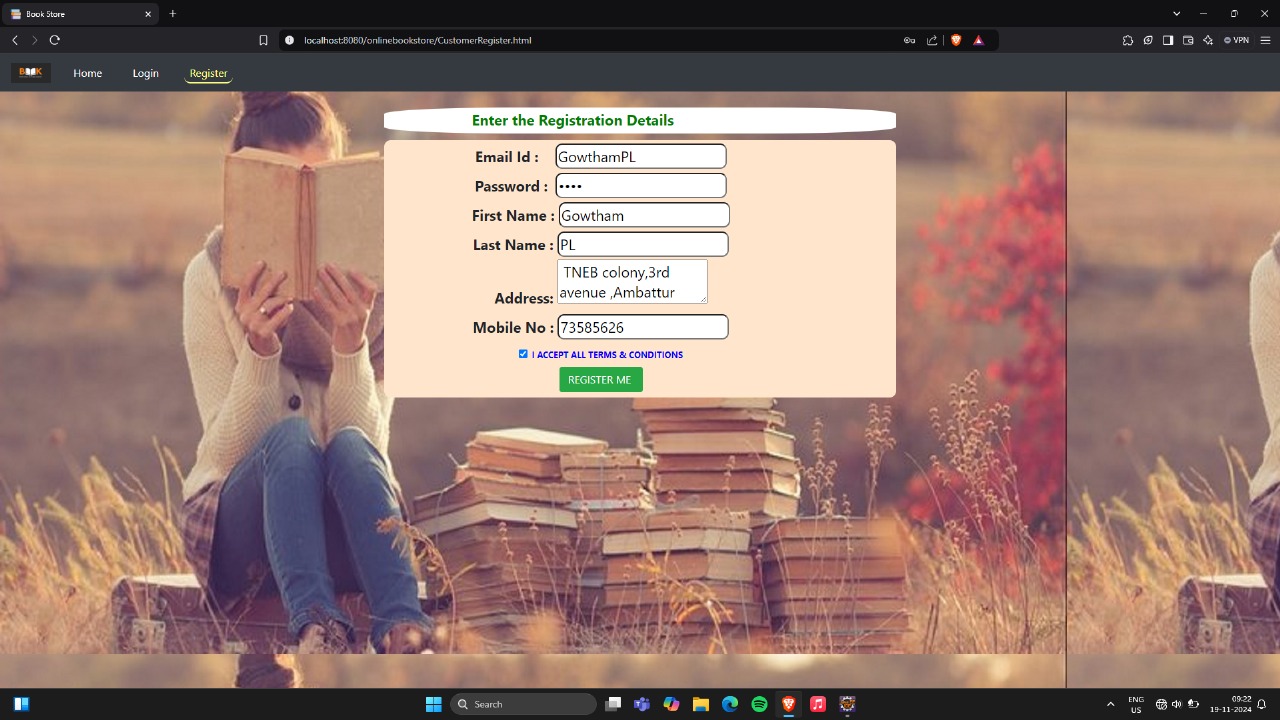
# 



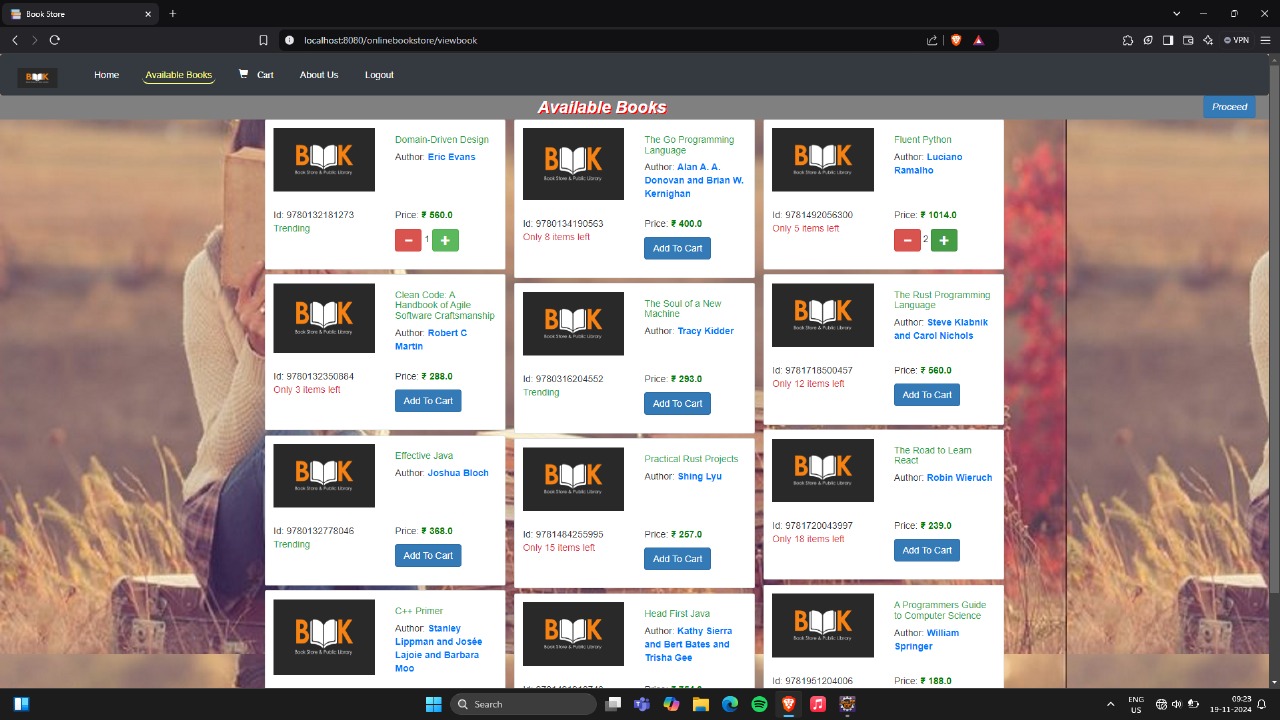
**Figure 4.1: Home Page**



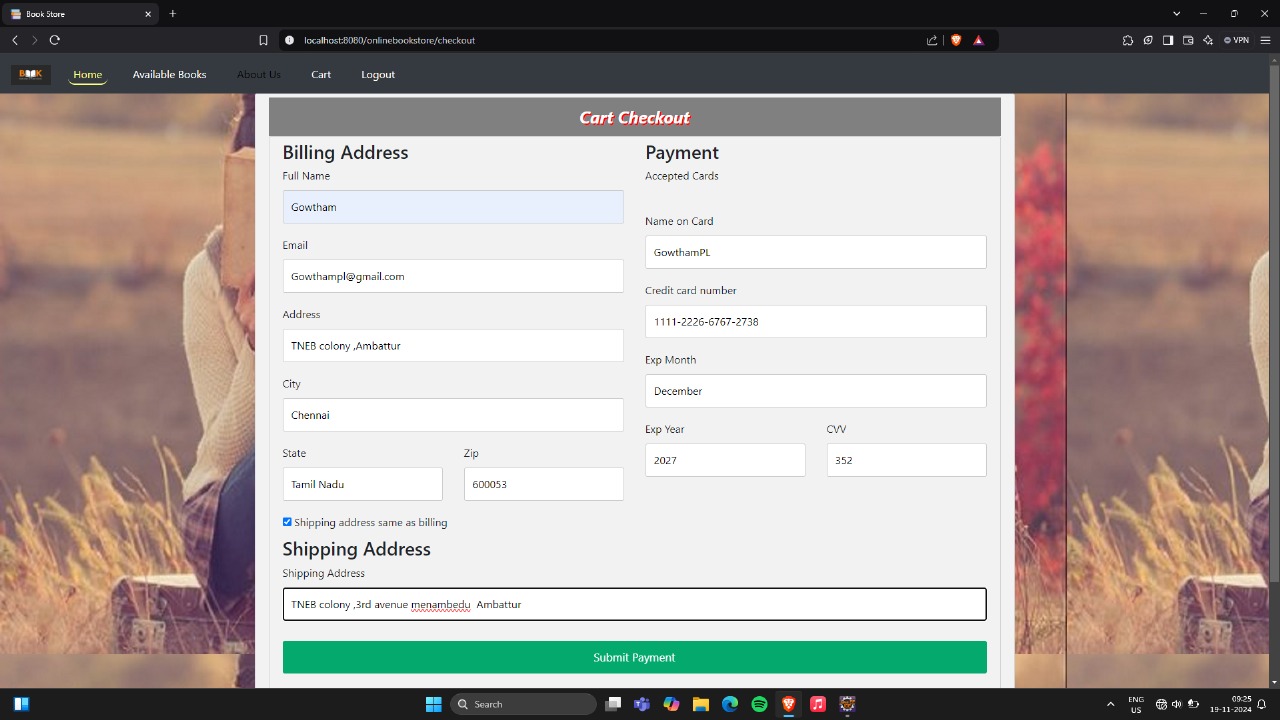
**Figure 4.2: Customer sign-in:**



**Figure 4.3: Main page**



**Figure 4.4: Shopping cart:**



**FIGURE 4.5 PAYMENT**

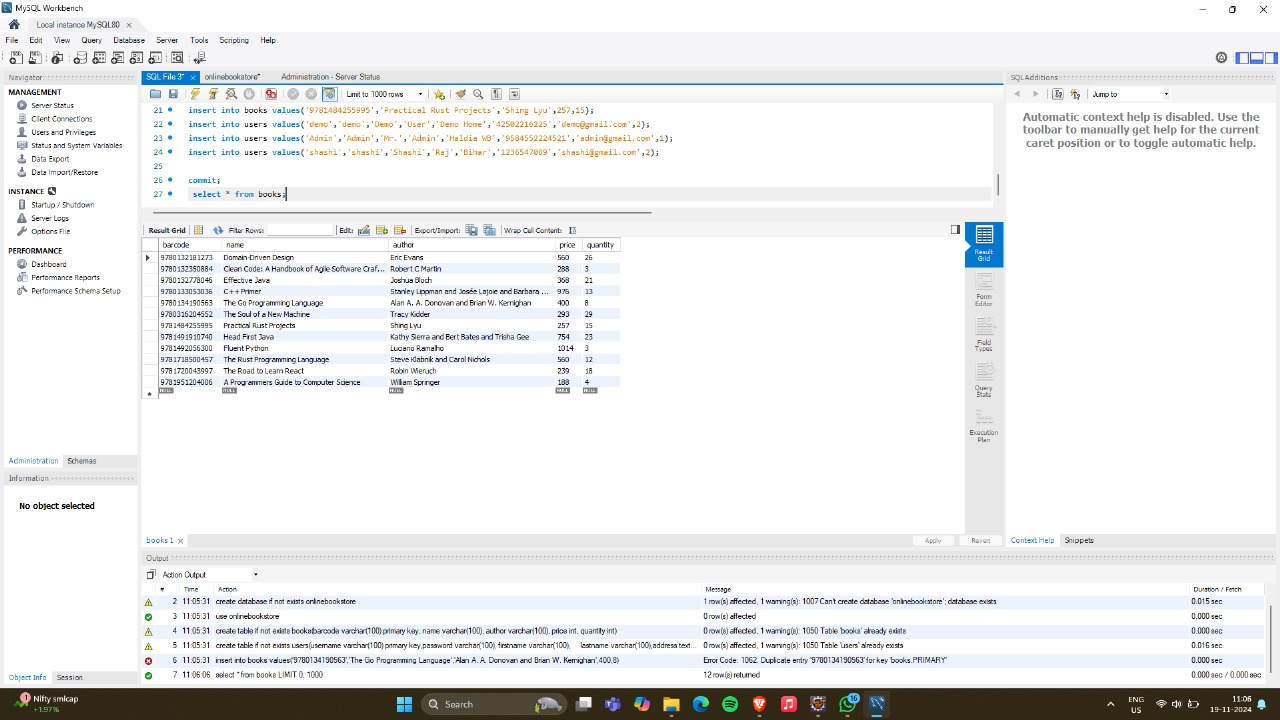
### 4.2 Database Design:

The database design is a crucial part of the system design process, where the data elements and structures identified during the analysis stage are organized for efficient storage and retrieval. A database is a collection of related data, stored with minimal redundancy, to serve multiple users quickly and effectively. The primary goal is to ensure that database access is easy, fast, cost-effective, and flexible. Data relationships are established, and unnecessary items are removed. Normalization is performed to achieve internal consistency, minimize redundancy, and ensure maximum stability. This process helps to reduce the storage requirements, avoid data inconsistencies, and optimize updates. MySQL has been selected as the database management system for developing the necessary databases.

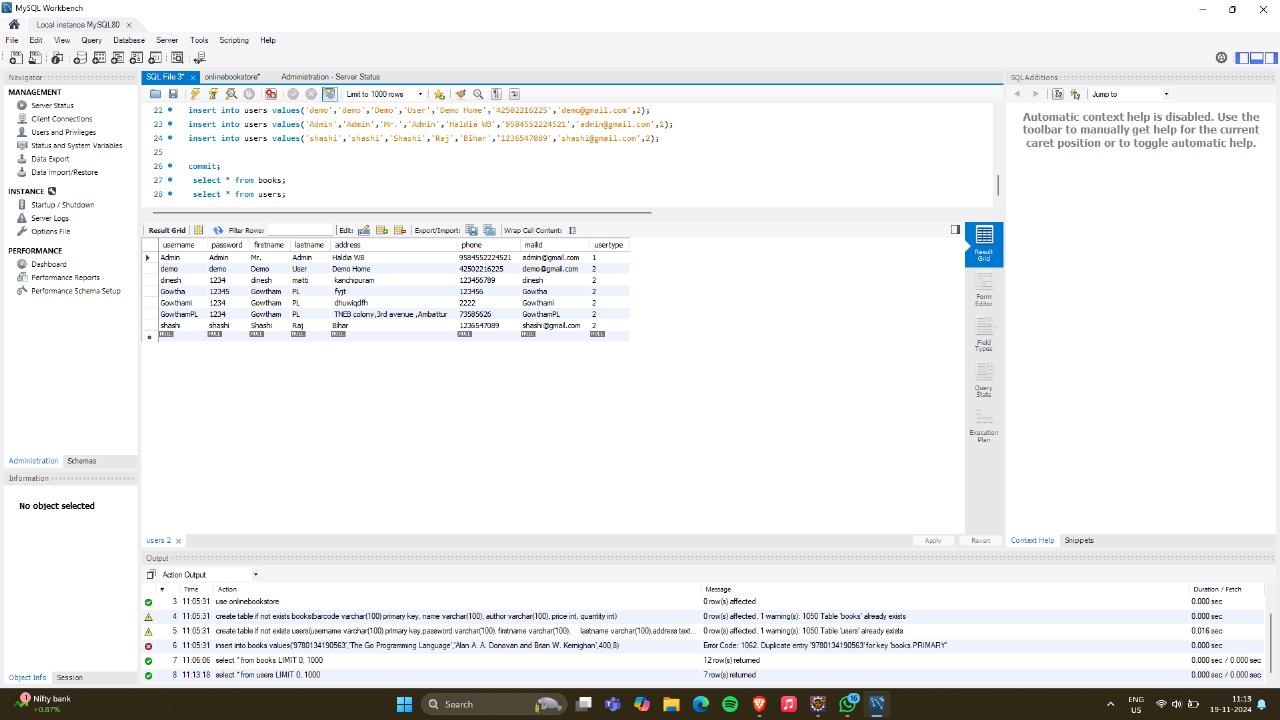
MySQL is a widely-used, open-source relational database management system known for its reliability, scalability, and performance. It is the go-to choice for many web applications, particularly those using the LAMP (Linux, Apache, MySQL, PHP) stack. MySQL enables developers to manage large datasets efficiently, offering features like fast query processing and strong data security. Its structured query language (SQL) makes data retrieval and manipulation intuitive. MySQL’s cross-platform compatibility and seamless integration with various programming languages and frameworks make it ideal for diverse applications, from small-scale websites to enterprise-level databases.

MySQL offers several advantages over other databases like PostgreSQL and Oracle. It is lightweight, making it faster and more efficient for web- based applications. MySQL's ease of use and quick setup process make it accessible for beginners, while its scalability supports large-scale systems. Compared to Oracle, MySQL is more cost-effective, particularly for startups and smaller enterprises. Additionally, MySQL’s broad community support provides extensive resources and documentation. Its performance is particularly optimized for read-heavy workloads, which is ideal for applications requiring fast data retrieval, making it a preferred choice for many businesses.

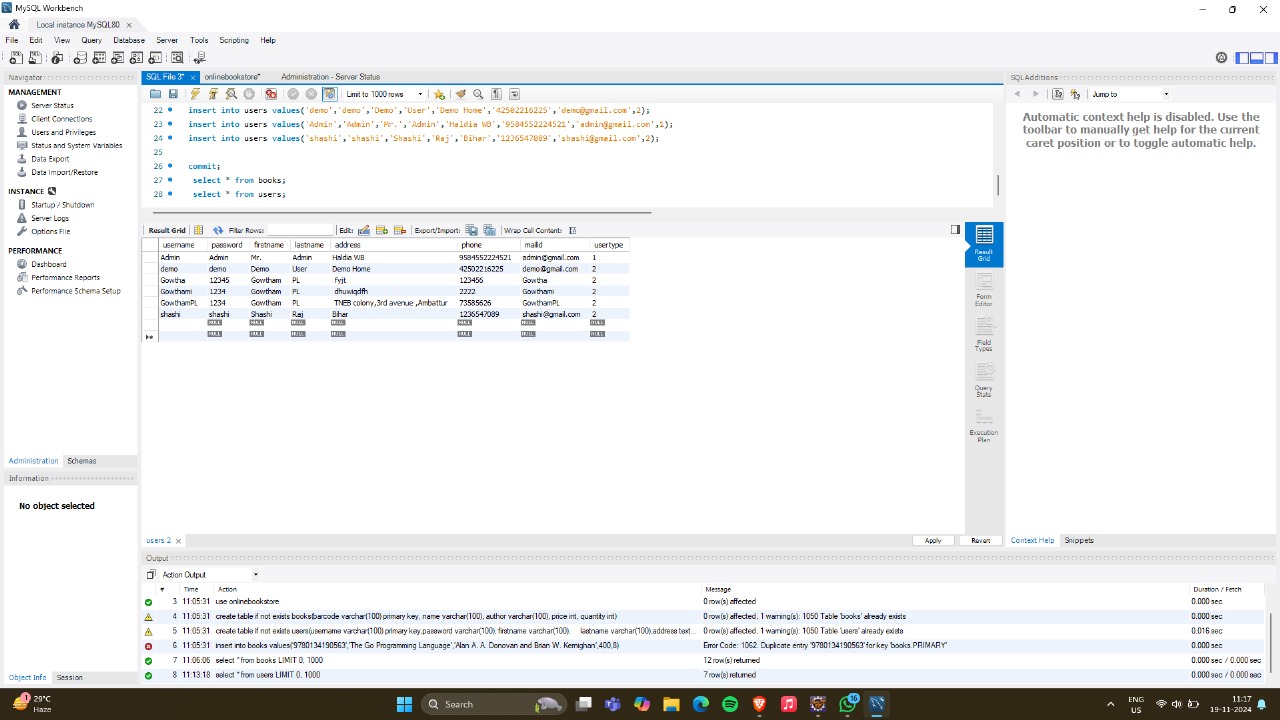
**Online Book Store System which contains 3 MySQL tables :**



**FIGURE 4.2.1 PODUCT DATABASE**



**FIGURE 4.2.2 ORDERS DATABASE**



**FIGURE 4.2.3 USERS DATABASE**

### 4.3 IMPLEMENTATIONS(CODE):

package com.bittercode.model;

import java.io.Serializable;

public class Book implements Serializable {

private String barcode;

private String name;

private String author;

private double price;

private int quantity;

public Book(String barcode, String name, String author, double price, int quantity) {

this.barcode = barcode;

this.name = name;

this.author = author;

this.setPrice(price);

this.quantity = quantity;

}

public Book() {

super();

}

public String getBarcode() {

return barcode;

}

public void setBarcode(String barcode) {

this.barcode = barcode;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

}

package com.bittercode.model;

import java.io.Serializable;

public class Cart implements Serializable {

private Book book;

private int quantity;

public Cart(Book book, int quantity) {

this.book = book;

this.quantity = quantity;

}

public Book getBook() {

return book;

}

public void setBook(Book book) {

this.book = book;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

}

package com.bittercode.model;

import java.io.Serializable;

import java.util.List;

public class User implements Serializable {

private String emailId;

private String password;

private String firstName;

private String lastName;

private Long phone;

private String address;

private List<UserRole> roles;

public String getEmailId() {

return emailId;

}

public void setEmailId(String emailId) {

this.emailId = emailId;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public Long getPhone() {

return phone;

}

public void setPhone(Long phone) {

this.phone = phone;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public List<UserRole> getRoles() {

return roles;

}

public void setRoles(List<UserRole> roles) {

this.roles = roles;

}

// public static User retrieveFromHttpServletRequest(HttpServletRequest req) {

// User user = new User();

// String pWord = req.getParameter(UsersDBConstants.COLUMN\_PASSWORD);

// String fName = req.getParameter(UsersDBConstants.COLUMN\_FIRSTNAME);

// String lName = req.getParameter(UsersDBConstants.COLUMN\_LASTNAME);

// String addr = req.getParameter(UsersDBConstants.COLUMN\_ADDRESS);

// String phNo = req.getParameter(UsersDBConstants.COLUMN\_PHONE);

// String mailId = req.getParameter(UsersDBConstants.COLUMN\_MAILID);

// user.setEmailId(mailId);

// user.setFirstName(fName);

// user.setLastName(lName);

// user.setPassword(pWord);

// user.setPhone(Long.parseLong(phNo));

// user.setAddress(addr);}}

5.Conclusion:

The Online Bookstore System is a robust and user-friendly platform designed to simplify the process of buying and managing books online. By utilizing Java for the frontend, MySQL for the backend, and JDBC for database connectivity, the system ensures smooth and efficient operations. It provides essential features like browsing books, secure transactions, and order management for customers, while offering inventory and user management tools for administrators.

This project successfully demonstrates the integration of various technologies to create a functional e-commerce application, addressing the needs of modern book retail. It lays the foundation for further enhancements, such as recommendations, reviews, and mobile support, making it a scalable and adaptable solution for the online book industry**.**

**6.Reference links:**

* <https://www.javatpoint.com/java-awt>
* <https://www.javatpoint.com/java-swing>
* <https://www.w3schools.com/sql/>
* https://www.geeksforgeeks.org/introduction-to-jdbc/

