**ABSTRACT**

The Library Management System is a software application designed to automate and streamline library operations. This document provides a comprehensive overview of the system, its features, components, and deployment steps. The system follows a three-tier architecture, with a presentation layer implemented using HTML, CSS, and JavaScript, an application layer consisting of Java servlets, and a data layer based on a relational database management system.

The Library Management System offers features such as user registration, book management (including adding and deleting books), book issuing and returning, book requests, and viewing available books. It supports two user roles: librarian and student, each with specific privileges and access to relevant features.

The document outlines the system workflow, starting from user registration to book management, issuing, returning, and requests. It also provides instructions for deploying the system, including setting up the web server environment, configuring the database, and deploying the server-side and client-side code.

The Library Management System improves efficiency and simplifies library management processes by automating tasks and providing a user-friendly interface. By following the documentation, users can effectively utilize and customize the system according to their specific library management requirements.

**PROBLEM STATEMENT**

The traditional manual processes involved in library management are time-consuming, error-prone, and inefficient. Librarians face challenges in maintaining accurate records of books, managing book issuance and returns, and facilitating book requests from students. Students often struggle to find available books and face delays in book issuance.

The problem is the lack of an automated system that can streamline library operations, improve accuracy, and provide a user-friendly interface for librarians and students. There is a need for a comprehensive Library Management System that can address these challenges by automating various tasks such as book management, book issuing and returning, and book requests. The system should provide an intuitive interface for users to search for books, request books, and track their borrowing history.

Additionally, the system needs to support multiple user roles, such as librarians and students, with role-based access control to ensure appropriate privileges and restrictions. It should also have a robust database to store and retrieve book and user information accurately.

The goal is to develop a Library Management System that enhances the efficiency of library operations, improves user experience, reduces manual errors, and provides real-time access to book availability and borrowing information. The system should be easy to deploy, maintain, and customize to meet the specific requirements of different libraries.

**SYSTEM REQUIREMENTS**

Language Used : HTML & Java Programming Language

IDE Tool : Eclipse

Server : Tomcat Server Version 9.0

Database : MySQL

Web Browser : Google Chrome

**HTML & Java Programming Language**

HTML (Hypertext Markup Language) is the standard markup language for creating web pages. It provides the structure and content of a web page by using various elements and tags. HTML is used to define the layout, headings, paragraphs, images, links, and other components of a webpage. It is a client-side language, meaning it runs on the user's web browser.

Java is a versatile and widely used programming language that allows developers to build robust and scalable applications. It is known for its platform independence, as Java programs can run on any operating system that has a Java Virtual Machine (JVM). Java is commonly used for server-side programming, and it provides powerful libraries and frameworks for web development.

**Eclipse**

Eclipse is a popular integrated development environment (IDE) that provides a comprehensive set of tools for Java development. It offers features such as code editing, debugging, testing, and deployment capabilities. Eclipse has a user-friendly interface and supports plugins for various programming languages, making it a preferred choice for many developers.

**Tomcat**

Apache Tomcat is an open-source web server and servlet container that is widely used for hosting Java-based web applications. It provides an environment for running Java Servlets, JavaServer Pages (JSP), and other Java web technologies. Tomcat is known for its simplicity, lightweight nature, and compatibility with different operating systems. Version 9.0 is one of the latest stable releases of Tomcat.

**MySQL**

MySQL is a popular open-source relational database management system (RDBMS) that is widely used in web development. It provides a reliable and scalable platform for storing and managing structured data. MySQL supports standard SQL queries and offers various features like transactions, indexing, and replication. It is a preferred choice for many web applications due to its performance, ease of use, and community support.

**Google Chrome**

Google Chrome is a widely used web browser developed by Google. It offers a fast and secure browsing experience and supports modern web technologies. Chrome is known for its compatibility with HTML, CSS, and JavaScript, making it an ideal choice for testing and running web applications. It provides developer tools for debugging and analyzing web pages, making it a preferred browser for web developers.

Using HTML and Java programming language, along with Eclipse as the IDE, Tomcat Server version 9.0 for hosting the application, MySQL as the database, and Google Chrome as the web browser, provides a robust and reliable foundation for developing a web-based application for AJP Hospital. These technologies and tools are well-established, widely used, and offer extensive community support, ensuring a smooth development process and efficient execution of the application.

**MODULE DESCRIPTION**

**User Management Module:**

- This module handles user registration, authentication, and role management. It allows librarians and students to create their accounts and login to the system.

- Functionalities:

- User registration: Librarians and students can register their accounts by providing the required information.

- User login: Users can authenticate themselves by entering their credentials.

- Role management: The system assigns appropriate roles (librarian or student) to registered users.

**Book Management Module:**

- This module deals with the management of books in the library. It allows librarians to add new books, delete existing books, and update book details.

- Functionalities:

- Add book: Librarians can add new books to the system by providing the book ID, title, author, and quantity.

- Delete book: Librarians can remove books from the system based on the book ID.

- Update book details: Librarians can modify the information related to a book, such as title, author, or quantity.

**Book Issuance Module:**

- This module handles the process of issuing books to students. It enables librarians to issue books to students and keep track of borrowed books.

- Functionalities:

- Issue book: Librarians can issue a book to a student by providing the book ID and student ID.

- Track borrowed books: The system maintains a record of borrowed books, including the book details, student details, and due date.

**Book Return Module:**

- This module manages the return of books by students. It allows students to return books they have borrowed and updates the book availability status.

- Functionalities:

- Return book: Students can initiate the return process by providing the book ID. The system updates the book status and removes it from the borrowed books list.

**Book Request Module:**

- This module enables students to request books that are currently unavailable. It allows them to submit book requests to librarians.

- Functionalities:

- Request book: Students can submit a request for a specific book by providing the book ID. The request is stored in the system for librarian review.

**Book Search Module:**

- This module provides a search functionality for users to find available books in the library. It allows users to search books by title, author, or other criteria.

- Functionalities:

- Search book: Users can search for books by entering keywords, author names, or other search criteria. The system retrieves and displays the matching books.

These modules collectively form the Library Management System, providing a comprehensive solution for managing books, users, book issuance, returns, and requests. The system aims to automate library operations, improve efficiency, and enhance the overall library experience for both librarians and students.

**TABLE DESCRIPTION**

**Users Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** |
| Id | Int | No | Primary | Null |
| Username | Varchar(255) | No |  | Null |
| Password | Varchar(255) | No |  | Null |
| Role | Enum | No |  | Null |

- Description: This table stores information about the users of the library management system, including librarians and students.

**Books Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** |
| Id | Int | No | Primary | Null |
| Title | Varchar(255) | No |  | Null |
| Author | Varchar(255) | No |  | Null |
| Quantity | Int | No |  | Null |

- Description: This table stores information about the books available in the library.

**Issued Books Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** |
| Book Id | Int | No | MUL | Null |
| User Id | Int | No | MUL | Null |

- Description: This table keeps track of the books issued to students.

**Book Requests Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** |
| Book Id | Int | No | MUL | Null |
| User Id | Int | No | MUL | Null |

- Description: This table stores the book requests made by students.

**HTML CODE**

**Index**

<!DOCTYPE html>

<html>

<head>

<title>Library Management System</title>

<style>

**body** {

font-family: *Arial, sans-serif*;

background-color: *#f5f5f5*;

text-align: *center*;

}

**h1** {

color: *#333*;

}

**form** {

margin-top: *20px*;

background-color: *#fff*;

padding: *20px*;

border: *1px solid #ccc*;

border-radius: *5px*;

width: *300px*;

margin-left: *auto*;

margin-right: *auto*;

}

**label** {

display: *block*;

margin-bottom: *5px*;

font-weight: *bold*;

}

**select** {

padding: *5px*;

width: *100%*;

margin-bottom: *10px*;

border: *1px solid #ccc*;

border-radius: *3px*;

}

**input**[type="submit"] {

padding: *10px 20px*;

background-color: *#007bff*;

color: *#fff*;

border: *none*;

border-radius: *5px*;

cursor: *pointer*;

width: *100%*;

}

**input**[type="submit"]*:hover* {

background-color: *#0056b3*;

}

</style>

<script>

**function**updateActions() {

**var** role = document.getElementById("role").value;

**var**actionOptions = document.getElementById("action").options;

**for** (**var**i = 0; i<actionOptions.length; i++) {

**var** option = actionOptions[i];

**if** ((role === "librarian" &&option.classList.contains("librarian-action")) || (role === "student" &&option.classList.contains("student-action"))) {

option.style.display = "block";

} **else** {

option.style.display = "none";

}

}

}

</script>

</head>

<body>

<h1>Welcome to the Library Management System</h1>

<form action=*"LibraryServlet"* method=*"post"*>

<label for=*"role"*>Select your role:</label>

<select name=*"role"* id=*"role"*onchange=*"updateActions()"*>

<option value=*"librarian"*>Librarian</option>

<option value=*"student"*>Student</option>

</select>

<br><br>

<label for=*"action"*>Select an action:</label>

<select name=*"action"* id=*"action"*>

<option value=*"addBook"* class=*"librarian-action"*>Add Book</option>

<option value=*"viewBooks"* class=*"librarian-action student-action"*>View Books</option>

<option value=*"deleteBook"* class=*"librarian-action"*>Delete Book</option>

<option value=*"issueBook"* class=*"librarian-action"*>Issue Book</option>

<option value=*"returnBook"* class=*"librarian-action student-action"*>Return Book</option>

<option value=*"requestBook"* class=*"student-action"*>Request Book</option>

</select>

<br><br>

<input type=*"submit"* value=*"Submit"*>

</form>

</body>

</html>

**JAVA CODE**

**Main Code**

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.\*;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/LibraryServlet")

public class LibraryServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

// JDBC database URL, username, and password

private static final String DB\_URL = "jdbc:mysql://localhost:3306/library1";

private static final String DB\_USER = "root";

private static final String DB\_PASS = "mysql";

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String role = request.getParameter("role");

String action = request.getParameter("action");

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASS);

if (role.equals("librarian")) {

if (action.equals("addBook"))

{

String idParam = request.getParameter("id");

String title = request.getParameter("title");

String author = request.getParameter("author");

String quantityParam = request.getParameter("quantity");

// Check if any of the parameters are null or empty

if (idParam == null || title == null || author == null || quantityParam == null )

{

out.println("Invalid book details. Please fill in all the fields.");

}

else

{

try {

int id = Integer.parseInt(idParam);

int quantity = Integer.parseInt(quantityParam);

PreparedStatementstmt = conn.prepareStatement

("INSERT INTO books (id, title, author, quantity) VALUES (?, ?,?,?)");

stmt.setInt(1, id);

stmt.setString(2, title);

stmt.setString(3, author);

stmt.setInt(4, quantity);

int rows = stmt.executeUpdate();

if (rows > 0) {

out.println("Book added successfully!");

} else {

out.println("Failed to add book!");

}

stmt.close();

} catch (NumberFormatException e) {

out.println("Invalid id or quantity. Please enter valid integer values.");

} catch (SQLException e) {

out.println("An error occurred: " + e.getMessage());

}

}

} else if (action.equals("viewBooks")) {

Statement stmt = conn.createStatement();

ResultSetrs = stmt.executeQuery("SELECT \* FROM books");

out.println("<table>");

out.println("<tr><th>ID</th><th>Title</th><th>Author</th><th>Quantity</th></tr>");

while (rs.next()) {

int id = rs.getInt("id");

String title = rs.getString("title");

String author = rs.getString("author");

int quantity = rs.getInt("quantity");

out.println("<tr><td>" + id + "</td><td>" + title + "</td><td>" + author + "</td><td>" + quantity + "</td></tr>");

}

out.println("</table>");

rs.close();

stmt.close();

} else if (action.equals("deleteBook")) {

String bookIdParam = request.getParameter("bookId");

if (bookIdParam == null || bookIdParam.isEmpty()) {

out.println("Invalid book ID. Please provide a valid book ID.");

} else {

int bookId = Integer.parseInt(bookIdParam);

PreparedStatementstmt = conn.prepareStatement("DELETE FROM books WHERE id = ?");

stmt.setInt(1, bookId);

int rows = stmt.executeUpdate();

if (rows > 0)

{

out.println("Book deleted successfully!");

} else

{

out.println("Failed to delete book!");

}

stmt.close();

}

} else if (action.equals("issueBook"))

{

String bookIdParam = request.getParameter("bookId");

String userIdParam = request.getParameter("userId");

if (bookIdParam == null || bookIdParam.isEmpty() || userIdParam == null || userIdParam.isEmpty()) {

out.println("Invalid book ID or user ID. Please provide valid IDs.");

}

else

{

int bookId = Integer.parseInt(bookIdParam);

int userId = Integer.parseInt(userIdParam);

PreparedStatementstmt = conn

.prepareStatement("INSERT INTO issued\_books (book\_id, user\_id) VALUES (?, ?)");

stmt.setInt(1, bookId);

stmt.setInt(2, userId);

int rows = stmt.executeUpdate();

if (rows > 0) {

out.println("Book issued successfully!");

} else

{

out.println("Failed to issue book!");

}

stmt.close();

}

} else if (action.equals("returnBook"))

{

String bookIdParam = request.getParameter("bookId");

if (bookIdParam == null || bookIdParam.isEmpty())

{

out.println("Invalid book ID. Please provide a valid book ID.");

} else

{

int bookId = Integer.parseInt(bookIdParam);

PreparedStatementstmt = conn.prepareStatement("DELETE FROM issued\_books WHERE book\_id = ?");

stmt.setInt(1, bookId);

int rows = stmt.executeUpdate();

if (rows > 0)

{

out.println("Book returned successfully!");

} else

{

out.println("Failed to return book!");

}

stmt.close();

}

}

} else if (role.equals("student"))

{

if (action.equals("viewBooks")) {

Statement stmt = conn.createStatement();

ResultSetrs = stmt.executeQuery("SELECT \* FROM books");

out.println("<table>");

out.println("<tr><th>ID</th><th>Title</th><th>Author</th><th>Quantity</th></tr>");

while (rs.next()) {

int id = rs.getInt("id");

String title = rs.getString("title");

String author = rs.getString("author");

int quantity = rs.getInt("quantity");

out.println("<tr><td>" + id + "</td><td>" + title + "</td><td>" + author + "</td><td>"+ quantity + "</td></tr>");

}

out.println("</table>");

rs.close();

stmt.close();

} else if (action.equals("requestBook"))

{

String bookIdParam = request.getParameter("bookId");

if (bookIdParam == null || bookIdParam.isEmpty()) {

out.println("Invalid book ID. Please provide a valid book ID.");

} else

{

int bookId = Integer.parseInt(bookIdParam);

PreparedStatement stmt = conn

.prepareStatement("INSERT INTO requested\_books (book\_id) VALUES (?)");

stmt.setInt(1, bookId);

int rows = stmt.executeUpdate();

if (rows > 0)

{

out.println("Book requested successfully!");

} else

{

out.println("Failed to request book!");

}

stmt.close();

}

} else if (action.equals("returnBook"))

{

String bookIdParam = request.getParameter("bookId");

if (bookIdParam == null || bookIdParam.isEmpty())

{

out.println("Invalid book ID. Please provide a valid book ID.");

} else

{

int bookId = Integer.parseInt(bookIdParam);

PreparedStatementstmt = conn.prepareStatement("DELETE FROM issued\_books WHERE book\_id = ?");

stmt.setInt(1, bookId);

int rows = stmt.executeUpdate();

if (rows > 0)

{

out.println("Book returned successfully!");

} else

{

out.println("Failed to return book!");

}

stmt.close();

}

}

}

conn.close();

} catch (Exception e) {

out.println("An error occurred: " + e.getMessage());

}

}}

**CONCLUSION**

The Library Management System is an efficient and user-friendly solution for managing library operations. It provides a range of functionalities to librarians and students, including user management, book management, book issuance, book return, book request, and book search. By automating various tasks and streamlining processes, the system enhances the efficiency and effectiveness of library operations.Through the user management module, librarians and students can easily register and login to the system, ensuring secure access to the library resources. The book management module enables librarians to add new books, delete existing books, and update book details, facilitating proper cataloging and organization of the library collection.

The book issuance module simplifies the process of issuing books to students, ensuring accurate tracking of borrowed books and due dates. The book return module allows students to return books, updating the availability status and making the books accessible to other users.The book request module provides students with the ability to request books that are currently unavailable, improving access to desired resources and facilitating better user satisfaction. The book search module enables users to quickly find books based on various search criteria, enhancing the discoverability and accessibility of library materials.

Overall, the Library Management System optimizes library operations, promotes effective resource utilization, and improves the user experience for both librarians and students. It helps streamline processes, reduce manual efforts, and ensure a smooth functioning of the library, ultimately contributing to an efficient and well-organized library environment.

**SCREENSHOTS**

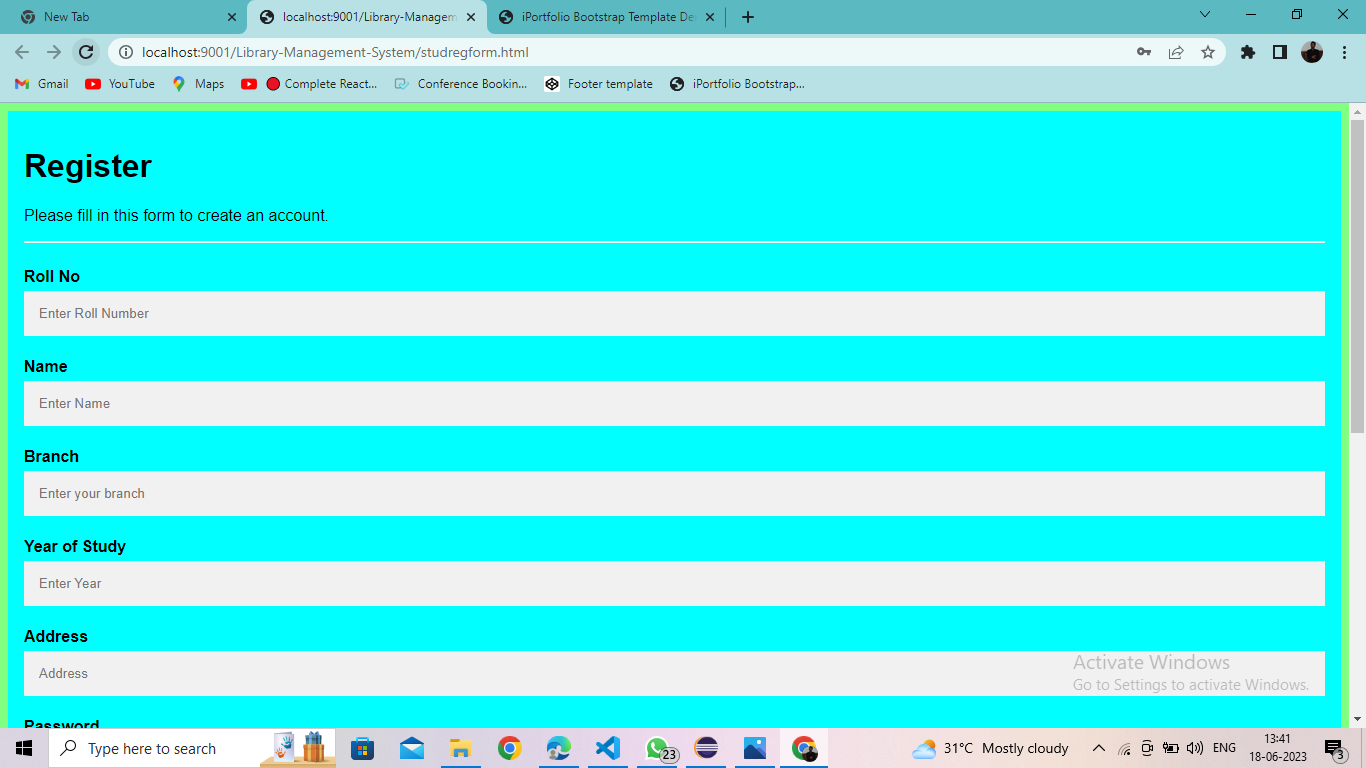
****

Figure 1 Register Page

The Figure above depicts the website where students can establish an account by entering

their information and then quickly login to the page.

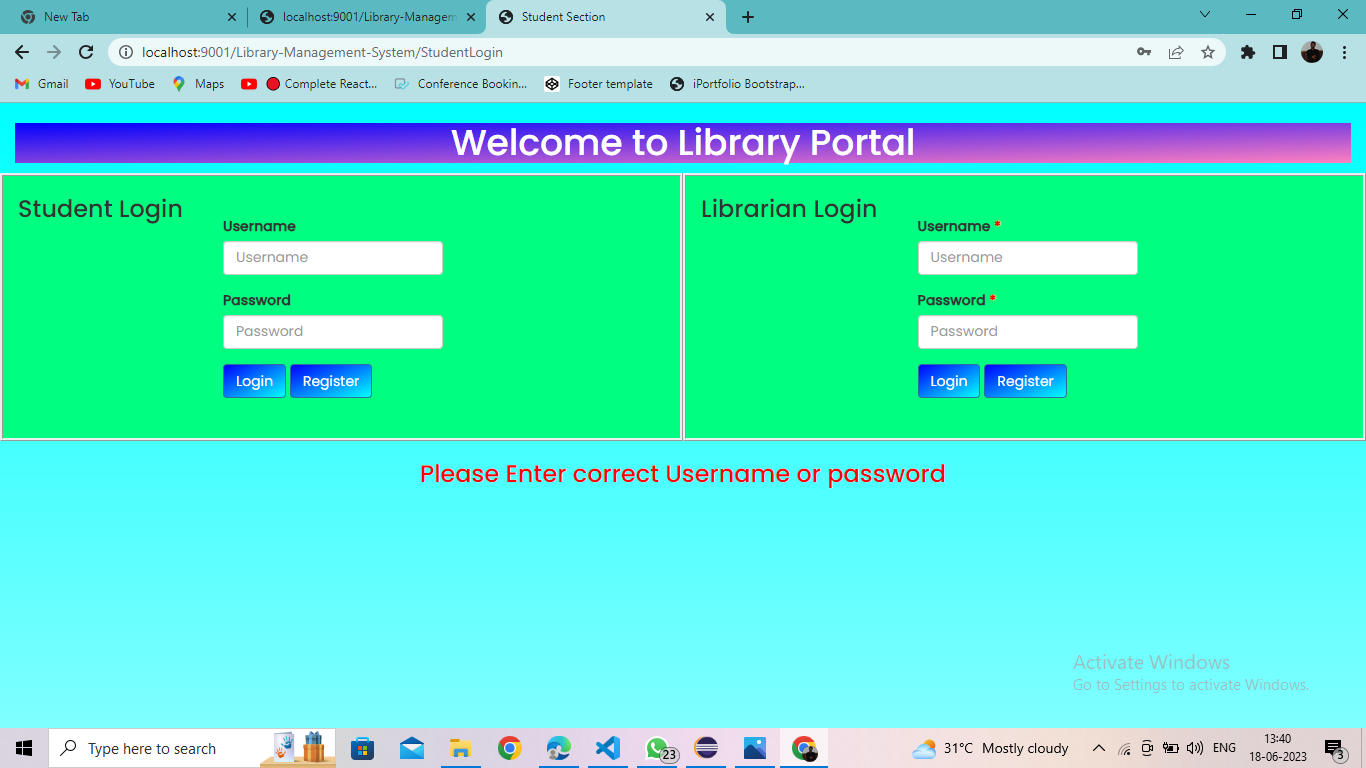
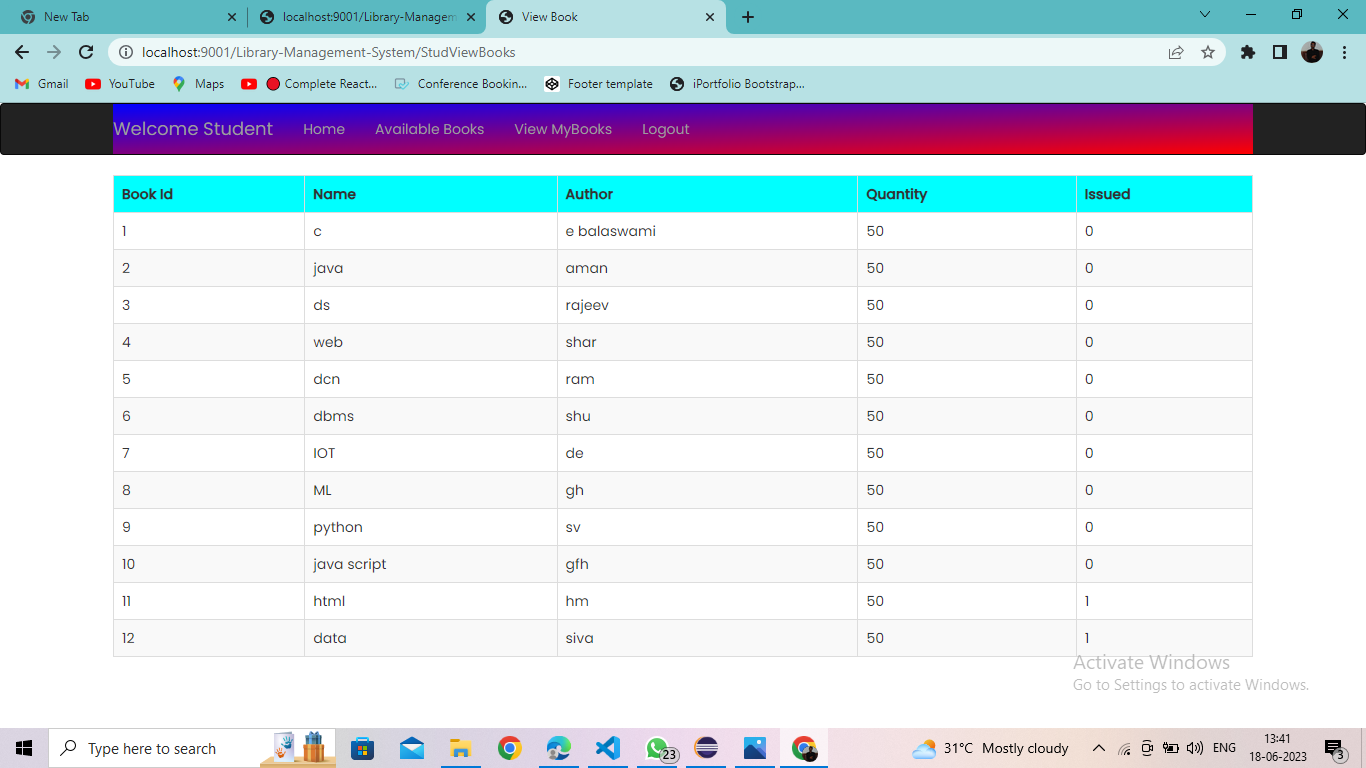


Figure 2 Portal Page

The Figure above depicts the gateway page to which the student or librarian can log in using their username and password.

Figure 3 Book Details

The figure above depicts book information such as book Id, name, author, quantity, and issued details.

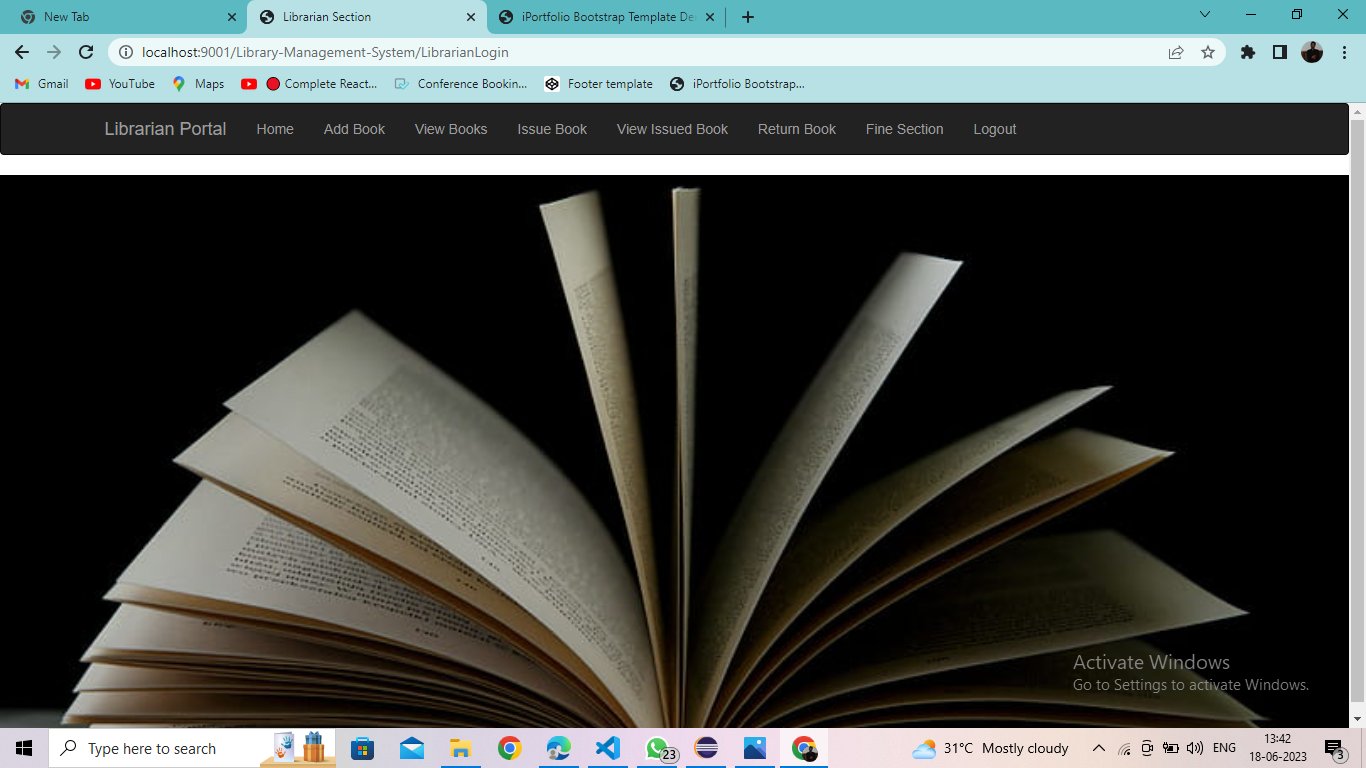
****

Figure 4 Librarian Portal

The figure above depicts a portal for librarians where they can monitor all library-related activity.

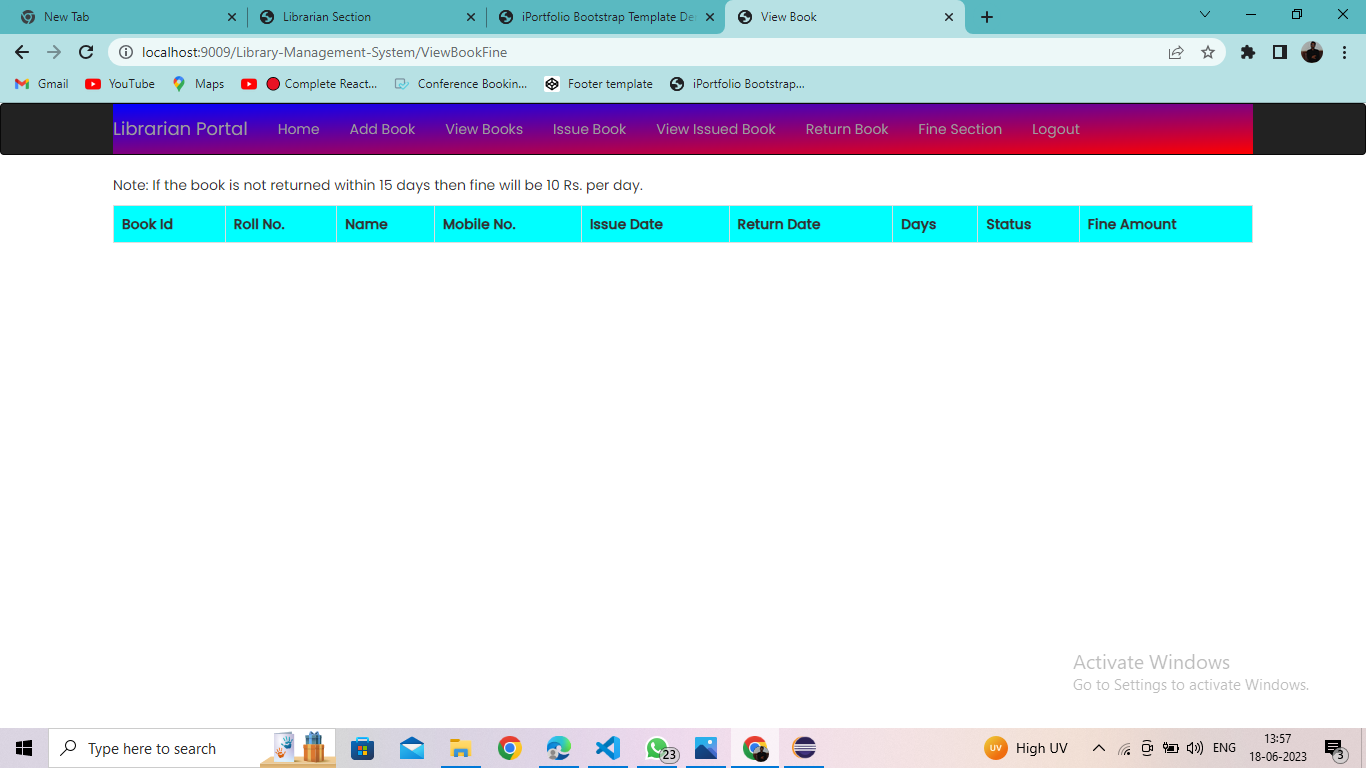
****

Figure 5 Librarian Portal Maintenance

This Figure depicts how the librarian can keep track of books issued, returned, due dates, and fine amounts for students.

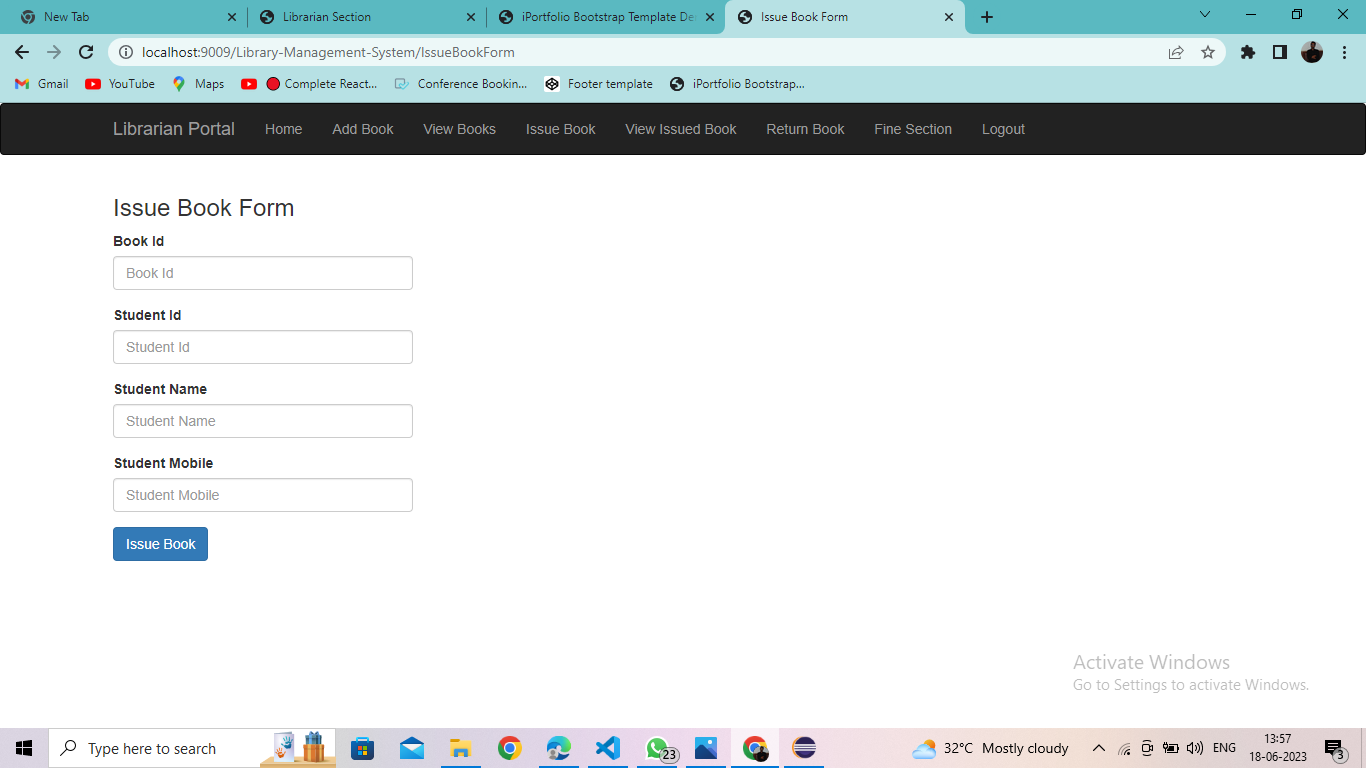
****

Figure 6 Issue Book Form

The Figure above depicts the issue book form details for student.

**REFERENCES**

**Books**

Herbert Schild, “Java: The Complete Reference”, 12th Edition, McGrawHill, 2021(Unit I-

III)

CDAC, “Core and Advanced Java - Black Book”, 1st Edition, Dreamtech Press, 2018 (Unit

IV- V)

Cay Horstmann, “Core Java -Volume 1: Fundamentals”, 12th Edition, Oracle Press, 2021

**References**

<https://projectgurukul.org/java-library-management-system/>

<https://code-projects.org/library-management-system-in-java-with-source-code/>

<https://github.com/OSSpk/Library-Management-System-JAVA>

<https://www.slideshare.net/pragatipilaniya/library-management-system-using-java-technology>