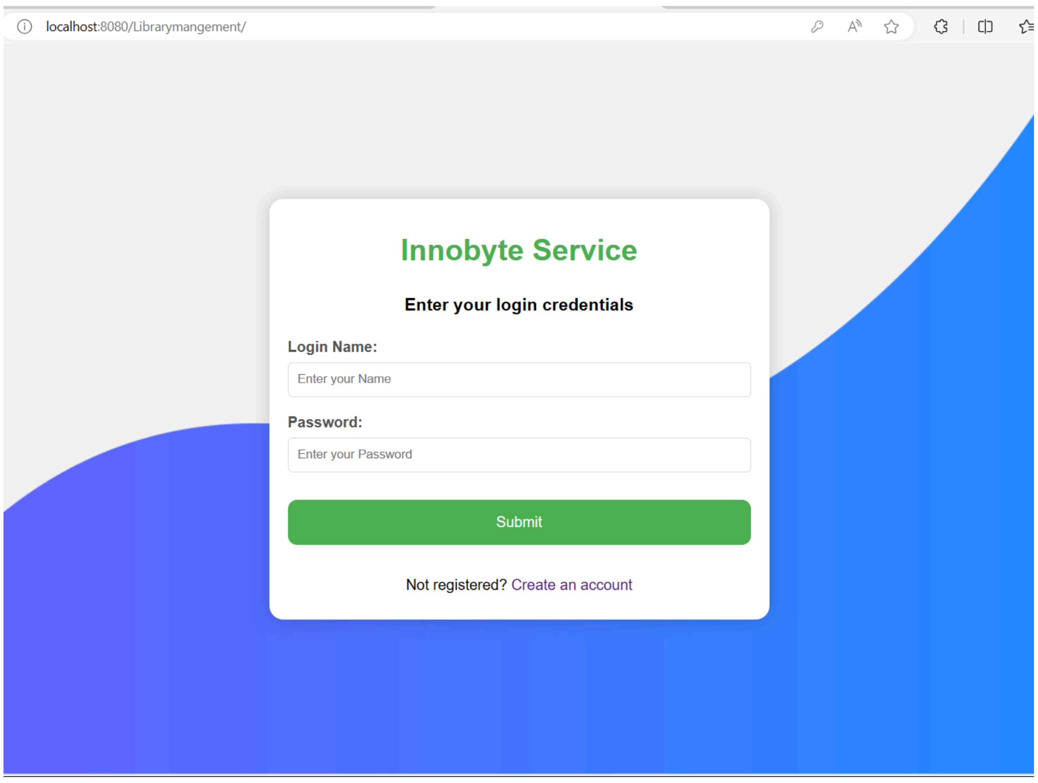
1.login page :index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>

<head>
      <title>Innobyte Library Login</title>
      <link rel="stylesheet"
            href="loginStyle.css">

<style>
body {
  background-image:
url('https://www.innobytess.com/assets/images/background.png');
}
</style>
</head>


<body>

      <div class="main">
            <h1>Innobyte Service</h1>
            <h3>Enter your login credentials</h3>
            <form action="LoginValid" method="get">
            <label for="Enter your Name">
                        Login Name:
                  </label>
                  <input type="Text"
                        id="Name"
                        name="Name"
                        placeholder="Enter your Name" required>


                  <label for="password">
                        Password:
                  </label>
                  <input type="password"
                        id="password"
                        name="password"
                        placeholder="Enter your Password" required>

                  <div class="wrap">

                        <button type="submit">
                              Submit
                        </button>
                  </div>
            </form>
            <script type="text/javascript">
```

```
                  console.log("Value to pass Name: " +
document.getElementById("Name").value);
                  console.log("Value to pass pass: " +
document.getElementById("password").value);

                  </script>
            <p>Not registered?
                  <a href="AccountCreate.html"
                  style="text-decoration: none;">
                        Create an account
                  </a>
            </p>
      </div>
</body>

</html>
```



Innobyte Service

Enter your login credentials

Login Name:

Enter your Name

Password:

Enter your Password

Submit

Not registered? Create an account

This HTML code represents a simple login page for the Innobyte Service. It allows users to input their login credentials (username and password) and submit them for validation. The page includes fields for entering a username and a password, along with a submit button. Additionally, there's a link for users to navigate to an account creation page if they are not registered yet.

The page is styled using CSS, with an external stylesheet (`loginStyle.css`) for additional styling, and it sets a background image using inline CSS. JavaScript is used to log the values of the username and password fields when the page loads.

Overall, the page provides a basic interface for users to log in to the Innobyte Service.

2.Servlet for login validation : LoginValid.java

```
package loginvalid;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
 * Servlet implementation class LoginValid
 */
@WebServlet("/LoginValid")
public class LoginValid extends HttpServlet {


	private static final long serialVersionUID = -2148945972546517189L;


	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {


		String Name=request.getParameter("Name");
		String Pass=request.getParameter("password");
		String url = "jdbc:mysql://localhost:3306/db";
    String user = "root";
    String dbPassword = "root";

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

```
            Connection conn = DriverManager.getConnection(url, user, dbPassword);
        String sql = "SELECT * FROM users WHERE username = ? AND passwords = ?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, Name);
            pstmt.setString(2, Pass);
            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                                    System.out.println("login success");
                                    response.sendRedirect("Library.html");
            } else{
                                    System.out.println("login failed");
                                    response.sendRedirect("index.jsp");
                            }
        }


                    catch (Exception e) {

            e.printStackTrace();
        }
    }
}
```

This servlet, named `LoginValid`, serves as the backend logic for handling user login authentication. Let's break down its functionality:

1. Servlet Configuration: The servlet is annotated with `@WebServlet("/LoginValid")`, specifying its URL mapping.
2. doGet Method: This method is invoked when an HTTP GET request is sent to the servlet, typically when the user submits the login form.
3. Retrieving Parameters: The servlet retrieves the username (`Name`) and password (`password`) parameters from the request using `request.getParameter()`.
4. Database Connection: It establishes a connection to the MySQL database using JDBC, specifying the database URL, username, and password.
5. Query Execution: The servlet prepares a SQL query to select user data from the database table `users` where the provided username and password match.
6. Parameterized Query: It uses a `PreparedStatement` to prevent SQL injection attacks by setting parameters dynamically using `setString()`.
7. Result Processing: The servlet executes the query and processes the result set. If a row is returned, indicating that the username and password match an existing user, it prints "login success" to the console and redirects the user to

`Library.html`. Otherwise, it prints "login failed" and redirects the user back to the login page (`index.jsp`).

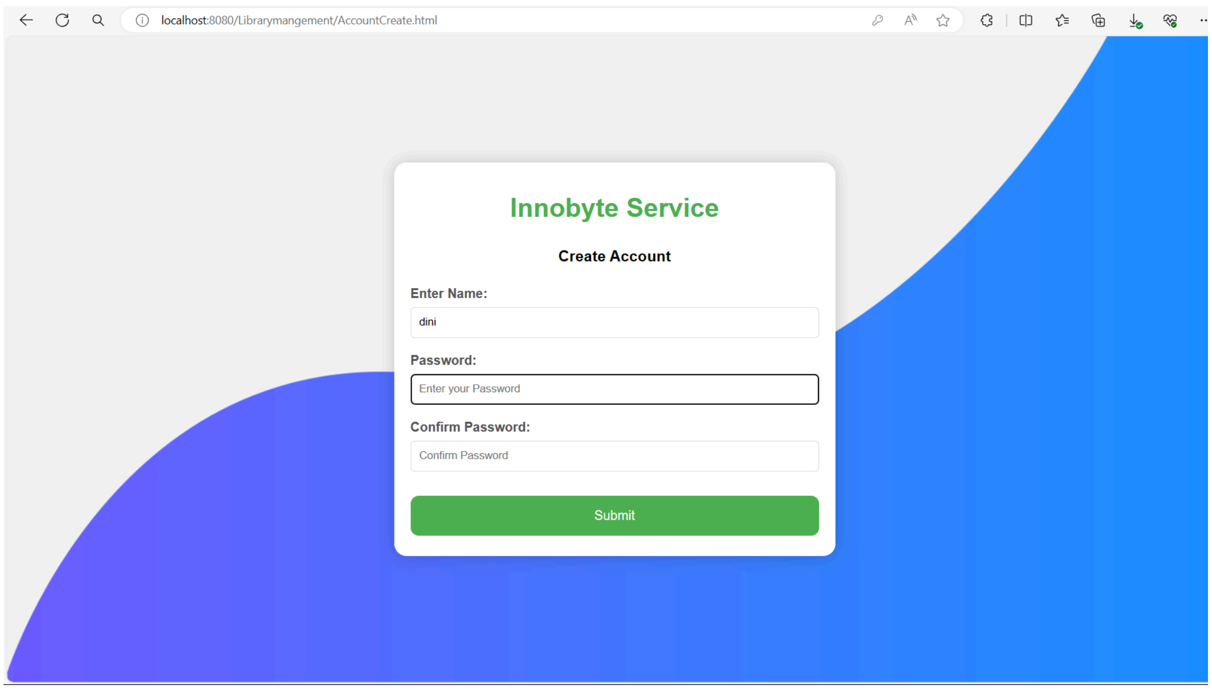
8. Exception Handling: The servlet catches any exceptions that occur during the database connection or query execution and prints the stack trace for debugging purposes.

Overall, this servlet provides the backend functionality to validate user credentials against a MySQL database and handle the login process accordingly.

3.Create Account html :AccountCreate.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet"
                href="loginStyle.css">
                <style>
body {
  background-image: url('https://www.innobytess.com/assets/images/background.png');
}
</style>
</head>
<body>
<div class="main">
                <h1>Innobyte Service</h1>
                <h3>Create Account</h3>
                <form action="AccountCreate" method="get">
                <label for="Enter your Name">
                                Enter Name:
                        </label>
                        <input type="Text"
                                id="Name"
                                name="Name"
                                placeholder="Enter your Name" required>
                        <label for="password">
                                Password:
                        </label>
                        <input type="password"
                                id="password"
                                name="password"
                                placeholder="Enter your Password" required>
                                <label for="password">
```

```
                    Confirm Password:
                </label>
                <input type="password"
                    id="confirm-password"
                    name="confirm-password"
                    placeholder="Confirm Password" required>

                <div class="wrap">
                    <button type="submit">
                        Submit
                    </button>
                </div>
            </form>
        </div>
</body>
</html>
```


localhost:8080/Librarymangement/AccountCreate.html

This HTML code represents a simple registration form for creating an account on the Innobyte Service. Let's break down its components:

1. **Document Type Declaration** (`<!DOCTYPE html>`): Declares the document type and version of HTML being used.
2. HTML Structure:
   - `<html>`: **The root element of the HTML document.**
   - `<head>`: **Contains meta-information about the HTML document, such as character encoding and title.**

   - `<meta charset="UTF-8">`: **Specifies the character encoding of the document as UTF-8.**
   - `<title>`: **Sets the title of the HTML document.**
   - `<link rel="stylesheet" href="loginStyle.css">`: **Links an external stylesheet named `loginStyle.css` to style the HTML elements.**
   - **Inline `<style>`: Defines additional CSS rules for styling the HTML elements, setting a background image for the body.**
3. **Body Content:**
   - `<body>`: Contains the visible content of the HTML document.
   - `<div class="main">`: Defines a main container for the content.
   - `<h1>` **and** `<h3>`: Headings providing titles for the form.
   - `<form>`: Specifies a form element for user input. It has an action **attribute pointing to a server-side script (`AccountCreate`) and uses the HTTP method `GET` to submit form data.**
   - `<label>`: **Labels for the input fields.**
   - `<input>`: Text input fields for the user to enter their name, password, and confirm password. The `type` attribute specifies the input type (text or password), the `id` attribute provides a unique identifier for each input field, the `name` attribute assigns a name to each input field, and the `placeholder` attribute provides a hint or example text for the user.
   - `<button>`: **Submits the form data when clicked.**
4. **Styling:**
   - The background image is set for the body using CSS.

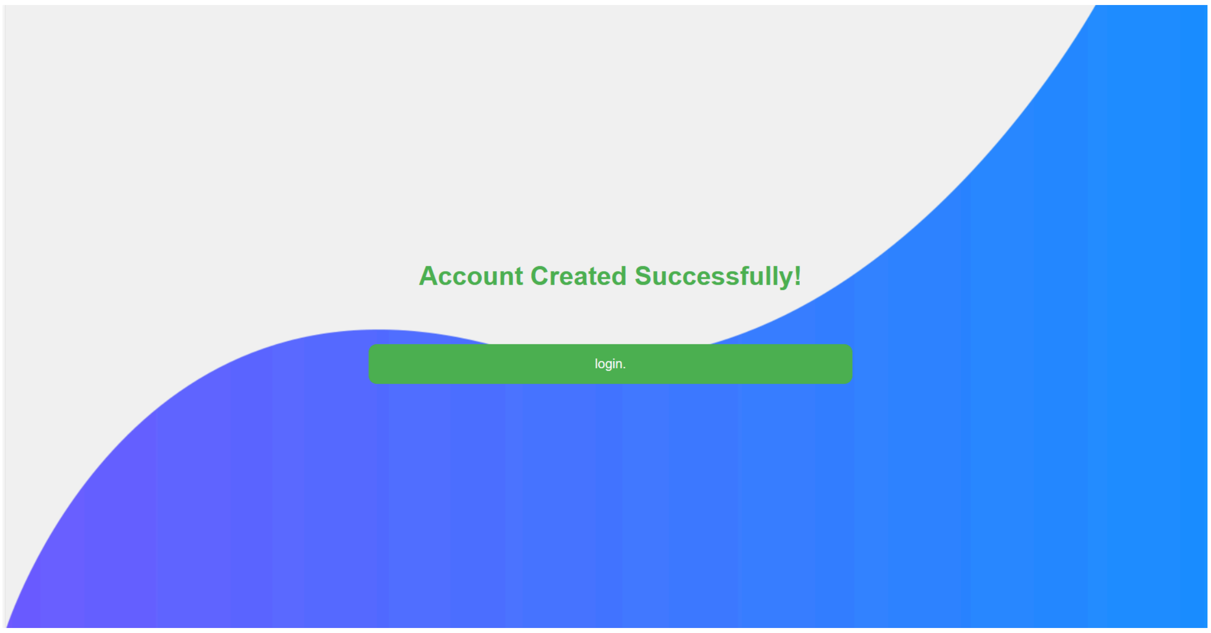
Overall, this HTML code creates a registration form for users to enter their name, password, and confirm password to create an account on the Innobyte Service.

4.Servlet for creating account:AccountCreate.java

```
package accountcreate;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

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("/AccountCreate")
public class AccountCreate extends HttpServlet {
	/**
	 *
	 */private static final long serialVersionUID = 1L;

	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
			PrintWriter out = response.getWriter();
			      String username = request.getParameter("Name");
			String password = request.getParameter("password");
			String confirmPassword = request.getParameter("confirm-password");
	boolean bool=password.equals(confirmPassword);
	System.out.println(bool);

			if (bool) {
			      System.out.println("password checked");

			      String url = "jdbc:mysql://localhost:3306/db";
			   String user = "root";
			   String dbPassword = "root";

			   try {
			       Class.forName("com.mysql.cj.jdbc.Driver");
			      Connection conn = DriverManager.getConnection(url, user,
dbPassword);
			      String insertSql = "INSERT INTO users (username, passwords)
VALUES (?, ?)";
			      PreparedStatement pstmt = conn.prepareStatement(insertSql);
			         pstmt.setString(1, username);
			         pstmt.setString(2, password);
			         int rowsInserted = pstmt.executeUpdate();

			         if (rowsInserted > 0) {
			               response.setContentType("text/html");
			               out.println("<!DOCTYPE html>");
			            out.println("<html>");
			            out.println("<head>");
			            out.println("<title>Success Page</title><style>body {\r\n"
					+ "	\r\n"
					+ "\r\n"
					+ "	display: flex;\r\n"
					+ "	align-items: center;\r\n"
```

```
                                        + "        justify-content: center;\r\n"
                                        + "        font-family: sans-serif;\r\n"
                                        + "        line-height: 1.5;\r\n"
                                        + "        min-height: 100vh;\r\n"
                                        + "        background: #f3f3f3;\r\n"
                                        + "        flex-direction: column;\r\n"
                                        + "        margin: 0;\r\n"
                                        + "}button {\r\n"
                                        + "        padding: 15px;\r\n"
                                        + "        border-radius: 10px;\r\n"
                                        + "        margin-top: 15px;\r\n"
                                        + "        margin-bottom: 15px;\r\n"
                                        + "        border: none;\r\n"
                                        + "        color: white;\r\n"
                                        + "        cursor: pointer;\r\n"
                                        + "        background-color: #4CAF50;\r\n"
                                        + "        width: 40%;\r\n"
                                        + "        font-size: 16px;\r\n"
                                        + "}h1 {\r\n"
                                        + "        color: #4CAF50;\r\n"
                                        + "}"
                                        + "body {\r\n"
                                        + "  background-image:
url('https://www.innobytess.com/assets/images/background.png');\r\n"
                                        + "}\r\n"
                                        + "</style>");
                              out.println("</head>");
                              out.println("<body>");
                              out.println("<h1>Account Created Successfully!</h1><br>");
                              out.println("<button id=\"redirectButton\">login.</button>");
                              out.print("<script>\r\n"
                                        + "        "
                                        + "        var button =
document.getElementById(\"redirectButton\");\r\n"
                                        + "\r\n"
                                        + "        "
                                        + "        button.addEventListener(\"click\", function()
{\r\n"
                                        + "            // Redirect to the login page\r\n"
                                        + "            window.location.href = \"index.jsp\";\r\n"
                                        + "        });\r\n"
                                        + "    </script>");
                              out.println("</body>");
                              out.println("</html>");
                         }else {
                              response.sendRedirect("AccountCreate.html");
                         }
```

```
                } catch (SQLException e) {

                                e.printStackTrace();
                        } catch (ClassNotFoundException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
              }else {
                   System.out.println("password unchecked");

              response.sendRedirect("AccountCreate.html");
     }
     }
```



This Java servlet named `AccountCreate` handles the creation of user accounts based on the submitted registration form data. Let's go through its key components:

1. Servlet Configuration: The servlet is annotated with `@WebServlet("/AccountCreate")`, specifying its URL mapping.
2. doGet Method: This method is invoked when an HTTP GET request is sent to the servlet, typically when the user submits the registration form.
3. Request Handling:
   - The servlet retrieves the username, password, and confirm password parameters from the request using `request.getParameter()`.
   - It checks if the password matches the confirm password using `password.equals(confirmPassword)`.
4. Database Interaction:
   - If the passwords match, the servlet establishes a connection to the MySQL database using JDBC, specifying the database URL, username, and password.

  - It prepares an SQL INSERT statement to insert the user's username and password into the `users` table.
  - It executes the INSERT statement using a `PreparedStatement` and checks if any rows were inserted successfully.
5. Response Handling:
  - If the account creation is successful, it generates an HTML response using a `PrintWriter`, indicating success and providing a button to redirect the user to the login page (`index.jsp`).
  - If the account creation fails, it redirects the user back to the account creation page (`AccountCreate.html`).
6. Exception Handling:
  - It catches any SQL exceptions that occur during the database connection or query execution and prints the stack trace for debugging purposes.
7. HTML Response: The HTML response generated upon successful account creation includes styling and a JavaScript function to redirect the user to the login page upon button click.

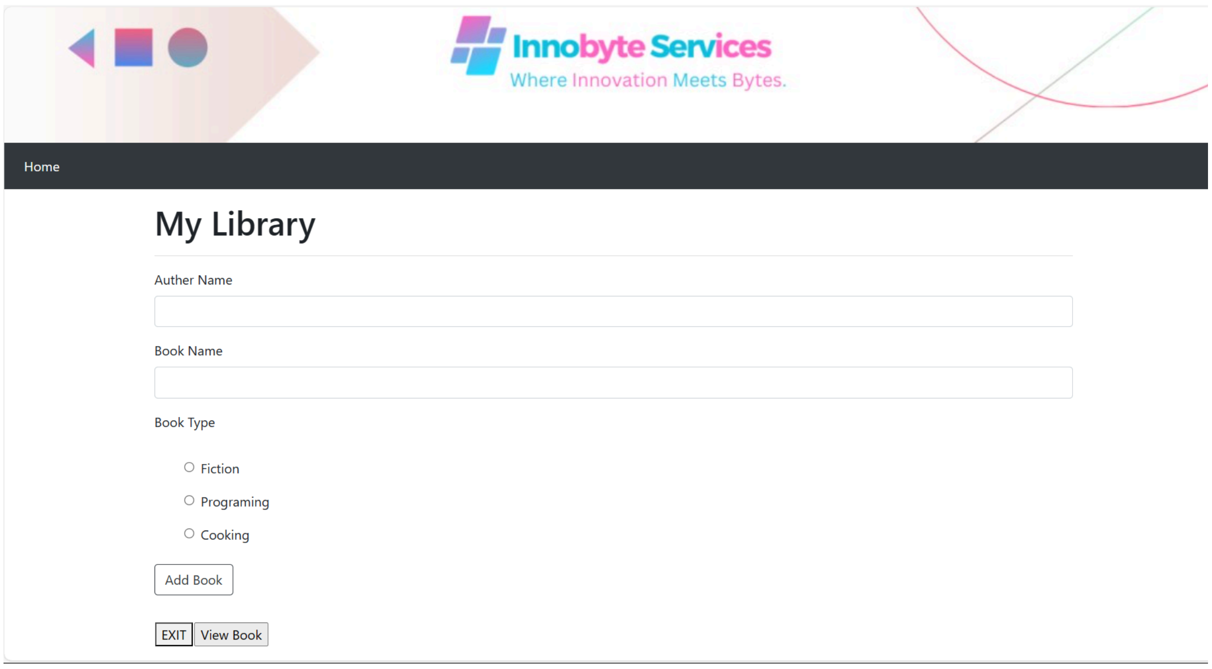
Overall, this servlet provides the backend functionality to create user accounts, validate passwords, interact with a MySQL database, and generate appropriate responses based on the outcome of the account creation process.

5.My Library management:Library.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-fit=no"
    />
    <link
      rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css
"

integrity="sha384-zCbKRCUGaJDkqS1kPbPd7TveP5iyJE0EjAuZQTgFLD2ylzuqKfdKlfG/eSr
txUkn"
      crossorigin="anonymous"
    />
    <title>Innobyte Library Management System</title>
  </head>

  <body>
        <img
src="https://media.licdn.com/dms/image/D563DAQEufhsuf3E_NA/image-scale_191_11
28/0/1708109140475/innobyte_services_cover?e=2147483647&v=beta&t=dZzNUf3waCkf
```

```
m_Qc9TT-z2X-0nFO9SeYQJScUlCtf_w" alt="InnoByte Solution" width="100%"
height="100%">
     <br>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <button
        class="navbar-toggler"
        type="button"
        data-toggle="collapse"
        data-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent"
        aria-expanded="false"
        aria-label="Toggle navigation"
      >
        <span class="navbar-toggler-icon"></span>
      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
          <li class="nav-item active">
            <a class="nav-link" href="#"
              >Home <span class="sr-only">(current)</span></a
            >
          </li>

        </ul>
      </div>
    </nav>
    <div id="alertuser"></div>

    <div class="container my-3">
      <h1>My Library</h1>
      <hr />
      <form action="AddBookServlet" method="post">
        <div class="form-group">
          <label for="exampleInputEmail1">Auther Name</label>
          <input
            type="text"
            class="form-control"
            name="Auther_Name"
            aria-describedby="emailHelp"
          />

        </div>
        <div class="form-group">
          <label for="exampleInputPassword1">Book Name</label>
          <input type="text" class="form-control" name="Book-Name" />
        </div>
        <div class="form-group">
          <label for="bookType">Book Type</label>
          <div class="check-boxes my-3 mx-5">
            <div class="form-check p-2">
              <input
                class="form-check-input"
                type="radio"
                name="check-box"
                id="Fiction"
                value="Fiction"
              />
              <label class="form-check-label" for="Fiction"> Fiction </label>
            </div>
            <div class="form-check p-2">
              <input
                class="form-check-input"
                type="radio"
                name="check-box"
                id="Programing"
                value="Programing"
              />
              <label class="form-check-label" for="Programing">
                Programing
              </label>
            </div>
            <div class="form-check p-2">
              <input
                class="form-check-input"
                type="radio"
                name="check-box"
                id="Cooking"
                value="Cooking"
              />
              <label class="form-check-label" for="Cooking"> Cooking </label>
            </div>
          </div>
        </div>

        <button type="submit" class="btn btn-outline-dark">Add Book</button>
      </form>
      <table class="table table-dark my-3">

    </div>
    <table>
            <tr>
                  <td>
                        <button
onclick="redirectToLoginPage()">EXIT</button>
                <script>
                     function redirectToLoginPage() {

                    window.location.href = "index.jsp";
                    }
                    </script>
                  </td>
                  <td>
                        <form action="ViewLibrary" method="get">
                <input type="submit" value="View Book">
                </form>
```

```
                    </td>

                </tr>
            </table>


        </body>
    </html>
```



My Library

Auther Name

Book Name

Book Type

Fiction

Programing

Cooking

This HTML code creates a web page for managing a library system. Here's a breakdown of its components:

1. Document Type Declaration (`<!DOCTYPE html>`): Specifies the document type and version of HTML being used.
2. HTML Structure:
   - `<html>`: The root element of the HTML document.
   - `<head>`: Contains meta-information about the HTML document, such as character encoding and title.
   - `<meta charset="utf-8">`: Specifies the character encoding of the document as UTF-8.
   - `<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">`: Defines the viewport properties for responsive design.
   - `<link rel="stylesheet" href="bootstrap.min.css">`: Links the Bootstrap CSS framework for styling the webpage.
   - `<title>`: Sets the title of the HTML document.
3. Body Content:
   - `<body>`: Contains the visible content of the HTML document.
   - `<img>`: Displays an image of the InnoByte Solution logo.

   - `<nav>`: Defines a navigation bar using Bootstrap's navbar component.
   - `<form>`: Allows users to add a new book to the library. It contains input fields for the author's name, book name, and radio buttons for selecting the book type (Fiction, Programming, Cooking).
   - `<button>`: Submits the form data to add a new book.
   - `<table>`: Displays the list of books in the library.
   - Two buttons placed in a separate `<table>`:
     - "EXIT": Redirects the user to the login page (`index.jsp`) when clicked.
     - "View Book": Submits the form data to view the library.
4. Scripting:
   - JavaScript function `redirectToLoginPage()` redirects the user to the login page (`index.jsp`) when the "EXIT" button is clicked.
5. Styling:
   - Bootstrap classes are used for styling components like the navigation bar, form inputs, buttons, etc.

Overall, this HTML code creates a user-friendly interface for managing a library system, allowing users to add new books and view the existing library collection.

6.View Library Servlet :ViewLibrary.java

```
package viewLirary;



import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/**

 * Servlet implementation class ViewLirary

 */

@WebServlet("/ViewLibrary")

public class ViewLibrary extends HttpServlet {

        private static final long serialVersionUID = 1L;




   public ViewLibrary() {

       super();

   }

        protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {



                response.getWriter().append("Served at:
").append(request.getContextPath());
```

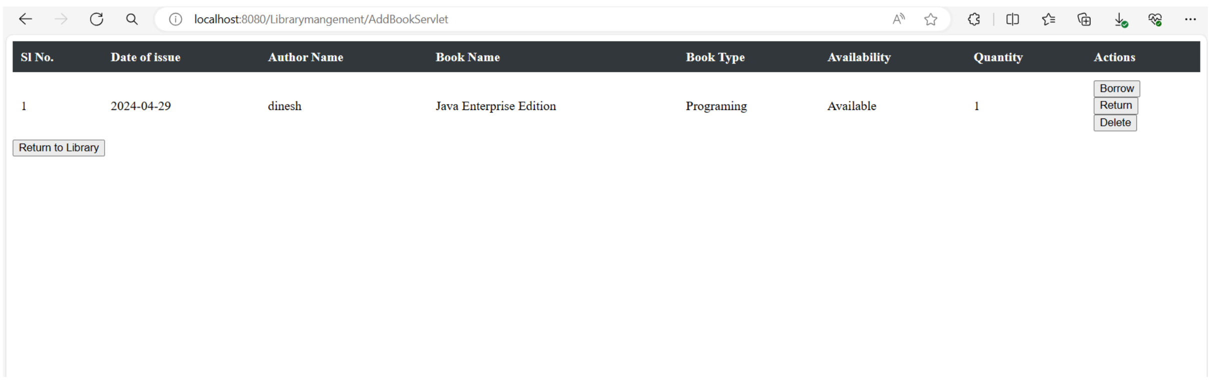
```
        response.setContentType("text/html");

        PrintWriter out = response.getWriter();

         String url = "jdbc:mysql://localhost:3306/db";

    String user = "root";

    String password = "root";



    try {

        // Load MySQL JDBC Driver

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



        // Create Connection

        Connection conn = DriverManager.getConnection(url, user, password);

        String selectSql = "SELECT * FROM Books";

        try (PreparedStatement selectPstmt = conn.prepareStatement(selectSql)) {

            ResultSet rs = selectPstmt.executeQuery();



            out.println("<html><head><link rel=\"stylesheet\"
href=\"ViewLibrary.css\"></head><body>");

            out.println("<table class='table table-dark my-3'>");
```

```
                    out.println("<thead><tr><th scope='col'>Sl No.</th><th scope='col'>Date of
issue</th><th scope='col'>Author Name</th><th scope='col'>Book Name</th><th
scope='col'>Book Type</th><th scope='col'>Availability</th><th
scope='col'>Quantity</th><th scope='col'>Actions</th></tr></thead>");

                    out.println("<tbody id='table-body'>");

                    int slNo = 1;

                    while (rs.next()) {

                        out.println("<tr>");

                        out.println("<td>" + slNo++ + "</td>");

                        out.println("<td>" + rs.getDate("Submission_Date") + "</td>");

                        out.println("<td>" + rs.getString("Auther_Name") + "</td>");

                        out.println("<td>" + rs.getString("Book_Name") + "</td>");

                        out.println("<td>" + rs.getString("Book_Type") + "</td>");

                        out.println("<td>" + (rs.getBoolean("Availability") ? "Available" : "Not
Available") + "</td>");

                        out.println("<td>"+rs.getInt("quantity")+"</td>");

                        out.println("<td><button onclick=\"borrowBook(" + rs.getInt("id") +
")\">Borrow</button><br>");

                        out.println("<button onclick=\"returnBook(" + rs.getInt("id") +
")\">Return</button><br>");

                        out.println("<button onclick=\"deleteBook(" + rs.getInt("id") +
")\">Delete</button></td>");
```

```
                out.println("</tr>");
            }
            out.println("</tbody></table><button id=\"redirectButton\">Return to
Library</button>");
            out.println("<script>"
                    + "        var button =
document.getElementById(\"redirectButton\");\r\n"
                    + "        button.addEventListener(\"click\", function() {\r\n"
                    + "            window.location.href = \"Library.html\";\r\n"
                    + "        });\r\n");
            out.println("function borrowBook(bookId) {");
            out.println("    var xhr = new XMLHttpRequest();");
            out.println("    xhr.open('POST', 'BorrowBookServlet?bookId=' + bookId,
true);");
            out.println("    xhr.send();");
            out.println("    xhr.onload = function() {");
            out.println("      if (xhr.status == 200) {");
            out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
            out.println("      } else {");
            out.println("        alert('An error occurred. Please try again later.');");
            out.println("      }");
            out.println("    };");
            out.println("}");
            out.println("function returnBook(bookId) {");
            out.println("    var xhr = new XMLHttpRequest();");
            out.println("    xhr.open('POST', 'ReturnBookServlet?bookId=' + bookId,
true);");
            out.println("    xhr.send();");
            out.println("    xhr.onload = function() {");
            out.println("      if (xhr.status == 200) {");
            out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
            out.println("      } else {");
            out.println("        alert('An error occurred. Please try again later.');");
            out.println("      }");
            out.println("    };");
            out.println("}");
            out.println("function deleteBook(bookId) {");
            out.println("    var xhr = new XMLHttpRequest();");
```

```
                out.println("    xhr.open('POST', 'DeleteBookServlet?bookId=' + bookId,
true);");

                out.println("    xhr.send();");

                out.println("    xhr.onload = function() {");

                out.println("      if (xhr.status == 200) {");

                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");

                out.println("      } else {");

                out.println("        alert('An error occurred. Please try again later.');");

                out.println("      }");

                out.println("    };");

                out.println("}");

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

                out.println("</body></html>");

            }



            conn.close();

        } catch (Exception e) {

            out.println("<script>alert('An error occurred. Please try again
later.');</script>");
```

```
                e.printStackTrace();

            }

        }



}
```



Sl No.	Date of issue	Author Name	Book Name	Book Type	Availability	Quantity	Actions
1	2024-04-29	dinesh	Java Enterprise Edition	Programing	Available	1	Borrow Return Delete

Return to Library

The `ViewLibrary` servlet is responsible for displaying the list of books in the library system. Here's an explanation of its functionality:

1. Servlet Configuration: The servlet is annotated with `@WebServlet("/ViewLibrary")`, specifying its URL mapping.
2. doGet Method: This method is invoked when an HTTP GET request is sent to the servlet. It retrieves and displays the list of books in the library.
3. Setting Response Content Type: It sets the content type of the response to "text/html" to ensure that the response is interpreted as HTML by the browser.
4. Database Connection: The servlet establishes a connection to the MySQL database using JDBC, specifying the database URL, username, and password.
5. Retrieving Book Data: It executes a SELECT query to retrieve all book records from the database table named "Books".
6. Dynamic HTML Generation: Upon successfully retrieving the book data, the servlet dynamically generates an HTML table to display the list of books.

    - Each row of the table corresponds to a book record retrieved from the database.
    - The table columns include information such as submission date, author name, book name, book type, availability, and quantity.
    - JavaScript functions are embedded within the HTML response to handle asynchronous requests for borrowing, returning, and deleting books.
    - These functions utilize XMLHttpRequests to communicate with the respective servlets (`BorrowBookServlet`, `ReturnBookServlet`, `DeleteBookServlet`) for updating the library view dynamically without refreshing the page.

7. Exception Handling: If any exception occurs during database interaction or HTML generation, an alert is displayed to the user, and the exception is printed for debugging purposes.

Overall, this servlet provides the frontend functionality to view the list of books in the library system and interact with the library through borrowing, returning, and deleting books asynchronously.

7.Add Book Servlet:AddBookServlet.java

```
package addBookServlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDate;

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("/AddBookServlet")
public class AddBookServlet extends HttpServlet {
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      // Retrieve form data
      String authorName = request.getParameter("Auther_Name");
      String bookName = request.getParameter("Book-Name");
      String bookType = request.getParameter("check-box");

      // Get current date
      LocalDate currentDate = LocalDate.now();

      // Database connection parameters
      String url = "jdbc:mysql://localhost:3306/db";
      String user = "root";
      String password = "root";

      try (Connection conn = DriverManager.getConnection(url, user, password)) {
         // Check if the book already exists
         String checkIfExistsQuery = "SELECT id, quantity FROM Books WHERE
Auther_Name = ? AND Book_Name = ? AND Book_Type = ?";
         try (PreparedStatement pstmt = conn.prepareStatement(checkIfExistsQuery)) {
            pstmt.setString(1, authorName);
            pstmt.setString(2, bookName);
            pstmt.setString(3, bookType);
            ResultSet rs1 = pstmt.executeQuery();

            if (rs1.next()) {
               // Book already exists, update quantity
               int bookId = rs1.getInt("id");
               int currentQuantity = rs1.getInt("quantity");

               String updateQuantityQuery = "UPDATE Books SET quantity = ? WHERE id =
?";
               try (PreparedStatement updateStmt =
conn.prepareStatement(updateQuantityQuery)) {
                  updateStmt.setInt(1, currentQuantity + 1);
                  updateStmt.setInt(2, bookId);
                  updateStmt.executeUpdate();
               }
            } else {
               // Insert the record with the current date
```

```
                    String insertQuery = "INSERT INTO Books (Auther_Name, Book_Name,
Book_Type, Submission_Date, quantity) VALUES (?, ?, ?, ?, ?)";
                    try (PreparedStatement insertStmt = conn.prepareStatement(insertQuery)) {
                        insertStmt.setString(1, authorName);
                        insertStmt.setString(2, bookName);
                        insertStmt.setString(3, bookType);
                        insertStmt.setDate(4, Date.valueOf(currentDate));
                        insertStmt.setInt(5, 1);
                        insertStmt.executeUpdate();
                    }
                }
            }

            // Prepare SQL statement to retrieve all books
            String selectSql = "SELECT * FROM Books";
            try (PreparedStatement selectPstmt = conn.prepareStatement(selectSql)) {
                ResultSet rs = selectPstmt.executeQuery();

                out.println("<html><head><link rel=\"stylesheet\"
href=\"ViewLibrary.css\"></head><body>");
                out.println("<table class='table table-dark my-3'>");
                out.println("<thead><tr><th scope='col'>Sl No.</th><th scope='col'>Date of
issue</th><th scope='col'>Author Name</th><th scope='col'>Book Name</th><th
scope='col'>Book Type</th><th scope='col'>Availability</th><th
scope='col'>Quantity</th><th scope='col'>Actions</th></tr></thead>");
                out.println("<tbody id='table-body'>");
                int slNo = 1;
                while (rs.next()) {
                    out.println("<tr>");
                    out.println("<td>" + slNo++ + "</td>");
                    out.println("<td>" + rs.getDate("Submission_Date") + "</td>");
                    out.println("<td>" + rs.getString("Auther_Name") + "</td>");
                    out.println("<td>" + rs.getString("Book_Name") + "</td>");
                    out.println("<td>" + rs.getString("Book_Type") + "</td>");
                    out.println("<td>" + (rs.getBoolean("Availability") ? "Available" : "Not Available")
+ "</td>");
                    out.println("<td>"+rs.getInt("quantity")+"</td>");
                    out.println("<td><button onclick=\"borrowBook(" + rs.getInt("id") +
")\">Borrow</button><br>");
                    out.println("<button onclick=\"returnBook(" + rs.getInt("id") +
")\">Return</button><br>");
                    out.println("<button onclick=\"deleteBook(" + rs.getInt("id") +
")\">Delete</button></td>");
                    out.println("</tr>");
                }
                out.println("</tbody></table><button id=\"redirectButton\">Return to
Library</button>");
                out.println("<script>"
                            + "        var button = document.getElementById(\"redirectButton\");\r\n"
                            + "        button.addEventListener(\"click\", function() {\r\n"
                            + "            window.location.href = \"Library.html\";\r\n"
                            + "        });\r\n");
                out.println("function borrowBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'BorrowBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("function returnBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'ReturnBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("function deleteBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'DeleteBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("</script>");
                out.println("</body></html>");
            }
          } catch (SQLException e) {
            out.println("<script>alert('An error occurred. Please try again later.');</script>");
```

```
            e.printStackTrace();
        }
    }
}
```

This Java servlet named `AddBookServlet` handles the addition of new books to the library system. Here's an explanation of its functionality:

1. Servlet Configuration: The servlet is annotated with `@WebServlet("/AddBookServlet")`, specifying its URL mapping.
2. doPost Method: This method is invoked when an HTTP POST request is sent to the servlet, typically when the user submits the form to add a new book.
3. Response Configuration:
    - The content type of the response is set to "text/html".
    - A PrintWriter object is initialized to write HTML content to the response.
4. Retrieving Form Data: The servlet retrieves the author's name, book name, and book type from the request parameters.
5. Database Connection: It establishes a connection to the MySQL database using JDBC, specifying the database URL, username, and password.
6. Checking Book Existence:
    - The servlet checks if the book already exists in the database by executing a SELECT query.
    - If the book already exists, it updates the quantity of the existing record.
    - If the book is not found, it inserts a new record into the database with the current date and quantity set to 1.
7. HTML Response:
    - Upon successful addition of the book, the servlet dynamically generates an HTML table displaying the list of books in the library.
    - The table includes options to borrow, return, or delete each book.
    - JavaScript functions are embedded in the HTML response to handle asynchronous requests for borrowing, returning, and deleting books.
    - An event listener is added to a button to return to the library page (`Library.html`).
8. Exception Handling:
    - If any SQLException occurs during database interaction, an alert is displayed to the user, and the exception is printed for debugging.

Overall, this servlet provides the backend functionality to add new books to the library system and dynamically update the library view without refreshing the page.

<u>8.Book Barrow Servlet:BarrowBookServlet.java</u>

```
package addBookServlet
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

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("/BorrowBookServlet")
public class BorrowBookServlet extends HttpServlet {
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        int bookId = Integer.parseInt(request.getParameter("bookId"));
        PrintWriter out = response.getWriter();
        // Database connection parameters
        String url = "jdbc:mysql://localhost:3306/db";
        String user = "root";
        String password = "root";

        try (Connection conn = DriverManager.getConnection(url, user, password)) {
            String updateQuantityQuery = "UPDATE Books SET quantity = quantity - 1 WHERE
id = ?";
            try (PreparedStatement updateStmt =
conn.prepareStatement(updateQuantityQuery)) {
                updateStmt.setInt(1, bookId);
                updateStmt.executeUpdate();
            }

            String selectSql = "SELECT * FROM Books";
            try (PreparedStatement selectPstmt = conn.prepareStatement(selectSql)) {
                ResultSet rs = selectPstmt.executeQuery();

                // Output table rows dynamically
                out.println("<html><head><link rel=\"stylesheet\"
href=\"ViewLibrary.css\"></head><body>");
                out.println("<table class='table table-dark my-3'>");
                out.println("<tbody id='table-body'>");
                int slNo = 1;
                while (rs.next()) {
                    out.println("<tr>");
                    out.println("<td>" + slNo++ + "</td>");
                    out.println("<td>" + rs.getDate("Submission_Date") + "</td>");
                    out.println("<td>" + rs.getString("Auther_Name") + "</td>");
                    out.println("<td>" + rs.getString("Book_Name") + "</td>");
                    out.println("<td>" + rs.getString("Book_Type") + "</td>");
                    out.println("<td>" + (rs.getBoolean("Availability") ? "Available" : "Not Available")
+ "</td>");
                    out.println("<td>"+rs.getInt("quantity")+"</td>");
                    out.println("<td><button onclick=\"borrowBook(" + rs.getInt("id") +
")\">Borrow</button><br>");
                    out.println("<button onclick=\"returnBook(" + rs.getInt("id") +
")\">Return</button><br>");
                    out.println("<button onclick=\"deleteBook(" + rs.getInt("id") +
")\">Delete</button></td>");
                    out.println("</tr>");
                }
                out.println("<script>");
                out.println("function borrowBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'BorrowBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("function returnBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'ReturnBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("function deleteBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'DeleteBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
```

```
            out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
            out.println("      } else {");
            out.println("        alert('An error occurred. Please try again later.');");
            out.println("      }");
            out.println("    };");
            out.println("}");
            out.println("</script>");
            out.println("</body></html>");
         }
      } catch (SQLException e) {
         out.println("<script>alert('An error occurred. Please try again later.');</script>");
         e.printStackTrace();
      }

   }
}
```

This Java servlet named `BorrowBookServlet` handles the borrowing of books from the library system. Here's an explanation of its functionality:

1. Servlet Configuration: The servlet is annotated with `@WebServlet("/BorrowBookServlet")`, specifying its URL mapping.
2. doPost Method: This method is invoked when an HTTP POST request is sent to the servlet, typically when the user clicks the "Borrow" button for a specific book.
3. Retrieving Book ID: The servlet retrieves the book ID from the request parameter.
4. Database Connection: It establishes a connection to the MySQL database using JDBC, specifying the database URL, username, and password.
5. Updating Book Quantity: It updates the quantity of the book in the database by executing an UPDATE query to decrement the quantity by 1.
6. Dynamic HTML Response: Upon successfully borrowing the book, the servlet dynamically generates an HTML table displaying the updated list of books in the library.
   - The table rows are dynamically generated based on the data retrieved from the database.
   - JavaScript functions are embedded in the HTML response to handle asynchronous requests for borrowing, returning, and deleting books.
   - These JavaScript functions send XMLHttpRequests to the respective servlets (`BorrowBookServlet`, `ReturnBookServlet`, `DeleteBookServlet`) to update the library view dynamically without refreshing the page.
7. Exception Handling: If any SQLException occurs during database interaction, an alert is displayed to the user, and the exception is printed for debugging.

Overall, this servlet provides the backend functionality to borrow books from the library system and dynamically update the library view without refreshing the page.

9.Return Book Servlet:ReturnBookServlet.java

```
package addBookServlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

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("/ReturnBookServlet")
public class ReturnBookServlet extends HttpServlet {
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        int bookId = Integer.parseInt(request.getParameter("bookId"));
        String name = request.getParameter("name");
        String password = request.getParameter("password");

        // Check user authentication here

        // Database connection parameters
        String url = "jdbc:mysql://localhost:3306/db";
        String user = "root";
        String dbPassword = "root";

        try (Connection conn = DriverManager.getConnection(url, user, dbPassword)) {
            String updateQuantityQuery = "UPDATE Books SET quantity = quantity + 1 WHERE
id = ?";
            try (PreparedStatement updateStmt =
conn.prepareStatement(updateQuantityQuery)) {
                updateStmt.setInt(1, bookId);
                updateStmt.executeUpdate();
            }
            String selectSql = "SELECT * FROM Books";
            try (PreparedStatement selectPstmt = conn.prepareStatement(selectSql)) {
                ResultSet rs = selectPstmt.executeQuery();

                // Output table rows dynamically
                out.println("<html><head><link rel=\"stylesheet\"
href=\"ViewLibrary.css\"></head><body>");
                out.println("<table class='table table-dark my-3'>");
                out.println("<tbody id='table-body'>");
                int slNo = 1;
                while (rs.next()) {
                    out.println("<tr>");
                    out.println("<td>" + slNo++ + "</td>");
                    out.println("<td>" + rs.getDate("Submission_Date") + "</td>");
                    out.println("<td>" + rs.getString("Auther_Name") + "</td>");
                    out.println("<td>" + rs.getString("Book_Name") + "</td>");
                    out.println("<td>" + rs.getString("Book_Type") + "</td>");
                    out.println("<td>" + (rs.getBoolean("Availability") ? "Available" : "Not Available")
+ "</td>");
                    out.println("<td>"+rs.getInt("quantity")+"</td>");
                    out.println("<td><button onclick=\"borrowBook(" + rs.getInt("id") +
")\">Borrow</button><br>");
                    out.println("<button onclick=\"returnBook(" + rs.getInt("id") +
")\">Return</button><br>");
                    out.println("<button onclick=\"deleteBook(" + rs.getInt("id") +
")\">Delete</button></td>");
                    out.println("</tr>");
                }

                out.println("<script>");
                out.println("function borrowBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'BorrowBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("function returnBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'ReturnBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("function deleteBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'DeleteBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("</script>");
                out.println("</body></html>");
              }
            } catch (SQLException e) {
              out.println("<script>alert('An error occurred. Please try again later.');</script>");
              e.printStackTrace();
            }

        }
}
```

This Java servlet named `ReturnBookServlet` handles the returning of books to the library system. Here's an explanation of its functionality:

1. Servlet Configuration: The servlet is annotated with `@WebServlet("/ReturnBookServlet")`, specifying its URL mapping.
2. doPost Method: This method is invoked when an HTTP POST request is sent to the servlet, typically when the user clicks the "Return" button for a specific book.
3. Retrieving Book ID: The servlet retrieves the book ID from the request parameter.
4. Database Connection: It establishes a connection to the MySQL database using JDBC, specifying the database URL, username, and password.
5. Updating Book Quantity: It updates the quantity of the book in the database by executing an UPDATE query to increment the quantity by 1.
6. Dynamic HTML Response: Upon successfully returning the book, the servlet dynamically generates an HTML table displaying the updated list of books in the library.

- The table rows are dynamically generated based on the data retrieved from the database.
- JavaScript functions are embedded in the HTML response to handle asynchronous requests for borrowing, returning, and deleting books.
- These JavaScript functions send XMLHttpRequests to the respective servlets (`BorrowBookServlet`, `ReturnBookServlet`, `DeleteBookServlet`) to update the library view dynamically without refreshing the page.

7. Exception Handling: If any SQLException occurs during database interaction, an alert is displayed to the user, and the exception is printed for debugging.

Overall, this servlet provides the backend functionality to return books to the library system and dynamically update the library view without refreshing the page.

10.Delete Book servlet:DeleteBookServlet.java

```
package addBookServlet;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

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("/DeleteBookServlet")
public class DeleteBookServlet extends HttpServlet {
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        int bookId = Integer.parseInt(request.getParameter("bookId"));

        // Database connection parameters
        String url = "jdbc:mysql://localhost:3306/db";
        String user = "root";
        String password = "root";
        try (Connection conn = DriverManager.getConnection(url, user, password)) {
            String deleteQuery = "DELETE FROM Books WHERE id = ?";
            try (PreparedStatement deleteStmt = conn.prepareStatement(deleteQuery)) {
                deleteStmt.setInt(1, bookId);
                deleteStmt.executeUpdate();
            }String selectSql = "SELECT * FROM Books";
            try (PreparedStatement selectPstmt = conn.prepareStatement(selectSql)) {
                ResultSet rs = selectPstmt.executeQuery();

                // Output table rows dynamically
                out.println("<html><head><link rel=\"stylesheet\"
href=\"ViewLibrary.css\"></head><body>");
                out.println("<table class='table table-dark my-3'>");
                out.println("<tbody id='table-body'>");
                int slNo = 1;
                while (rs.next()) {
                    out.println("<tr>");
                    out.println("<td>" + slNo++ + "</td>");
                    out.println("<td>" + rs.getDate("Submission_Date") + "</td>");
                    out.println("<td>" + rs.getString("Auther_Name") + "</td>");
                    out.println("<td>" + rs.getString("Book_Name") + "</td>");
                    out.println("<td>" + rs.getString("Book_Type") + "</td>");
                    out.println("<td>" + (rs.getBoolean("Availability") ? "Available" : "Not Available")
+ "</td>");
                    out.println("<td>"+rs.getInt("quantity")+"</td>");
                    out.println("<td><button onclick=\"borrowBook(" + rs.getInt("id") +
")\">Borrow</button><br>");
                    out.println("<button onclick=\"returnBook(" + rs.getInt("id") +
")\">Return</button><br>");
                    out.println("<button onclick=\"deleteBook(" + rs.getInt("id") +
")\">Delete</button></td>");
                    out.println("</tr>");
                }
                out.println("<script>");
                out.println("function borrowBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'BorrowBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("function returnBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'ReturnBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("function deleteBook(bookId) {");
                out.println("    var xhr = new XMLHttpRequest();");
                out.println("    xhr.open('POST', 'DeleteBookServlet?bookId=' + bookId, true);");
                out.println("    xhr.send();");
                out.println("    xhr.onload = function() {");
                out.println("      if (xhr.status == 200) {");
                out.println("        document.getElementById('table-body').innerHTML =
xhr.responseText;");
                out.println("      } else {");
                out.println("        alert('An error occurred. Please try again later.');");
                out.println("      }");
                out.println("    };");
                out.println("}");
                out.println("</script>");
                out.println("</body></html>");
            }
          } catch (SQLException e) {
            out.println("<script>alert('An error occurred. Please try again later.');</script>");
            e.printStackTrace();
          }
        }
}
```

This Java servlet named `DeleteBookServlet` handles the deletion of books from the library system. Here's an explanation of its functionality:

1. Servlet Configuration: The servlet is annotated with `@WebServlet("/DeleteBookServlet")`, specifying its URL mapping.
2. doPost Method: This method is invoked when an HTTP POST request is sent to the servlet, typically when the user clicks the "Delete" button for a specific book.
3. Retrieving Book ID: The servlet retrieves the book ID from the request parameter.

4. Database Connection: It establishes a connection to the MySQL database using JDBC, specifying the database URL, username, and password.
5. Deleting Book Record: It executes a DELETE query to remove the book record from the database based on the provided book ID.
6. Dynamic HTML Response: Upon successfully deleting the book, the servlet dynamically generates an HTML table displaying the updated list of books in the library.
   - The table rows are dynamically generated based on the data retrieved from the database.
   - JavaScript functions are embedded in the HTML response to handle asynchronous requests for borrowing, returning, and deleting books.
   - These JavaScript functions send XMLHttpRequests to the respective servlets (`BorrowBookServlet`, `ReturnBookServlet`, `DeleteBookServlet`) to update the library view dynamically without refreshing the page.
7. Exception Handling: If any SQLException occurs during database interaction, an alert is displayed to the user, and the exception is printed for debugging.

Overall, this servlet provides the backend functionality to delete books from the library system and dynamically update the library view without refreshing the page.

11.pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>Librarymangement</groupId>
  <artifactId>Librarymangement</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <build>
    <plugins>
```

```
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <release>20</release>
        </configuration>
      </plugin>
      <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.2.3</version>
      </plugin>
    </plugins>

  </build>


  <dependencies>
          <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.27</version>
</dependency>
          <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
</dependency>
  </dependencies>
</project>
```

This is a Maven project configuration file (`pom.xml`) for a web application named "Librarymanagement" with packaging type "war" (Web ARchive). Let's break down its key elements:

1. Project Information:
    - `modelVersion`: Specifies the version of the POM model. In this case, it's set to 4.0.0, which is the latest version.
    - `groupId`: Identifies the group or organization that the project belongs to. Here, it's set to "Librarymanagement".
    - `artifactId`: Specifies the unique identifier for the project. In this case, it's also "Librarymanagement".
    - `version`: Indicates the version of the project. It's set to 0.0.1-SNAPSHOT, which is a common convention for an initial development version.
2. Packaging Type:

        - `packaging`: Specifies the type of packaging used for the project. Here, it's set to "war" to indicate that the project will be packaged as a Web ARchive file.
3. Build Configuration:
    - `build`: Contains configuration settings for the build process.
        - `plugins`: Specifies Maven plugins to be used during the build.
            - `maven-compiler-plugin`: Configures the Maven Compiler Plugin, which is responsible for compiling the Java source code in the project.
                - `configuration`: Specifies additional configuration for the plugin.
                    - `release`: Indicates the Java version to be used for compilation. Here, it's set to 20.
            - `maven-war-plugin`: Configures the Maven WAR Plugin, which is responsible for building the WAR file for the web application.
4. Dependencies:
    - `dependencies`: Lists the external dependencies required by the project.
        - `mysql-connector-java`: Specifies the MySQL Connector/J dependency, which is used to connect the web application to a MySQL database.
        - `javax.servlet-api`: Specifies the Java Servlet API dependency, which is needed for developing servlet-based web applications. The scope is set to "provided" because the Servlet API will be provided by the servlet container at runtime.

Overall, this `pom.xml` file defines the project structure, build configuration, and dependencies required for developing a web application for library management using Maven.

12.DatabaseSchema

```
CREATE DATABASE db;
USE db;
CREATE TABLE users(
Username VARCHAR(30),
passwords INT
);
CREATE TABLE Books (
   id INT AUTO_INCREMENT PRIMARY KEY,
   Submission_Date datetime,
   Auther_Name VARCHAR(255) NOT NULL,
   Book_Name VARCHAR(255) NOT NULL,
   Book_Type VARCHAR(100),
   quantity INT NOT NULL DEFAULT 1, -- New column for quantity
   availability BOOLEAN NOT NULL DEFAULT TRUE
);
```

The SQL script you provided creates a MySQL database named "db" and defines two tables within it: "users" and "Books".

1. Creating the Database and Switching to it:
    - `CREATE DATABASE db;`: This statement creates a new MySQL database named "db".
    - `USE db;`: This statement selects the "db" database, making it the current database for subsequent SQL statements.
2. Creating the "users" Table:
    - `CREATE TABLE users(`: This statement begins the creation of the "users" table.
    - `Username VARCHAR(30),`: Defines a column named "Username" with a maximum length of 30 characters, storing usernames.
    - `passwords INT`: Defines a column named "passwords" to store integer values representing passwords.
3. Creating the "Books" Table:
    - `CREATE TABLE Books (`: Begins the creation of the "Books" table.
    - `id INT AUTO_INCREMENT PRIMARY KEY,`: Defines a primary key column named "id" that auto-increments with each new row.
    - `Submission_Date datetime,`: Defines a column named "Submission_Date" to store datetime values representing the submission date of a book.
    - `Auther_Name VARCHAR(255) NOT NULL,`: Defines a column named "Auther_Name" to store author names with a maximum length of 255 characters. The "NOT NULL" constraint ensures that this field cannot be empty.
    - `Book_Name VARCHAR(255) NOT NULL,`: Defines a column named "Book_Name" to store book names with a maximum length of 255 characters. The "NOT NULL" constraint ensures that this field cannot be empty.
    - `Book_Type VARCHAR(100),`: Defines a column named "Book_Type" to store book types with a maximum length of 100 characters.
    - `quantity INT NOT NULL DEFAULT 1,`: Defines a column named "quantity" to store the quantity of books available. The "NOT NULL" constraint ensures that this field cannot be empty, and the "DEFAULT 1" specifies a default value of 1 if not explicitly provided.

- `availability BOOLEAN NOT NULL DEFAULT TRUE`: Defines a column named "availability" to indicate whether the book is available or not, using a boolean value. The "NOT NULL" constraint ensures that this field cannot be empty, and the "DEFAULT TRUE" specifies a default value of true if not explicitly provided.

Overall, this SQL script sets up the database structure for managing users and books, providing a foundation for a library management system.

13.Configurations Setup

1.This is an Maven java Project Build by Eclipse Enterprise Edition
2.Deploy the web application to a servlet container like Apache Tomcat Server V9.0 or above
3.Use JDBC driver jar V 8.0.27 or above.

****************************** THANK YOU **