NITTE (Deemed to be University) | NMAM INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

Laboratory Manual

JAVA AND SPRING FRAMEWORK LAB

CS3603-1

Year: 2024-25

List of Experiments

	• Ensure that all required fields (such as Employee ID, Name, Age, Department, and Email) are properly validated in employeeResult.jsp upon form submission. • Implement server-side checks to verify that: o Employee ID is a non-empty string. o Name contains only letters and spaces. o Age is a valid numeric value between 18 and 65. o Department is selected from a predefined list of options. o Email follows a valid format (e.g., name@example.com). Display clear and descriptive error messages directly on employeeResult.jsp next to each field if validation criteria are not met, guiding users to correct their input.		
3.	Develop a JSP application that integrates JDBC to perform CRUD (Create, Read, Update, and Delete) operations on a \`students\` database table. The application should: ▪ Create: Insert new student records with fields such as roll-no, name, email, and age. ▪ Read: Retrieve and display all student records from the database. ▪ Update: Modify existing student records. ▪ Delete: Remove student records from the database.	**19-25**	**2, 3**
4.	Develop a web application using Spring Boot to perform CRUD operations on Book information. The application should use Thymeleaf for the view layer, Spring MVC for the controller layer, Spring Data JPA and Hibernate for data access, and MySQL as the database. The application should manage book attributes such as title, authors, edition, publication and price.	**26-31**	**4, 5**
5.	Develop a Spring Boot application that provides RESTful APIs for performing CRUD operations on Mobile information. The application should use Spring Data JPA and Hibernate for data access to MySQL database that stores Mobile details such as model name, model number, device operating system, manufacturer, and country of origin. JSON may be considered as the return value of the API. Encapsulate error handling feature for situations such as *record not found*. Demonstrate APIs working using curl command and Postman API testing tool.	**32-36**	**4, 6**

Experiment No. 1a
Implement a Java Servlet (SimpleServlet) that handles both HTTP GET and POST requests. The servlet should:

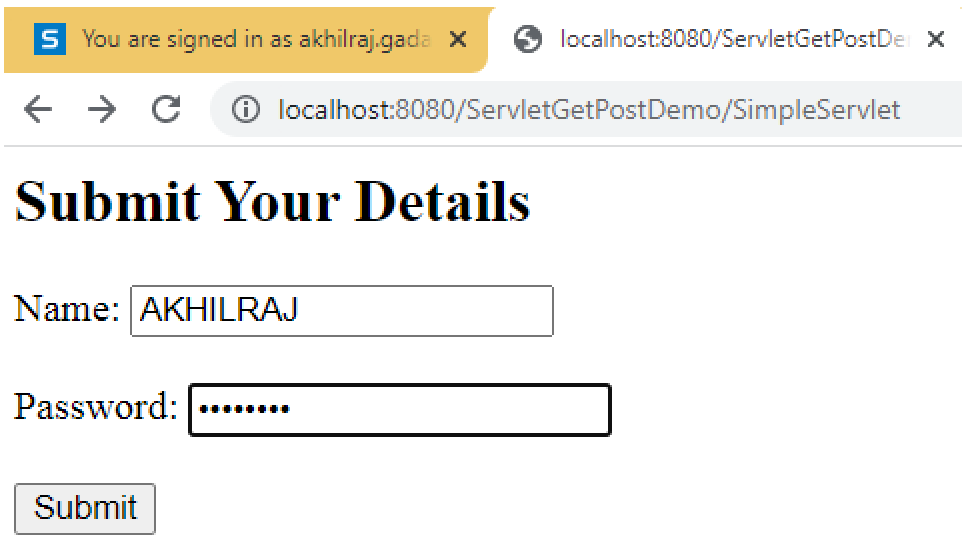
- For GET requests, respond with a simple HTML page that includes a form to submit a name parameter.
- For POST requests, retrieve username and password parameters from the request and respond with an HTML page displaying these parameters.

Include necessary servlet methods to read data from the client and send appropriate responses.

SimpleServlet.java

```
package com.example;

import java.io.IOException;
import java.io.PrintWriter;
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("/SimpleServlet")
public class SimpleServlet extends HttpServlet {

     private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, 
HttpServletResponse response) throws ServletException, 
IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h2>Submit Your Details</h2>");
        out.println("<form method='POST' 
action='SimpleServlet'>");
        out.println("Name: <input type='text' 
name='name'><br><br>");
        out.println("Password: <input type='password' 
name='password'><br><br>");
        out.println("<input type='submit' value='Submit'>");
        out.println("</form>");
        out.println("</body></html>");
    }

    @Override
    protected void doPost(HttpServletRequest request, 
HttpServletResponse response) throws ServletException, 
IOException {
        String name = request.getParameter("name");
        String password = request.getParameter("password");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h2>Submitted Data</h2>");
        out.println("Name: " + name + "<br>");
        out.println("Password: " + password + "<br>");
        out.println("</body></html>");
    }
}
```

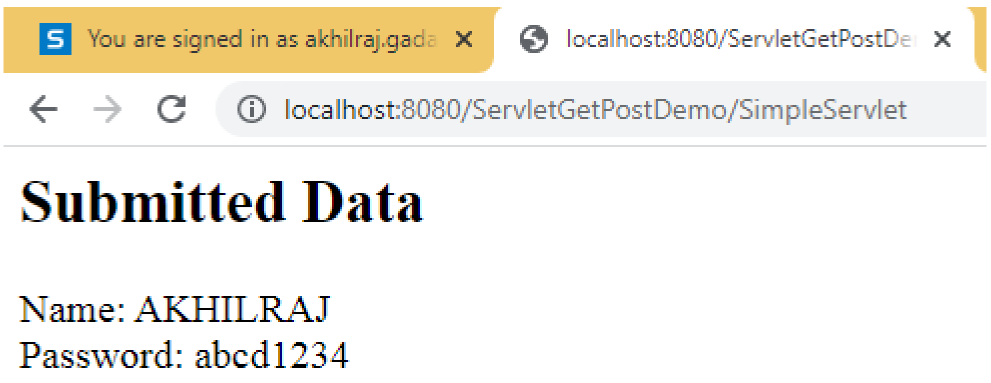
Expected Output


You are signed in as akhilraj.gada
localhost:8080/ServletGetPostDe
localhost:8080/ServletGetPostDemo/SimpleServlet
Submit

Submit Your Details

Name: AKHILRAJ

Password: ••••••••


You are signed in as akhilraj.gada
localhost:8080/ServletGetPostDe
localhost:8080/ServletGetPostDemo/SimpleServlet

Submitted Data

Name: AKHILRAJ
Password: abcd1234

Experiment No. 1b

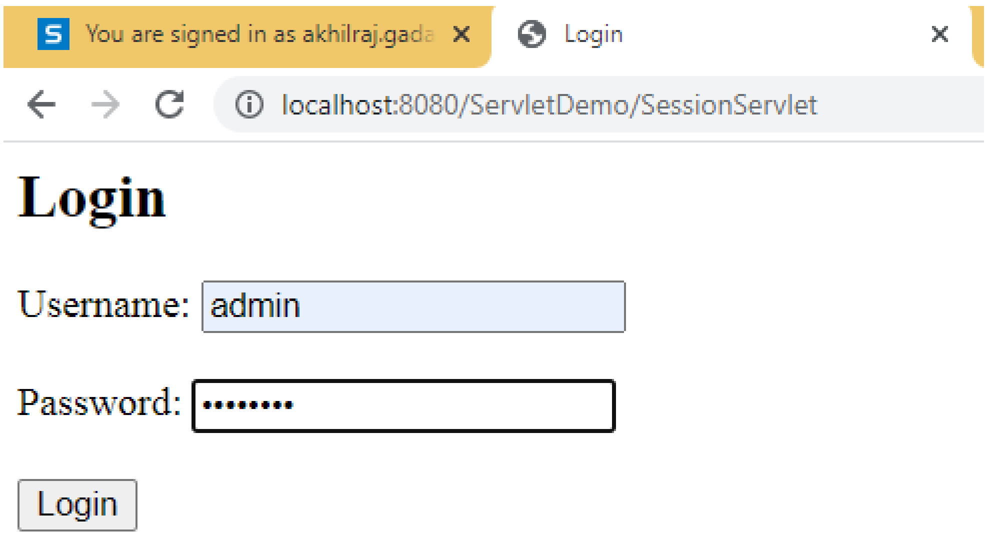
Implement a Java Servlet (`SessionServlet`) to demonstrate session tracking using cookies and session objects. The servlet should:

- Display a login form when accessed initially.
- Handle login functionality via a POST request, validating username and password.
- Upon successful login, store username in a session attribute and display a welcome message with logout option.
- Implement logout functionality to invalidate the session upon user request.

Display session information including session ID, creation time, and last accessed time on the response page.

```
package com.example;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/ServletDemo/SessionServlet")
public class SessionServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use
following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet
SessionServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet SessionServlet at " +
request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }

    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Check if the logout action is requested
        String action = request.getParameter("action");
        if ("logout".equals(action)) {
            HttpSession session = request.getSession(false);
            if (session != null) {
                session.invalidate(); // Invalidate the session
            }
            response.sendRedirect("SessionServlet"); //
Redirect to the login page
            return;
        }

        // Check if session exists
        HttpSession session = request.getSession(false);
        if (session != null && session.getAttribute("username")
!= null) {
            // User is logged in, display welcome message and
session info

out.println("<html><head><title>Welcome</title></head><body>");
            out.println("<h2>Welcome, " +
session.getAttribute("username") + "!</h2>");
            out.println("<p><a
href='SessionServlet?action=logout'>Logout</a></p>");
            out.println("<h3>Session Information:</h3>");
            out.println("<p>Session ID: " + session.getId() +
"</p>");
            out.println("<p>Creation Time: " + new
java.util.Date(session.getCreationTime()) + "</p>");
            out.println("<p>Last Accessed Time: " + new
java.util.Date(session.getLastAccessedTime()) + "</p>");
            out.println("</body></html>");
        } else {
            // User is not logged in, display login form

out.println("<html><head><title>Login</title></head><body>");
            out.println("<h2>Login</h2>");
            out.println("<form method='post'
action='SessionServlet'>");
            out.println("Username: <input type='text'
name='username'><br><br>");
            out.println("Password: <input type='password'
name='password'><br><br>");
            out.println("<input type='submit' value='Login'>");
            out.println("</form>");
            out.println("</body></html>");
        }
    }
    protected void doPost(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        // Process login form submission
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // For demonstration purposes, validate against
hardcoded credentials
        if (username != null && username.equals("admin") &&
password != null && password.equals("password")) {
            // Login successful, create session and store
username
            HttpSession session = request.getSession(true); //
create new session if not exists
            session.setAttribute("username", username);

            // Redirect to doGet to display welcome message and
session info
            response.sendRedirect("SessionServlet");
        } else {
            // Login failed, redirect back to login page
            response.sendRedirect("SessionServlet");
        }
    }

    protected void doDelete(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        // Process logout (via DELETE request)
        HttpSession session = request.getSession(false);
        if (session != null) {
            session.invalidate(); // invalidate session
        }
        response.sendRedirect("SessionServlet");
    }

}
```

Expected Output


You are signed in as akhilraj.gada

Login

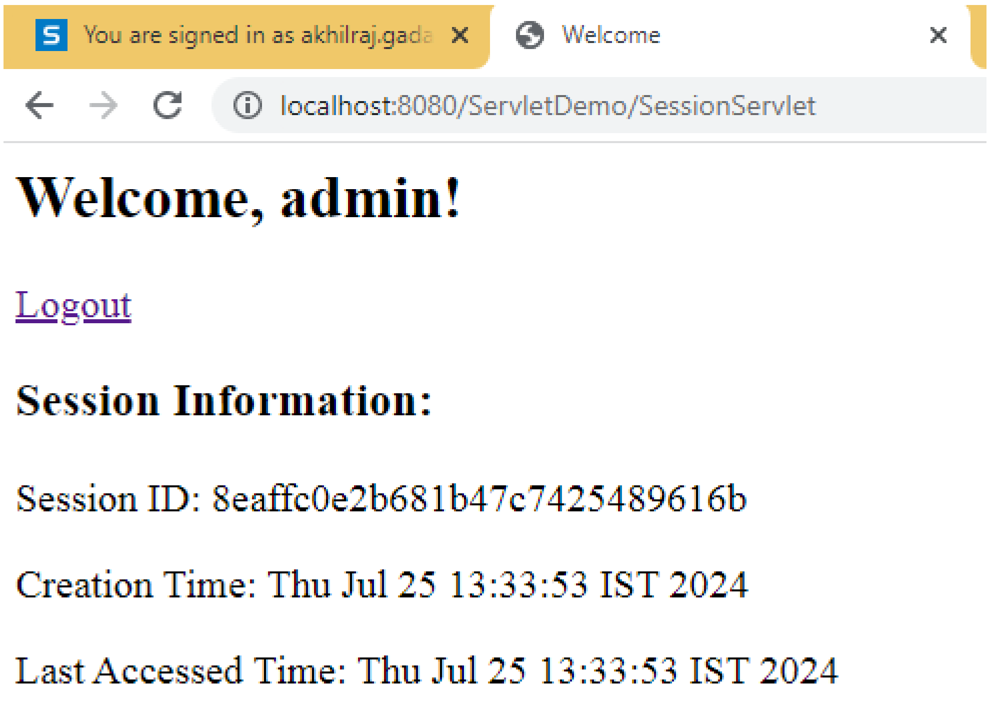
localhost:8080/ServletDemo/SessionServlet

Login

Username: admin

Password: •••••••••

Login


You are signed in as akhilraj.gada

Welcome

localhost:8080/ServletDemo/SessionServlet

Welcome, admin!

Logout

Session Information:

Session ID: 8eaffc0e2b681b47c7425489616b

Creation Time: Thu Jul 25 13:33:53 IST 2024

Last Accessed Time: Thu Jul 25 13:33:53 IST 2024

Experiment No. 2a

Design a JSP application for managing employee information.

- Create a JSP page (employeeForm.jsp) with an HTML form to collect employee data (e.g., Employee ID, Name, Age, Department, and Email) using the POST method.
- Create a JSP page (employeeResult.jsp) to retrieve the submitted form data using request.getParameter().
- Use JSP expressions and scriptlets to process the retrieved data. For example:
    - Concatenate the employee's name and department into a single string.
    - Calculate the birth year from the age.
    - Format the email address.

Display the processed data in a formatted manner in employeeResult.jsp.

<u>employeeForm.jsp</u>

```
<!DOCTYPE html>
<html>
<head>
    <title>Employee Form</title>
</head>
<body>
    <h2>Employee Information Form</h2>
     <form action="employeeResult.jsp" method="POST">
        <table>
            <tr>
                <td>Employee ID:</td>
                <td><input type="text" name="employeeId"
required></td>
            </tr>
            <tr>
                <td>Name:</td>
                <td><input type="text" name="name"
required></td>
            </tr>
            <tr>
                <td>Age:</td>
                <td><input type="number" name="age"
required></td>
            </tr>
            <tr>
                <td>Department:</td>
                <td><input type="text" name="department"
required></td>
            </tr>
            <tr>
                <td>Email:</td>
                <td><input type="email" name="email"
required></td>
            </tr>
            <tr>
```

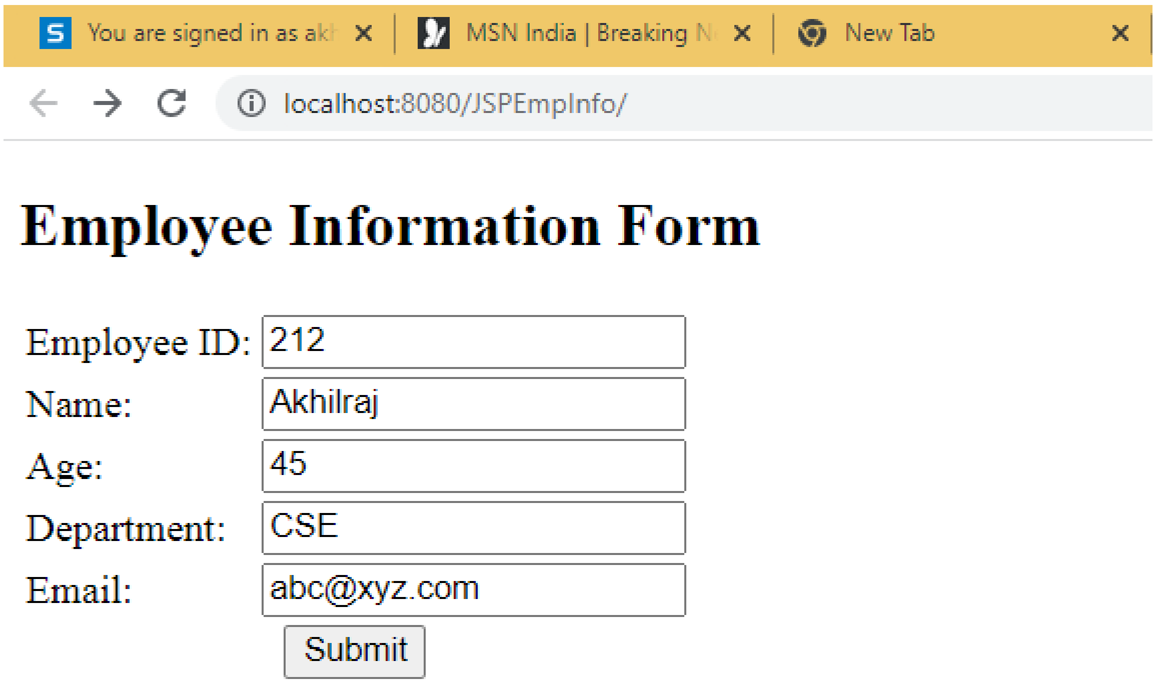
```
                <td colspan="2" style="text-align: center;">
                    <input type="submit" value="Submit">
                </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

<u>employeeResult.jsp</u>

```
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<%@ page import="java.util.Calendar" %>
<!DOCTYPE html>
<html>
<head>
    <title>Employee Information</title>
</head>
<body>
    <h2>Employee Information Result</h2>
    <%
        // Retrieve form data
        String employeeId = request.getParameter("employeeId");
        String name = request.getParameter("name");
        String ageStr = request.getParameter("age");
        String department = request.getParameter("department");
        String email = request.getParameter("email");

        // Process data
        int age = Integer.parseInt(ageStr);
        Calendar calendar = Calendar.getInstance();
        int currentYear = calendar.get(Calendar.YEAR);
        int birthYear = currentYear - age;
        String nameAndDepartment = name + " (" + department +
")";

        // Format email
        String formattedEmail = email.toLowerCase();
    %>
    <table border="1">
        <tr>
            <th>Employee ID</th>
            <td><%= employeeId %></td>
        </tr>
        <tr>
            <th>Name and Department</th>
            <td><%= nameAndDepartment %></td>
        </tr>
        <tr>
```

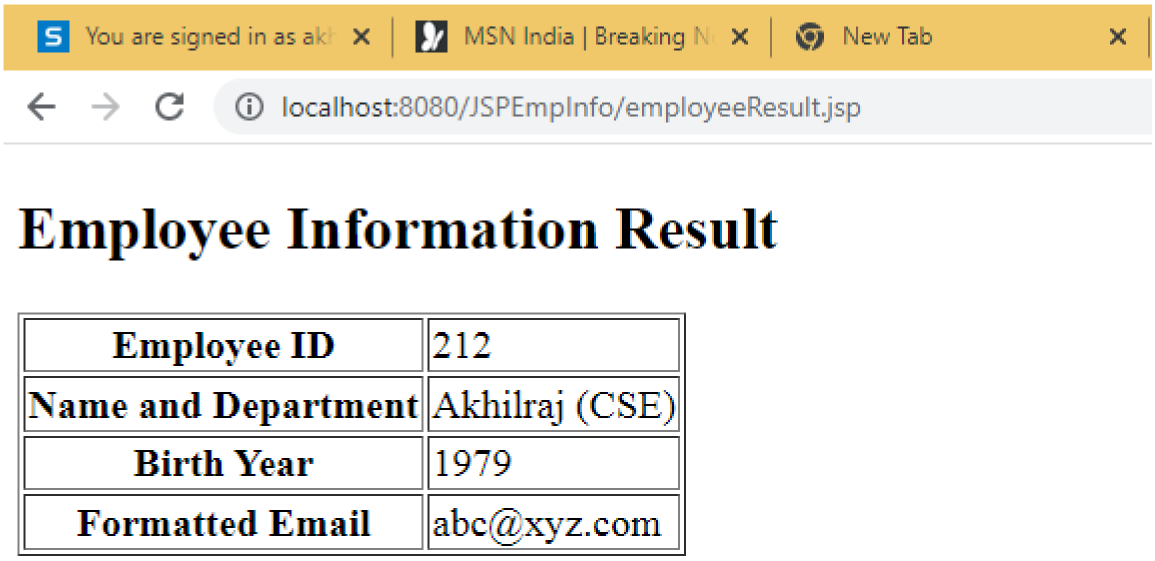
```
            <th>Birth Year</th>
            <td><%= birthYear %></td>
        </tr>
        <tr>
            <th>Formatted Email</th>
            <td><%= formattedEmail %></td>
        </tr>
    </table>
</body>
</html>
```

Expected Output



Employee Information Form

Employee ID:	212
Name:	Akhilraj
Age:	45
Department:	CSE
Email:	abc@xyz.com



Employee Information Result

Employee ID	212
Name and Department	Akhilraj (CSE)
Birth Year	1979
Formatted Email	abc@xyz.com

Experiment No. 2b
Enhance the employeeForm.jsp page to implement robust server-side validation using JSP for managing employee information.

- Ensure that all required fields (such as Employee ID, Name, Age, Department, and Email) are properly validated in employeeResult.jsp upon form submission.
- Implement server-side checks to verify that:
  - Employee ID is a non-empty string.
  - Name contains only letters and spaces.
  - Age is a valid numeric value between 18 and 65.
  - Department is selected from a predefined list of options.
  - Email follows a valid format (e.g., name@example.com).

Display clear and descriptive error messages directly on employeeResult.jsp next to each field if validation criteria are not met, guiding users to correct their input.

employeeForm.jsp

```
<!DOCTYPE html>
<html>
<head>
    <title>Employee Form</title>
</head>
<body>
    <h2>Employee Information Form</h2>
    <form action="employeeResult.jsp" method="POST">
        Employee ID: <input type="text" name="employeeId"
required><br>
        Name: <input type="text" name="name" required><br>
        Age: <input type="number" name="age" required><br>
        Department:
        <select name="department" required>
            <option value="">Select</option>
            <option value="HR">HR</option>
            <option value="Finance">Finance</option>
            <option value="IT">IT</option>
            <option value="Sales">Sales</option>
        </select><br>
        Email: <input type="email" name="email" required><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

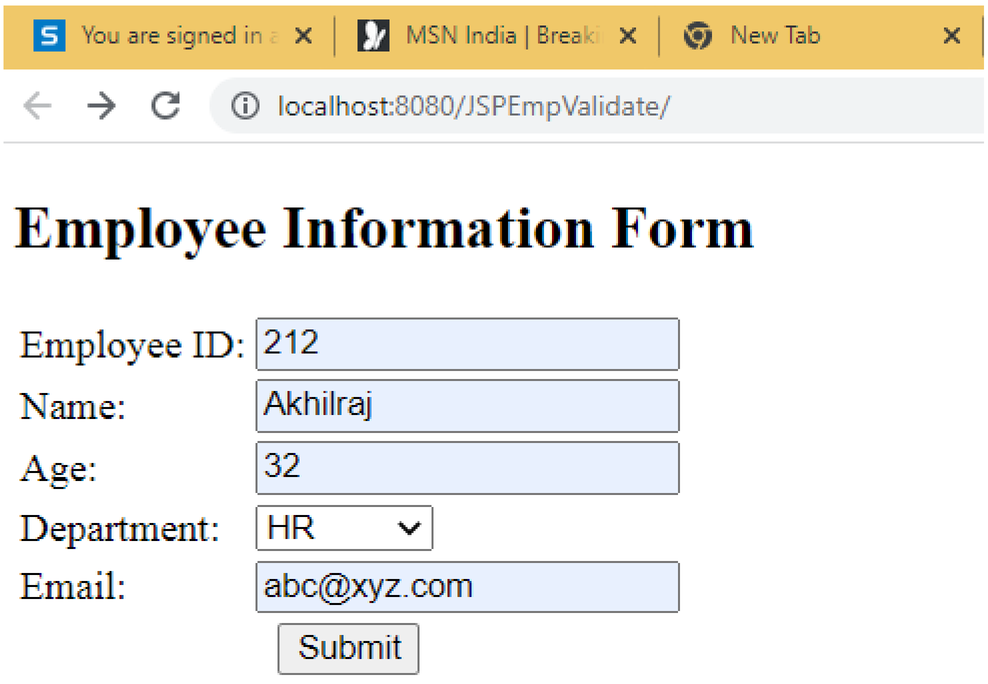
employeeResult.jsp

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

```
<%@ page import="java.util.regex.*" %>
<!DOCTYPE html>
<html>
<head>
    <title>Employee Information</title>
</head>
<body>
    <h2>Employee Information Result</h2>
    <%
        // Retrieve form data
        String employeeId = request.getParameter("employeeId");
        String name = request.getParameter("name");
        String ageStr = request.getParameter("age");
        String department = request.getParameter("department");
        String email = request.getParameter("email");

        // Initialize error messages
        String employeeIdError = "";
        String nameError = "";
        String ageError = "";
        String departmentError = "";
        String emailError = "";

        // Validation flags
        boolean isValid = true;

        // Validate Employee ID
        if (employeeId == null || employeeId.trim().isEmpty())
{
            employeeIdError = "Employee ID is required.";
            isValid = false;
        }

        // Validate Name
        if (name == null || !name.matches("^[a-zA-Z\\s]+$")) {
            nameError = "Name must contain only letters and
spaces.";
            isValid = false;
        }

        // Validate Age
        int age = 0;
        try {
            age = Integer.parseInt(ageStr);
            if (age < 18 || age > 65) {
                ageError = "Age must be between 18 and 65.";
                isValid = false;
            }
        } catch (NumberFormatException e) {
            ageError = "Age must be a valid numeric value.";
            isValid = false;
        }

        // Validate Department
        if (department == null || department.trim().isEmpty())
{
            departmentError = "Department is required.";
            isValid = false;
        } else if (!department.matches("HR|Finance|IT|Sales"))
{
            departmentError = "Invalid department selected.";
            isValid = false;
        }

        // Validate Email
        if (email == null || !email.matches("^[\\w.-]+@[\\w.-
]+\\.[a-zA-Z]{2,}$")) {
            emailError = "Email must be in a valid format
(e.g., name@example.com).";
            isValid = false;
        }

        if (isValid) {
            // Process data
            int currentYear =
java.util.Calendar.getInstance().get(java.util.Calendar.YEAR);
            int birthYear = currentYear - age;
            String nameAndDepartment = name + " (" + department
+ ")";

            // Format email
            String formattedEmail = email.toLowerCase();
    %>
            <table border="1">
                <tr>
                    <th>Employee ID</th>
                    <td><%= employeeId %></td>
                </tr>
                <tr>
                    <th>Name and Department</th>
                    <td><%= nameAndDepartment %></td>
                </tr>
                <tr>
                    <th>Birth Year</th>
                    <td><%= birthYear %></td>
                </tr>
                <tr>
                    <th>Formatted Email</th>
                    <td><%= formattedEmail %></td>
                </tr>
            </table>
    <%
```

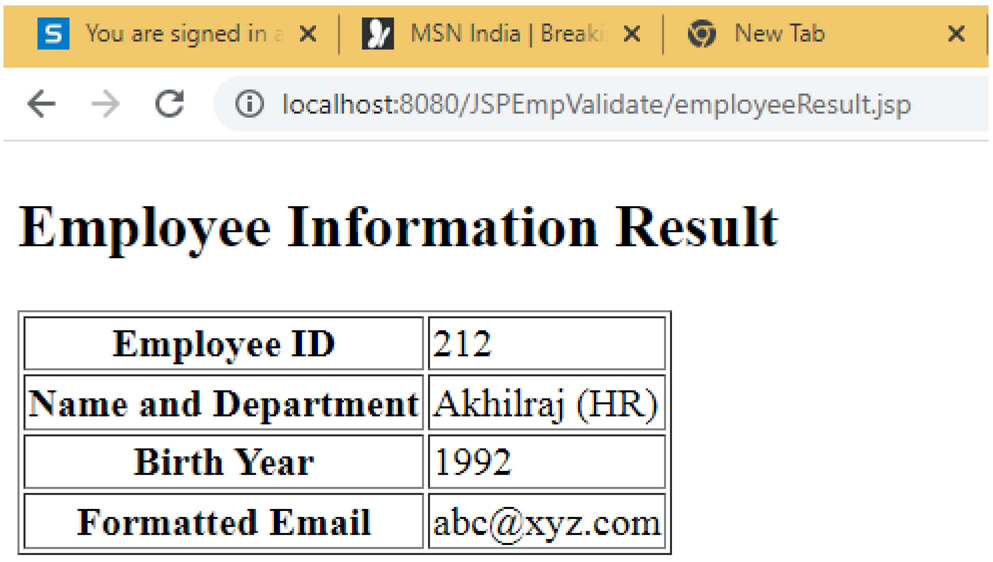
```
        } else {
            // Display error messages and guide the user
    %>
            <h3>Validation Errors</h3>
            <ul>
                <% if (!employeeIdError.isEmpty()) { %>
                    <li><b>Employee ID:</b> <%= employeeIdError
%></li>
                <% } %>
                <% if (!nameError.isEmpty()) { %>
                    <li><b>Name:</b> <%= nameError %></li>
                <% } %>
                <% if (!ageError.isEmpty()) { %>
                    <li><b>Age:</b> <%= ageError %></li>
                <% } %>
                <% if (!departmentError.isEmpty()) { %>
                    <li><b>Department:</b> <%= departmentError
%></li>
                <% } %>
                <% if (!emailError.isEmpty()) { %>
                    <li><b>Email:</b> <%= emailError %></li>
                <% } %>
            </ul>
            <!--<a href="employeeForm.jsp">Go back to the
form</a>-->
            <a href="index.html">Go back to the form</a>
    <%
        }
    %>
</body>
</html>
```

Expected Output



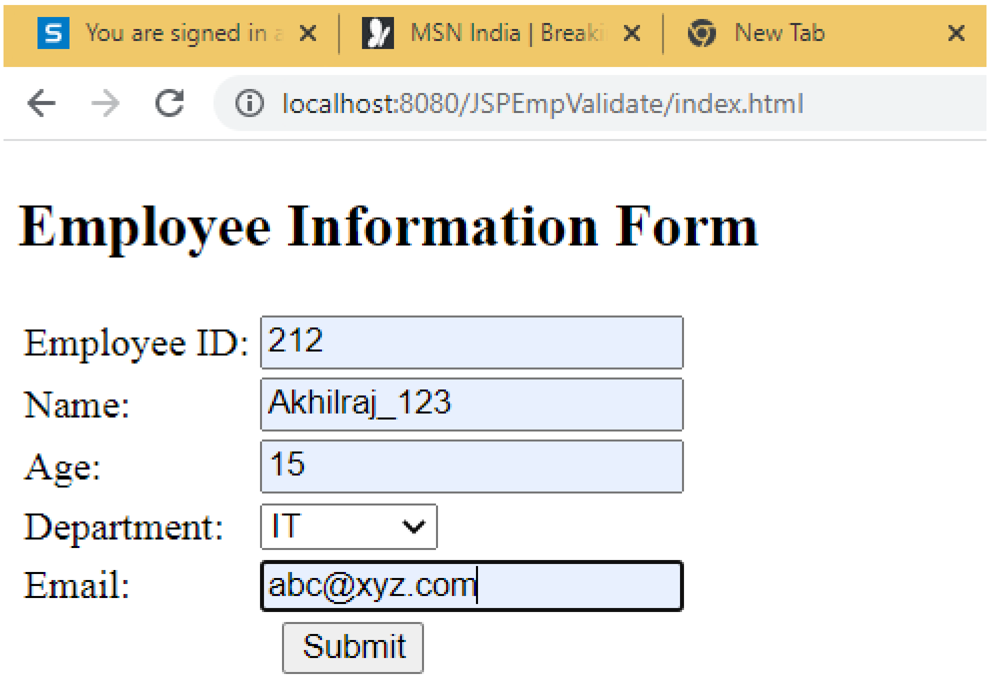
Employee Information Form

Employee ID:	212
Name:	Akhilraj
Age:	32
Department:	HR
Email:	abc@xyz.com



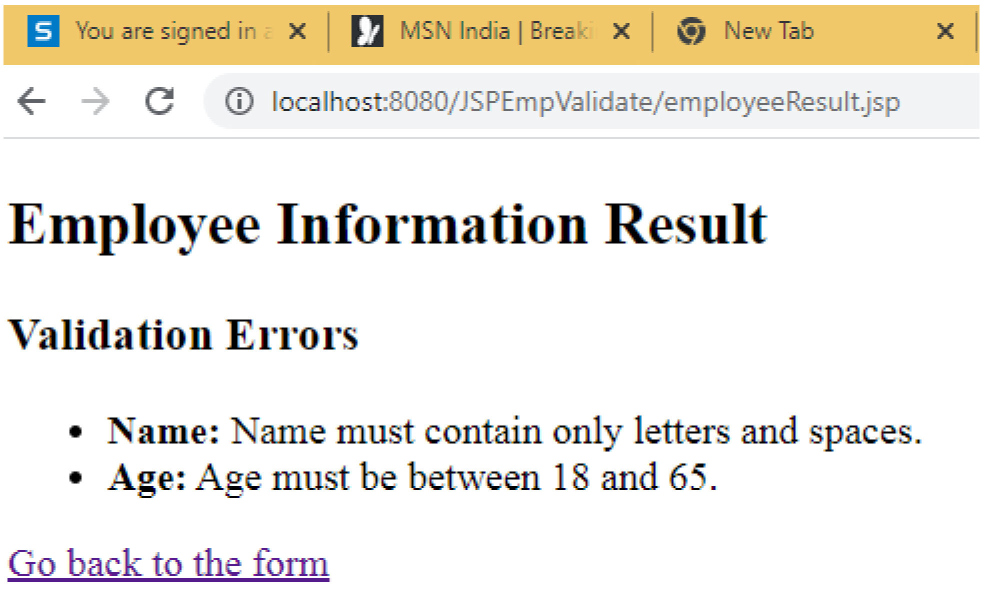
Employee Information Result

Employee ID	212
Name and Department	Akhilraj (HR)
Birth Year	1992
Formatted Email	abc@xyz.com



Employee Information Form

Employee ID:	212
Name:	Akhilraj_123
Age:	15
Department:	IT
Email:	abc@xyz.com



Employee Information Result

Validation Errors

- **Name:** Name must contain only letters and spaces.
- **Age:** Age must be between 18 and 65.

Go back to the form

Experiment No. 3

Develop a JSP application that integrates JDBC to perform CRUD (Create, Read, Update, and Delete) operations on a `students` database table. The application should:

- Create: Insert new student records with fields such as roll-no, name, email, and age.
- Read: Retrieve and display all student records from the database.
- Update: Modify existing student records.
- Delete: Remove student records from the database.

DBConnection.java

```
package com.studentcrud;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
    private static String jdbcURL =
"jdbc:mysql://localhost:3306/studentDB";

    private static String jdbcUsername = "root";
    private static String jdbcPassword = "password";

    public static Connection getConnection() {
        Connection connection = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection(jdbcURL,
jdbcUsername, jdbcPassword);
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }
}
```

Student.java

```
package com.studentcrud;

public class Student {
    private int rollNo;
    private String name;
    private String email;
    private int age;

    // Getters
    public int getRollNo() {
        return rollNo;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }

    public int getAge() {
        return age;
    }

    // Setters
    public void setRollNo(int rollNo) {
        this.rollNo = rollNo;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public void setAge(int age) {
        this.age = age;
    }

    // toString method
    @Override
    public String toString() {
        return "Student{" +
                "rollNo=" + rollNo +
                ", name='" + name + '\'' +
                ", email='" + email + '\'' +
                ", age=" + age +
                '}';
    }
}
```

<u>StudentDAO.java</u>

```
package com.studentcrud;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
```

```
public class StudentDAO {
    public void addStudent(Student student) {
        String sql = "INSERT INTO students (roll_no, name,
email, age) VALUES (?, ?, ?, ?)";
        try (Connection connection =
DBConnection.getConnection();
             PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {
            preparedStatement.setInt(1, student.getRollNo());
            preparedStatement.setString(2, student.getName());
            preparedStatement.setString(3,
student.getEmail());
            preparedStatement.setInt(4, student.getAge());
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public List<Student> getAllStudents() {
        List<Student> students = new ArrayList<>();
        String sql = "SELECT * FROM students";
        try (Connection connection =
DBConnection.getConnection();
             Statement statement =
connection.createStatement();
             ResultSet resultSet =
statement.executeQuery(sql)) {
            while (resultSet.next()) {
                Student student = new Student();

student.setRollNo(resultSet.getInt("roll_no"));
                student.setName(resultSet.getString("name"));

student.setEmail(resultSet.getString("email"));
                student.setAge(resultSet.getInt("age"));
                students.add(student);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return students;
    }

    public void updateStudent(Student student) {
        String sql = "UPDATE students SET name = ?, email = ?,
age = ? WHERE roll_no = ?";
        try (Connection connection =
DBConnection.getConnection();
```

```
             PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {
            preparedStatement.setString(1, student.getName());
            preparedStatement.setString(2,
student.getEmail());
            preparedStatement.setInt(3, student.getAge());
            preparedStatement.setInt(4, student.getRollNo());
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void deleteStudent(int rollNo) {
        String sql = "DELETE FROM students WHERE roll_no = ?";
        try (Connection connection =
DBConnection.getConnection();
             PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {
            preparedStatement.setInt(1, rollNo);
            preparedStatement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

<u>addStudent.jsp</u>

```
<%@ page import="com.studentcrud.Student,
com.studentcrud.StudentDAO" %>
<%
    if(request.getParameter("rollNo") != null) {
        int rollNo =
Integer.parseInt(request.getParameter("rollNo"));
        String name = request.getParameter("name");
        String email = request.getParameter("email");
        int age =
Integer.parseInt(request.getParameter("age"));

        Student student = new Student();
        student.setRollNo(rollNo);
        student.setName(name);
        student.setEmail(email);
        student.setAge(age);

        StudentDAO dao = new StudentDAO();
        dao.addStudent(student);

        out.println("Student added successfully.");
    }
```

```
%>
<html>
<body>
    <h2>Add Student</h2>
    <form action="addStudent.jsp" method="post">
        Roll No: <input type="text" name="rollNo"><br>
        Name: <input type="text" name="name"><br>
        Email: <input type="text" name="email"><br>
        Age: <input type="text" name="age"><br>
        <input type="submit" value="Add Student">
    </form>
    <a href="viewStudents.jsp">View Students</a>
</body>
</html>
```

viewStudents.jsp

```
<%@ page import="java.util.List, com.studentcrud.Student,
com.studentcrud.StudentDAO" %>
<%
    StudentDAO dao = new StudentDAO();
    List<Student> students = dao.getAllStudents();
%>
<html>
<body>
    <h2>View Students</h2>
    <table border="1">
        <tr>
            <th>Roll No</th>
            <th>Name</th>
            <th>Email</th>
            <th>Age</th>
            <th>Actions</th>
        </tr>
        <%
            for(Student student : students) {
        %>
        <tr>
            <td><%= student.getRollNo() %></td>
            <td><%= student.getName() %></td>
            <td><%= student.getEmail() %></td>
            <td><%= student.getAge() %></td>
            <td>
                <a href="updateStudent.jsp?rollNo=<%=
student.getRollNo() %>">Edit</a> |
                <a href="deleteStudent.jsp?rollNo=<%=
student.getRollNo() %>">Delete</a>
            </td>
        </tr>
        <%
            }
        %>
    </table>
    <a href="addStudent.jsp">Add Student</a>
</body>
</html>
```

updateStudent.jsp

```
<%@ page import="com.studentcrud.Student,
com.studentcrud.StudentDAO" %>
<%
    StudentDAO dao = new StudentDAO();
    int rollNo =
Integer.parseInt(request.getParameter("rollNo"));
    Student student = new Student();
    for(Student s : dao.getAllStudents()) {
        if(s.getRollNo() == rollNo) {
            student = s;
            break;
        }
    }

    if(request.getParameter("name") != null) {
        String name = request.getParameter("name");
        String email = request.getParameter("email");
        int age =
Integer.parseInt(request.getParameter("age"));

        student.setName(name);
        student.setEmail(email);
        student.setAge(age);

        dao.updateStudent(student);
        out.println("Student updated successfully.");
    }
%>

<html>
<body>
    <h2>Update Student</h2>
    <form action="updateStudent.jsp?rollNo=<%=
student.getRollNo() %>" method="post">
        Name: <input type="text" name="name" value="<%=
student.getName() %>"><br>
        Email: <input type="text" name="email" value="<%=
student.getEmail() %>"><br>
        Age: <input type="text" name="age" value="<%=
student.getAge() %>"><br>
        <input type="submit" value="Update Student">
    </form>
    <a href="viewStudents.jsp">View Students</a>
</body>
</html>
```

deleteStudent.jsp

```
<%@ page import="com.studentcrud.StudentDAO" %>
<%
    int rollNo =
Integer.parseInt(request.getParameter("rollNo"));
    StudentDAO dao = new StudentDAO();
    dao.deleteStudent(rollNo);
    out.println("Student deleted successfully.");
%>
<html>
<body>
    <a href="viewStudents.jsp">View Students</
```

Experiment No. 4
Develop a web application using Spring Boot to perform CRUD operations on Book information. The application should use Thymeleaf for the view layer, Spring MVC for the controller layer, Spring Data JPA and Hibernate for data access, and MySQL as the database. The application should manage book attributes such as title, authors, edition, publication and price.

Staff.java

```
package com.example.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Staff {

     @Id
     @GeneratedValue(strategy = GenerationType.IDENTITY)
     Long id;
     String name;
     int age;
     String email;

     public Staff() {
          // TODO Auto-generated constructor stub
     }

     public Long getId() {
          return id;
     }

     public void setId(Long id) {
          this.id = id;
     }

     public String getName() {
          return name;
     }

     public void setName(String name) {
          this.name = name;
     }

     public int getAge() {
          return age;
     }

     public void setAge(int age) {
          this.age = age;
     }

     public String getEmail() {
          return email;
     }

     public void setEmail(String email) {
          this.email = email;
     }

     @Override
     public String toString() {
          return "Staff [id=" + id + ", name=" + name + ",
age=" + age + ", email=" + email + "]";
     }


}
```

StaffManager.java

```
package com.example.demo;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class StaffManager {

     @Autowired
     StaffRepository repository;

     public List<Staff> getAllStaff(){
          return repository.findAll();
     }

     public Staff getStaffById(Long id) {
          return repository.findById(id).get();
     }

     public void saveStaff(Staff staff) {
          repository.save(staff);
     }

     public void deleteStaff(Long id) {
          repository.deleteById(id);
     }
}
```

StaffRepository.java

```
package com.example.demo;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface StaffRepository extends JpaRepository<Staff,
Long> {

}
```

AppController.java

```
package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class AppController {

     @Autowired
     StaffManager manager;

     @GetMapping("/")
     public String showHomePage(Model model) {

     model.addAttribute("stafflist",manager.getAllStaff());
          return "index";
     }

     @GetMapping("/new")
     public String showCreatePage() {
          return "create_book_form";
     }

     @PostMapping("/save")
     public String saveStaff(@ModelAttribute Staff staff) {
          manager.saveStaff(staff);
          return "redirect:/";
     }

     @GetMapping("/edit/{id}")
     public String editStaff(@PathVariable Long id, Model
model) {
          model.addAttribute("staff",
manager.getStaffById(id));
          return "edit_form";

     }

     @GetMapping("/delete/{id}")
     public String deleteStaff(@PathVariable Long id, Model
model) {
          manager.deleteStaff(id);
          return "redirect:/";

     }

}
```

index.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<div align="center">
<h1>Staff List</h1>
<br/>
<a href="/new">Create New Staff</a>
<table border="1" cellpadding="10" cellspacing="0">
<tr>
     <th>ID</th>
     <th>Name</th>
     <th>Age</th>
     <th>Email</th>
     <th>Action</th>
</tr>
<tr th:each ="staff:${stafflist}">
     <td th:text="${staff.id}"/>
     <td th:text="${staff.name}"/>
     <td th:text="${staff.age}"/>
     <td th:text="${staff.email}"/>
     <td>
          <a th:href="@{'/edit/' + ${staff.id}}">Edit</a>
          <a th:href="@{'/delete/' + ${staff.id}}">Delete</a>
     </td>
     </tr>
</table>
</div>
```

```
</body>
</html>
```

create_book_form.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<div align="center">
<h1>Create New Staff</h1>
<br/>
<form action="/save" method="post">
<table border="0" cellpadding="10" cellspacing="10">
	<tr>
		<td>Name</td>
		<td><Input type="text" name="name"/></td>
	</tr>
	<tr>
		<td>Age</td>
		<td><Input type="text" name="age"/></td>
	</tr>
	<tr>
		<td>Email</td>
		<td><Input type="text" name="email"/></td>
	</tr>
	<tr>
		<td colspan="2" align="center">
			<button type="submit">Save</button>
		</td>
	</tr>
</table>
</form>
</div>
</body>
</html>
```

edit_form.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<div align="center">
```

```
<h1>Edit Staff</h1>
<br/>
<form action="#" th:action="@{/save}" th:object="${staff}"
method="post">
<table border="0" cellpadding="10" cellspacing="0">
     <tr>
          <td>ID</td>
          <td><input type="text" th:field="*{id}"
readonly="readonly"/></td>
     </tr>
     <tr>
          <td>Name</td>
          <td><input type="text" th:field="*{name}"/></td>
     </tr>
     <tr>
          <td>Age</td>
          <td><input type="text" th:field="*{age}"/></td>
     </tr>
     <tr>
          <td>Email</td>
          <td><input type="text" th:field="*{email}"/></td>
     </tr>
     <tr>
          <td colspan="2" align="center">
               <button type="submit">Save</button>
          </td>
     </tr>
</table>

</form>
</div>
</body>
</html>
```

Experiment No. 5
Develop a Spring Boot application that provides RESTful APIs for performing CRUD operations on Mobile information. The application should use Spring Data JPA and Hibernate for data access to MySQL database that stores Mobile details such as model name, model number, device operating system, manufacturer, and country of origin. JSON may be considered as the return value of the API. Encapsulate error handling feature for situations such as record not found. Demonstrate APIs working using curl command and Postman API testing tool.

Staff.java

```
package com.example.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Staff {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	Long id;
	String name;
	int age;
	String email;

	public Staff() {
		// TODO Auto-generated constructor stub
	}

	public Long getId() {
		return id;
	}

	public void setId(Long id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public int getAge() {
		return age;
	}
	public void setAge(int age) {
		this.age = age;
	}

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	@Override
	public String toString() {
		return "Staff [id=" + id + ", name=" + name + ", 
age=" + age + ", email=" + email + "]";
	}


}
```

StaffManager.java

```
package com.example.demo;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class StaffManager {

	@Autowired
	StaffRepository repository;

	public List<Staff> getAllStaff(){
		return repository.findAll();
	}

	public Staff getStaffById(Long id) {
		return repository.findById(id).get();
	}

	public void saveStaff(Staff staff) {
		repository.save(staff);
	}

	public void deleteStaff(Long id) {
		repository.deleteById(id);
	}

	public Boolean exists(Long id) {
		return repository.existsById(id);
	}
}
```

StaffRepository.java

```
package com.example.demo;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface StaffRepository extends JpaRepository<Staff,
Long> {

}
```

AppController.java

```
package com.example.demo;

import java.util.List;
import java.util.NoSuchElementException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.HttpStatusCode;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class AppController {

	@Autowired
	StaffManager manager;

	@GetMapping("/staff")
	public List<Staff> listAllStaff() {
		return manager.getAllStaff();
```

```
     }

//   @GetMapping("/staff/{id}")
//   public Staff getStaff(@PathVariable Long id) {
//        return manager.getStaffById(id);
//   }

     @GetMapping("/staff/{id}")
     public ResponseEntity getStaff(@PathVariable Long id) {
          try {
               Staff staff = manager.getStaffById(id);
               return                                        new
ResponseEntity<Staff>(staff,HttpStatus.OK);
               }
               catch(NoSuchElementException e) {
                    return                                   new
ResponseEntity<Staff>(HttpStatus.NOT_FOUND);
               }
     }

     @PostMapping("/staff")
     public void saveStaff(@RequestBody Staff staff) {
          manager.saveStaff(staff);
     }

     @PutMapping("/staff")
     public void updateStaff(@RequestBody Staff staff) {
          manager.saveStaff(staff);
     }

//   @DeleteMapping("/staff/{id}")
//   public void deleteStaff(@PathVariable Long id) {
//        manager.deleteStaff(id);
//   }


     //Using  <Staff>  Template  for  ResponseEntity  is  Optional
here
     @DeleteMapping("/staff/{id}")
     public  ResponseEntity<Staff>  deleteStaff  (@PathVariable
Long id) {

          if(manager.exists(id))
          {
               manager.deleteStaff(id);
               return new ResponseEntity<Staff>(HttpStatus.OK);
          }
          else {
               return                                        new
ResponseEntity<Staff>(HttpStatus.NOT_FOUND);
          }
          }
      }
```