

ASSIGNMENT-MCA424

Struts , Hibernate and Spring

Submitted By: MUHAMMAD ANSHAD P A

Roll no: 18

MCA

Employee Management System –

using Struts framework and Hibernate for database operations.

➤ Struts_EMS/src/main/java/com/example/**action**/EmployeeAction.java

```
package com.example.action;

import com.example.model.Employee;
import com.example.service.EmployeeService;
import com.example.service.EmployeeServiceImpl;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.Preparable;

import java.util.List;
import java.util.regex.Pattern;

public class EmployeeAction extends ActionSupport implements Preparable {

    private Employee employee;
    private List<Employee> employees;
    private Long employeeId;
    private EmployeeService employeeService = new EmployeeServiceImpl();

    // Pattern for email validation
    private static final String EMAIL_PATTERN =
        "^[_A-Za-z0-9-\\+]+(\\.[_A-Za-z0-9-]+)*@"
        + "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$";

    @Override
    public void prepare() throws Exception {
        if (employeeId != null) {
            employee = employeeService.getEmployeeById(employeeId);
        }
    }

    // Method to validate the form
```

```

@Override
public void validate() {
    if (employee != null) {
        // Validate name
        if (employee.getName() == null ||
employee.getName().trim().isEmpty()) {
            addFieldError("employee.name", "Name is required");
        }

        // Validate email
        if (employee.getEmail() == null ||
employee.getEmail().trim().isEmpty()) {
            addFieldError("employee.email", "Email is required");
        } else if (!Pattern.matches(EMAIL_PATTERN, employee.getEmail())) {
            addFieldError("employee.email", "Invalid email format");
        } else if (!employeeService.isEmailUnique(employee.getEmail(),
employee.getId())) {
            addFieldError("employee.email", "Email already exists");
        }

        // Validate salary
        if (employee.getSalary() == null) {
            addFieldError("employee.salary", "Salary is required");
        } else if (employee.getSalary() <= 0) {
            addFieldError("employee.salary", "Salary must be greater than
zero");
        }
    }
}

// Action method to display the form for adding a new employee
public String addForm() {
    this.employee = new Employee();
    return SUCCESS;
}

// Action method to add a new employee
public String add() {
    try {
        employeeService.addEmployee(employee);
        addActionMessage("Employee added successfully");
        return SUCCESS;
    } catch (Exception e) {
        addActionError("Failed to add employee: " + e.getMessage());
        return INPUT;
    }
}

// Action method to display all employees
public String list() {
    try {
        employees = employeeService.getAllEmployees();
        return SUCCESS;
    } catch (Exception e) {
        addActionError("Failed to fetch employees: " + e.getMessage());
    }
}

```

```

        return ERROR;
    }
}

// Action method to display the form for editing an employee
public String editForm() {
    return SUCCESS;
}

// Action method to update an employee
public String update() {
    try {
        employeeService.updateEmployee(employee);
        addActionMessage("Employee updated successfully");
        return SUCCESS;
    } catch (Exception e) {
        addActionError("Failed to update employee: " + e.getMessage());
        return INPUT;
    }
}

// Action method to delete an employee
public String delete() {
    try {
        employeeService.deleteEmployee(employeeId);
        addActionMessage("Employee deleted successfully");
        return SUCCESS;
    } catch (Exception e) {
        addActionError("Failed to delete employee: " + e.getMessage());
        return ERROR;
    }
}

// Getters and Setters
public Employee getEmployee() {
    return employee;
}

public void setEmployee(Employee employee) {
    this.employee = employee;
}

public List<Employee> getEmployees() {
    return employees;
}

public void setEmployees(List<Employee> employees) {
    this.employees = employees;
}

public Long getEmployeeId() {
    return employeeId;
}

```

```
        public void setEmployeeId(Long employeeId) {
            this.employeeId = employeeId;
        }
    }
    ➤
```

➤ **Struts_EMS/src/main/java/com/example/dao/EmployeeDAO.java**

```
package com.example.dao;

import com.example.model.Employee;
import java.util.List;

public interface EmployeeDAO {
    Long addEmployee(Employee employee);
    void updateEmployee(Employee employee);
    void deleteEmployee(Long id);
    Employee getEmployeeById(Long id);
    List<Employee> getAllEmployees();
    boolean isEmailUnique(String email, Long id);
}
```

➤ **Struts_EMS/src/main/java/com/example/dao/EmployeeDAOImpl.java**

```
package com.example.dao;

import com.example.model.Employee;
import com.example.util.HibernateUtil;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.query.Query;
import java.util.List;

public class EmployeeDAOImpl implements EmployeeDAO {

    @Override
    public Long addEmployee(Employee employee) {
        Transaction transaction = null;
        Long employeeId = null;

        try (Session session =
            HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();
            employeeId = (Long) session.save(employee);
            transaction.commit();
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
        }

        return employeeId;
    }

    @Override
    public void updateEmployee(Employee employee) {
        Transaction transaction = null;

        try (Session session =
            HibernateUtil.getSessionFactory().openSession()) {
            transaction = session.beginTransaction();

            Employee existingEmployee = session.get(Employee.class,
                employee.getId());
            if (existingEmployee != null) {
                if (employee.getName() != null &&
                    !employee.getName().isEmpty()) {
                    existingEmployee.setName(employee.getName());
                }
                if (employee.getEmail() != null &&
                    !employee.getEmail().isEmpty()) {
                    existingEmployee.setEmail(employee.getEmail());
                }
            }
            transaction.commit();
        }
    }
}
```

```

        if (employee.getSalary() != null) {
            existingEmployee.setSalary(employee.getSalary());
        }

        session.update(existingEmployee);
        transaction.commit();
    }
} catch (Exception e) {
    if (transaction != null) {
        transaction.rollback();
    }
    e.printStackTrace();
}
}

@Override
public void deleteEmployee(Long id) {
    Transaction transaction = null;

    try (Session session =
        HibernateUtil.getSessionFactory().openSession()) {
        transaction = session.beginTransaction();
        Employee employee = session.get(Employee.class, id);
        if (employee != null) {
            session.delete(employee);
        }
        transaction.commit();
    } catch (Exception e) {
        if (transaction != null) {
            transaction.rollback();
        }
        e.printStackTrace();
    }
}

@Override
public Employee getEmployeeById(Long id) {
    Employee employee = null;

    try (Session session =
        HibernateUtil.getSessionFactory().openSession()) {
        employee = session.get(Employee.class, id);
    } catch (Exception e) {
        e.printStackTrace();
    }

    return employee;
}

@Override
public List<Employee> getAllEmployees() {
    List<Employee> employees = null;

    try (Session session =
        HibernateUtil.getSessionFactory().openSession()) {

```

```

        employees = session.createQuery("FROM Employee",
Employee.class).list();
    } catch (Exception e) {
        e.printStackTrace();
    }

    return employees;
}

@Override
public boolean isEmailUnique(String email, Long id) {
    try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
        String hql = "FROM Employee WHERE email = :email AND id <>
:id";
        Query<Employee> query = session.createQuery(hql,
Employee.class);
        query.setParameter("email", email);
        query.setParameter("id", id != null ? id : -1L); // Avoid null
ID

        Employee employee = query.uniqueResult();

        return employee == null; // If no other employee has this
email, it's unique
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
}

```

➤ **Struts_EMS/src/main/java/com/example/model/Employee.java**

```

package com.example.model;

import javax.persistence.*;

@Entity
@Table(name = "employees")
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "name", nullable = false)
    private String name;

    @Column(name = "email", nullable = false, unique = true)
    private String email;
}

```

```
@Column(name = "salary", nullable = false)
private Double salary;

@Column(name = "designation")
private String designation;

// Constructors
public Employee() {
}

public Employee(String name, String email, Double salary, String
designation) {
    this.name = name;
    this.email = email;
    this.salary = salary;
    this.designation = designation;
}

// Getters and Setters
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 String getEmail() {
    return email;
}

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

public Double getSalary() {
    return salary;
}

public void setSalary(Double salary) {
    this.salary = salary;
}

public String getDesignation() {
    return designation;
}
```



```
public void setDesignation(String designation) {  
    this.designation = designation;  
}  
  
@Override  
public String toString() {  
    return "Employee{" +  
        "id=" + id +  
        ", name='" + name + '\'' +  
        ", email='" + email + '\'' +  
        ", salary=" + salary +  
        ", designation='" + designation + '\'' +  
        '}';  
}  
}
```

```
}
```

➤ **Struts_EMS/src/main/java/com/example/ util/HibernateUtil.java**

```
package com.example.util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory =
        buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            return new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            System.err.println("Initial SessionFactory creation failed: " +
ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }

    public static void shutdown() {
        getSessionFactory().close();
    }
}
```

➤ **Struts_EMS/src/main /webapp/WEB-INF/views/addEmployee.jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<jsp:include page="header.jsp" />

<div class="card">
    <div class="card-header">
        <h3 class="card-title">Add New Employee</h3>
    </div>
    <div class="card-body">
        <s:form action="addEmployee" method="post" theme="simple"
cssClass="needs-validation">
            <div class="form-group">
                <label for="name">Name <span class="text-
danger">*</span></label>
                <s:textfield name="employee.name" cssClass="form-control"
id="name" />
                <s:fielderror fieldName="employee.name" cssClass="error" />
            </div>

            <div class="form-group">
                <label for="email">Email <span class="text-
danger">*</span></label>
                <s:textfield name="employee.email" cssClass="form-control"
id="email" />
                <s:fielderror fieldName="employee.email" cssClass="error"
/>
            </div>

            <div class="form-group">
                <label for="salary">Salary <span class="text-
danger">*</span></label>
                <s:textfield name="employee.salary" cssClass="form-control"
id="salary" type="number" step="0.01" />
                <s:fielderror fieldName="employee.salary" cssClass="error"
/>
            </div>

            <div class="form-group">
                <label for="designation">Designation</label>
                <s:textfield name="employee.designation" cssClass="form-
control" id="designation" />
            </div>

            <div class="form-group">
                <s:submit value="Add Employee" cssClass="btn btn-primary"
/>

                <a href="<s:url action='listEmployees'/">" class="btn btn-
secondary">Cancel</a>
            </div>
        </s:form>
    </div>
</div>
```

```

        </s:form>
    </div>
</div>

<jsp:include page="footer.jsp" />

```

➤ **Struts_EMS/src/main/webapp/WEB-INF/views/editEmployee.jsp**

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<jsp:include page="header.jsp" />

<div class="card">
    <div class="card-header">
        <h3 class="card-title">Edit Employee</h3>
    </div>
    <div class="card-body">
        <s:form action="updateEmployee" method="post" theme="simple"
cssClass="needs-validation">
            <s:hidden name="employee.id" />

            <div class="form-group">
                <label for="name">Name <span class="text-
danger">*</span></label>
                <s:textfield name="employee.name" cssClass="form-control"
id="name" />
                <s:fielderror fieldName="employee.name" cssClass="error" />
            </div>

            <div class="form-group">
                <label for="email">Email <span class="text-
danger">*</span></label>
                <s:textfield name="employee.email" cssClass="form-control"
id="email" />
                <s:fielderror fieldName="employee.email" cssClass="error"
/>
            </div>

            <div class="form-group">
                <label for="salary">Salary <span class="text-
danger">*</span></label>
                <s:textfield name="employee.salary" cssClass="form-control"
id="salary" type="number" step="0.01" />
                <s:fielderror fieldName="employee.salary" cssClass="error"
/>
            </div>

            <div class="form-group">
                <label for="designation">Designation</label>
                <s:textfield name="employee.designation" cssClass="form-
control" id="designation" />
            </div>

```

```

        <div class="form-group">
            <s:submit value="Update Employee" cssClass="btn btn-
primary" />
            <a href="<s:url action='listEmployees' />" class="btn btn-
secondary">Cancel</a>
        </div>
    </s:form>
</div>
</div>
</div>
<jsp:include page="footer.jsp" />

```

➤ **Struts_EMS/src/main /webapp/WEB-INF/views/error.jsp**

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<jsp:include page="header.jsp" />

<div class="card border-danger">
    <div class="card-header bg-danger text-white">
        <h3 class="card-title">Error</h3>
    </div>
    <div class="card-body">
        <h5>An error occurred:</h5>
        <s:if test="hasActionErrors()">
            <div class="alert alert-danger">
                <s:actionerror/>
            </div>
        </s:if>
        <s:else>
            <div class="alert alert-danger">
                Unknown error occurred. Please try again or contact
administrator.
            </div>
        </s:else>
        <a href="<s:url action='listEmployees' />" class="btn btn-
primary">Return to Employee List</a>
    </div>
</div>

<jsp:include page="footer.jsp" />

```

➤ **Struts_EMS/src/main /webapp/WEB-INF/views/listEmployees.jsp**

```

<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<jsp:include page="header.jsp" />

<div class="card">
    <div class="card-header d-flex justify-content-between align-items-
center">
        <h3 class="card-title">Employees</h3>
        <a href="<s:url action='addEmployeeForm'/">" class="btn btn-
primary">Add New Employee</a>
    </div>
    <div class="card-body">
        <s:if test="employees != null && !employees.isEmpty()">
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Name</th>
                        <th>Email</th>
                        <th>Salary</th>
                        <th>Designation</th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody>
                    <s:iterator value="employees">
                        <tr>
                            <td><s:property value="id" /></td>
                            <td><s:property value="name" /></td>
                            <td><s:property value="email" /></td>
                            <td><s:property value="salary" /></td>
                            <td><s:property value="designation" /></td>
                            <td>
                                <a href="<s:url
action='editEmployeeForm'><s:param name='employeeId'
value='%{id}'/></s:url>" class="btn btn-sm btn-info">Edit</a>
                                <a href="<s:url
action='deleteEmployee'><s:param name='employeeId' value='%{id}'/></s:url>"
class="btn btn-sm btn-danger" onclick="return confirm('Are you sure you
want to delete this employee?')">Delete</a>
                            </td>
                        </tr>
                    </s:iterator>
                </tbody>
            </table>
        </s:if>
        <s:else>
            <div class="alert alert-info">No employees found.</div>
        </s:else>
    </div>
</div>

<jsp:include page="footer.jsp" />

```

➤ **Struts_EMS/src/main /webapp/WEB-INF/views/header.jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Albin's Employee Management System</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.
css">
    <style>
        .error {
            color: red;
        }
        .message {
            color: green;
        }
    </style>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
        <a class="navbar-brand" href="<s:url
action='listEmployees'/>">Albin's Employee Management System</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link" href="<s:url
action='listEmployees'/>">View Employees</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="<s:url
action='addEmployeeForm'/>">Add Employee</a>
                </li>
            </ul>
        </div>
    </div>
</nav>
<div class="container mt-4">
<!-- Display action errors/messages -->
<s:if test="hasActionErrors()">
    <div class="alert alert-danger">
        <s:actionerror/>
    </div>
</s:if>
```

```
<s:if test="hasActionMessages()">
  <div class="alert alert-success">
    <s:actionmessage/>
  </div>
</s:if>
```


➤ **Struts_EMS//src/main/resources/struts.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">
<struts>
    <constant name="struts.devMode" value="true" />
    <constant name="struts.ui.theme" value="simple" />

    <package name="employee" extends="struts-default" namespace="/">

        <default-action-ref name="listEmployees" />
        <!-- Global Results -->
        <global-results>
            <result name="error">/WEB-INF/views/error.jsp</result>
        </global-results>

        <!-- Employee list action -->
        <action name="listEmployees"
class="com.example.action.EmployeeAction" method="list">
            <result name="success">/WEB-
INF/views/listEmployees.jsp</result>
        </action>

        <!-- Add employee form display action -->
        <action name="addEmployeeForm"
class="com.example.action.EmployeeAction" method="addForm">
            <result name="success">/WEB-INF/views/addEmployee.jsp</result>
        </action>

        <!-- Add employee action -->
        <action name="addEmployee"
class="com.example.action.EmployeeAction" method="add">
            <result name="success"
type="redirectAction">listEmployees</result>
            <result name="input">/WEB-INF/views/addEmployee.jsp</result>
        </action>

        <!-- Edit employee form display action -->
        <action name="editEmployeeForm"
class="com.example.action.EmployeeAction" method="editForm">
            <result name="success">/WEB-INF/views/editEmployee.jsp</result>
        </action>

        <!-- Update employee action -->
        <action name="updateEmployee"
class="com.example.action.EmployeeAction" method="update">
            <result name="success"
type="redirectAction">listEmployees</result>
            <result name="input">/WEB-INF/views/editEmployee.jsp</result>
        </action>
```

```

        <!-- Delete employee action -->
        <action name="deleteEmployee"
class="com.example.action.EmployeeAction" method="delete">
            <result name="success"
type="redirectAction">listEmployees</result>
        </action>
    </package>
</struts>

```

➤ Struts_EMS//src/main/resources/hibernate.cfg.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <!-- Database connection settings -->
        <property
name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/employee_db?createDatabaseIfNotExist=true</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password"></property>

        <!-- SQL dialect -->
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="hibernate.show_sql">true</property>

        <!-- Automatically update database schema -->
        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- Ensure proper session management -->
        <property
name="hibernate.current_session_context_class">thread</property>

        <!-- Names the annotated entity classes -->
        <mapping class="com.example.model.Employee"/>
    </session-factory>
</hibernate-configuration>

```

➤ **Struts_EMS /src/main/java/com/example/service/EmployeeService.java**

```
package com.example.service;

import com.example.model.Employee;
import java.util.List;

public interface EmployeeService {
    Long addEmployee(Employee employee);
    void updateEmployee(Employee employee);
    void deleteEmployee(Long id);
    Employee getEmployeeById(Long id);
    List<Employee> getAllEmployees();
    boolean isEmailUnique(String email, Long id);
}
```

➤ **Struts_EMS /src/main/java/com/example/service/EmployeeServiceImpl.java**

```
package com.example.service;

import com.example.dao.EmployeeDAO;
import com.example.dao.EmployeeDAOImpl;
import com.example.model.Employee;

import java.util.List;

public class EmployeeServiceImpl implements EmployeeService {

    private EmployeeDAO employeeDAO = new EmployeeDAOImpl();

    @Override
    public Long addEmployee(Employee employee) {
        return employeeDAO.addEmployee(employee);
    }

    @Override
    public void updateEmployee(Employee employee) {
        employeeDAO.updateEmployee(employee);
    }

    @Override
    public void deleteEmployee(Long id) {
        employeeDAO.deleteEmployee(id);
    }

    @Override
    public Employee getEmployeeById(Long id) {
        return employeeDAO.getEmployeeById(id);
    }

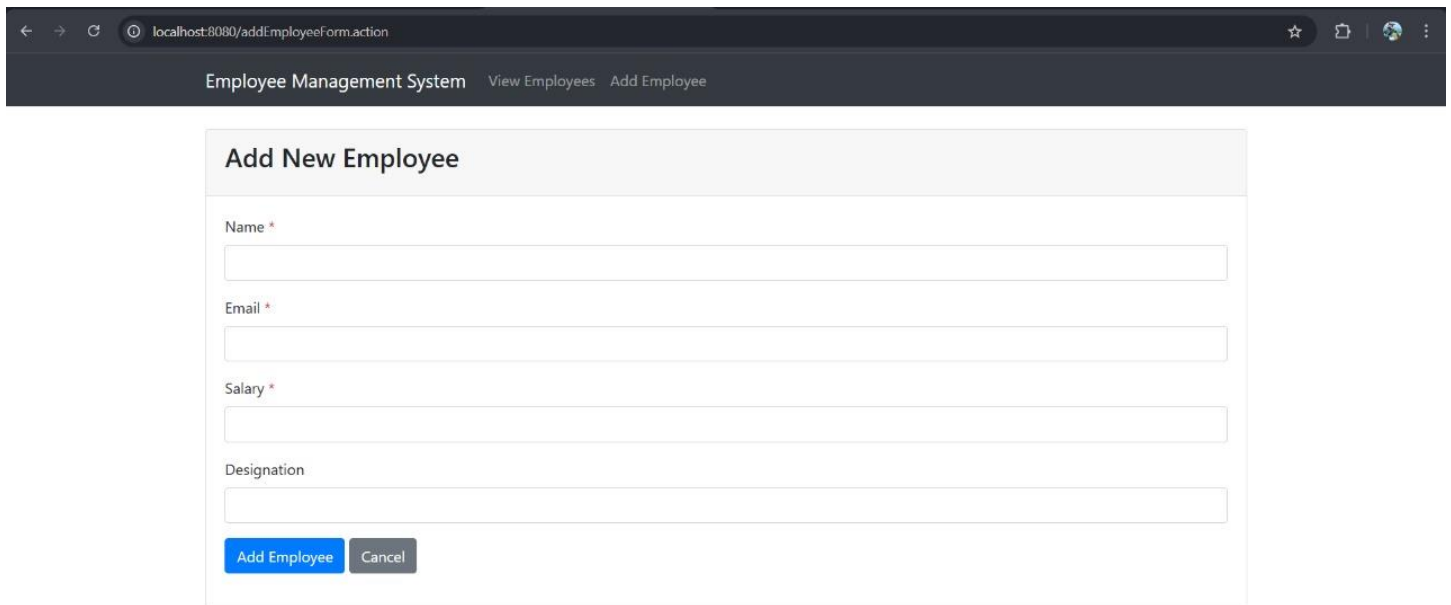
    @Override
    public List<Employee> getAllEmployees() {
        return employeeDAO.getAllEmployees();
    }
}
```

```
@Override
public boolean isEmailUnique(String email, Long id) {
    return employeeDAO.isEmailUnique(email, id);
}
```

```
}
```

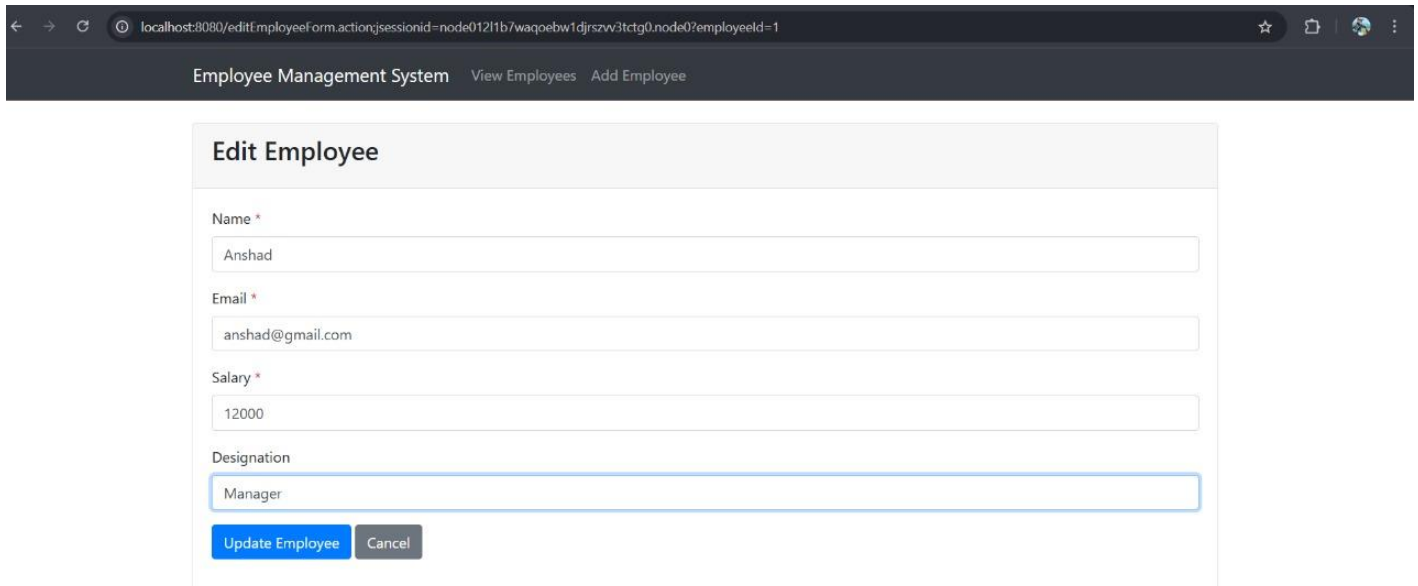
OUTPUT:

- Add Employee



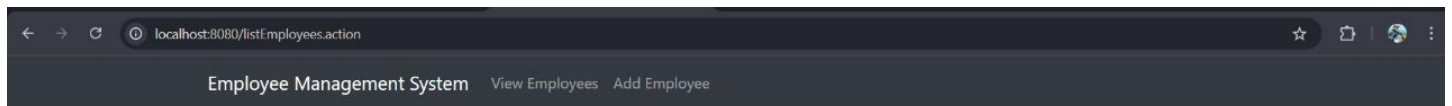
The screenshot shows a web browser window with the URL `localhost:8080/addEmployeeForm.action`. The browser's address bar and navigation icons are visible. Below the browser window, there is a dark navigation bar with the text "Employee Management System" and two links: "View Employees" and "Add Employee". The main content area displays a form titled "Add New Employee". The form contains four input fields: "Name *" (with a red asterisk), "Email *" (with a red asterisk), "Salary *" (with a red asterisk), and "Designation". Each field is empty. At the bottom of the form, there are two buttons: "Add Employee" (in blue) and "Cancel" (in grey).

- Update Employee



The screenshot shows a web browser window with the URL `localhost:8080/editEmployeeForm.action?sessionId=node01211b/waqoebw1djrsvv3tctg0,node0?employeeId=1`. The browser's address bar and navigation icons are visible. Below the browser window, there is a dark navigation bar with the text "Employee Management System" and two links: "View Employees" and "Add Employee". The main content area displays a form titled "Edit Employee". The form contains four input fields: "Name *" (with a red asterisk), "Email *" (with a red asterisk), "Salary *" (with a red asterisk), and "Designation". The "Name" field contains the text "Anshad", the "Email" field contains "anshad@gmail.com", the "Salary" field contains "12000", and the "Designation" field contains "Manager". At the bottom of the form, there are two buttons: "Update Employee" (in blue) and "Cancel" (in grey).

- **List Employees**



Employees						Add New Employee	
ID	Name	Email	Salary	Designation	Actions		
14	Anshad	anshad@gmail.com	\$10000	Software Engineer	Edit	Delete	
15	test	test@gmail.com	\$200000	CEO	Edit	Delete	