Web Essentials

From Code to Career

The **Web Essentials** Workshop is a hands-on, beginner-friendly program designed to introduce you to the core technologies of modern web development.  
 In this workshop, you’ll learn how websites are built from the ground up — from designing responsive user interfaces to connecting them with dynamic server-side systems.

You’ll gain both **frontend** and **backend** skills, learning how data flows between browsers, servers, and databases. By the end of the workshop, you’ll understand how real-world web applications work and how to create one yourself.

Technologies Covered

* **Frontend:** HTML5, CSS3, JavaScript
* **Backend:** JSP (Java Server Pages), MySQL Database
* **Tools:** Apache Tomcat, VS Code, MySQL Workbench

Key Takeaways

* Master the building blocks of web pages — HTML, CSS, and JavaScript
* Learn to design interactive and responsive user interfaces
* Understand how to create dynamic web content and handle user input
* Learn how servers work and how JSP connects front-end and back-end
* Get introduced to database connectivity using MySQL
* Experience the complete workflow — from coding to deployment
* Build your own mini web application project

Workshop Outcomes

By the end of this workshop, students will be able to:

* Develop complete **frontend interfaces** using HTML, CSS, and JavaScript
* Create **interactive web pages** that respond to user actions
* Connect their frontend with backend using **JSP and JDBC**
* Manage data storage and retrieval using **MySQL**
* Understand **how a full-stack web application works** end-to-end
* Confidently start building and hosting their own projects

CONTENTS

|  |  |  |
| --- | --- | --- |
| 1 | What is the Internet? |  |
| 2 | How a website is accessed |  |
| 3 | The Client-Server Model |  |
| 4 | The Flow of a Web Application |  |
| 5 | Frontend + Backend = Full Stack |  |
| 6 | Tools & Software Required |  |
| 7 | HTML — Hyper Text Markup Language |  |
| 8 | CSS – Cascading Style Sheets |  |
| 9 | JavaScript (JS) – The Brain of the Web |  |
| 10 | Summary – HTML, CSS, and JavaScript |  |
| 11 | Mini Project 1: Personal Portfolio Web Page |  |
| 12 | Mini Project 2: Login Page (Frontend Only) |  |
| 13 | Mini Project 3: Registration Page |  |
| 14 | Mini Project 4: Dashboard Page (CRUD Ready) |  |

JSP (Java Server Pages)

JSP is a server-side technology used to create dynamic web pages using Java. It allows embedding Java code into HTML pages using special JSP tags.

Why JSP?

* Easier than Servlets for UI-based web apps
* Supports Java directly inside HTML
* Enables separation of presentation (HTML) and business logic (Java)

Typical Uses

* Login and Registration forms
* Dynamic Dashboards
* CRUD Applications connected to Databases

How JSP Works

1. Browser sends an HTTP request to the server (like Tomcat).
2. The JSP page is converted into a Java servlet (by JSP engine).
3. The servlet is compiled into bytecode and executed by the JVM.
4. Output (usually HTML) is sent back to the browser.

Flow

Browser → Tomcat Server → JSP Engine → Servlet → Compiled Java → HTML

Tools & Software Setup

Before starting JSP development, ensure these tools are installed and configured properly.

|  |  |  |
| --- | --- | --- |
| Tool / Software | Purpose | Recommended Version / Notes |
| JDK (Java Development Kit) | To compile and run Java/JSP code | JDK 17 or higher |
| Apache Tomcat Server | Web server to deploy and run JSP/Servlet applications | Tomcat 9 or 10 |
| Eclipse IDE / IntelliJ IDEA | IDE for Java Web Projects | Eclipse Enterprise Edition preferred |
| MySQL Community Server | Database for CRUD operations | MySQL 8.0 |
| MySQL Workbench | GUI tool to manage MySQL visually | Optional but recommended |
| JDBC Connector (mysql-connector-j.jar) | Allows Java to connect with MySQL | Place inside /WEB-INF/lib/ |
| Web Browser (Chrome/Edge/Firefox) | To test your JSP pages | Latest version |
| Apache Maven (Optional) | For dependency management | Optional but useful for advanced builds |

Setup Steps

1. Install JDK - Download [Here](https://www.oracle.com/java/technologies/javase-downloads.html)
2. Add JAVA\_HOME environment variable.
3. Install Apache Tomcat - Download [Here](https://tomcat.apache.org/download-90.cgi)
4. Extract and configure Tomcat inside Eclipse:

*Window → Preferences → Server → Runtime Environments → Add Apache Tomcat.*

1. Install MySQL & Workbench
2. Create a database named jsp\_demo.
3. Inside Workbench, run:

*CREATE DATABASE jsp\_demo;*

*USE jsp\_demo;*

*CREATE TABLE users (*

*id INT PRIMARY KEY AUTO\_INCREMENT,*

*name VARCHAR(100),*

*email VARCHAR(100),*

*password VARCHAR(100)*

*);*

1. Add JDBC Driver
2. Copy mysql-connector-j-8.x.jar into:

WebContent/WEB-INF/lib/

1. Run JSP Project

In Eclipse → Right-click Project → Run As → Run on Server (select Tomcat).

JSP Elements

|  |  |  |
| --- | --- | --- |
| Type | Description | Syntax Example |
| Directives | Instructions to JSP container | *<%@ page language="java" %>* |
| Scripting Elements | Embed Java code | *<% int x = 10; %>* |
| Actions/Tags | Control behavior dynamically | *<jsp:useBean>* |
| Implicit Objects | Predefined objects (like request, session, out) | *<%= request.getParameter("name") %>* |

Scripting Elements

|  |  |  |
| --- | --- | --- |
| Element | Syntax | Description |
| Declaration | *<%! int count = 0; %>* | Declare variables/methods |
| Scriptlet | *<% out.println("Hello"); %>* | Write Java code |
| Expression | *<%= count %>* | Print value directly |

**Example:**

*<html>*

*<body>*

*<%! int num = 10; %>*

*<% num += 5; %>*

*<h3>Result is: <%= num %></h3>*

*</body>*

*</html>*

Directives

**Page Directive**

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

**Include Directive**

<%@ include file="header.jsp" %>

**Taglib Directive**

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

Actions / Tags

**Standard Action Tags**

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

<jsp:forward page="home.jsp" />

**JavaBeans Example**

<jsp:useBean id="user" class="com.demo.User" scope="session" />

<jsp:setProperty name="user" property="username" value="John" />

<jsp:getProperty name="user" property="username" />

JSP Life Cycle

1. Translation Phase – JSP → Servlet
2. Compilation Phase – Servlet → Class file
3. Initialization – jspInit()
4. Request Processing – \_jspService()
5. Destruction – jspDestroy()

How JSP is Compiled

index.jsp  →  index\_jsp.java  →  index\_jsp.class  →  executed

Tomcat handles this automatically inside:

/work/Catalina/localhost/YourApp/

Implicit Objects

|  |  |  |
| --- | --- | --- |
| Object | Description | Example JSP Code |
| request | Gets data from client | <%= request.getParameter("username") %> |
| response | Sends data to client | <% response.setContentType("text/html"); %> |
| session | Stores user session | <%= session.getAttribute("user") %> |
| application | Shared data across app | <%= application.getAttribute("siteName") %> |
| out | Writes output to browser | <% out.println("Welcome to JSP!"); %> |
| config | Servlet configuration | <%= config.getInitParameter("adminEmail") %> |
| pageContext | Access all namespaces | <%= pageContext.getRequest().getRemoteAddr() %> |

Exception Handling

Exception handling in JSP allows you to manage runtime errors gracefully.

* The directive *<%@ page errorPage="error.jsp" %>* specifies that if an exception occurs on the current page, control is transferred to *error.jsp*.
* In *error.jsp*, the directive *<%@ page isErrorPage="true" %>* enables access to the implicit exception object.
* *<%= exception.getMessage() %>* displays the actual error message that caused the exception.

JSP + JavaBeans

JSP and JavaBeans integration allows separation of business logic (JavaBean) from presentation (JSP).

User.java

*package com.demo;*

*public class User {*

*private String name;*

*public String getName() { return name; }*

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

*}*

In JSP:

*<jsp:useBean id="user" class="com.demo.User" />*

*<jsp:setProperty name="user" property="name" value="Chetan" />*

*Hello <jsp:getProperty name="user" property="name" />!*

* The JavaBean class User (in com.demo package) defines private properties with public getter and setter methods.
* In JSP, <jsp:useBean> creates or locates an instance of the bean.
* <jsp:setProperty> assigns a value to a bean property.
* <jsp:getProperty> retrieves and displays the value of a bean property.  
  This approach promotes code reusability, cleaner design, and easier maintenance.

JSP + Database Connectivity (JDBC + MySQL)

1. **Install MySQL:** Set up the MySQL server on your system.
2. **Create database jsp\_demo:** This database will store your application data (e.g., a user’s table).
3. **Add mysql-connector-j.jar to /lib/:** This JAR file is the MySQL JDBC driver that enables communication between JSP and the MySQL database.
4. **Using JDBC in JSP:**

*<%@ page import="java.sql.\*" %>*

*<%*

*Connection con = DriverManager.getConnection(*

*"jdbc:mysql://localhost:3306/jsp\_demo", "root", "password"*

*);*

*Statement st = con.createStatement();*

*ResultSet rs = st.executeQuery("SELECT \* FROM users");*

*while(rs.next()) {*

*out.println(rs.getString("name"));*

*}*

*con.close();*

*%>*

* **<%@ page import="java.sql.\*" %>** imports the JDBC classes.
* **DriverManager.getConnection()** establishes a connection to the database.
* **Statement** and **ResultSet** are used to execute SQL queries and process results.
* **out.println()** displays the retrieved data (e.g., user names) on the web page.
* **con.close()** closes the connection to free resources.

This approach allows dynamic data retrieval and displays from a MySQL database within a JSP page.

MVC Architecture in JSP (Model–View–Controller)

MVC (Model–View–Controller) is a software design pattern that separates an application into three main logical components:

* Model → Handles data and business logic
* View → Manages UI and presentation
* Controller → Controls the flow between Model and View

This separation makes the application more organized, reusable, and easier to maintain.

Why MVC?

|  |  |
| --- | --- |
| Problem (Without MVC) | Solution (With MVC) |
| HTML and Java code mixed inside JSP files | Separation of logic and presentation |
| Hard to maintain large projects | Easy to manage in separate layers |
| Code duplication and poor scalability | Reusable and scalable structure |
| Testing is difficult | Each layer can be tested individually |

MVC in JSP Workflow

Let’s break it down step-by-step:

1. User (Client) interacts with a web page (e.g., submits a login form).
2. The request goes to a Controller (Servlet).
3. The Controller communicates with the Model to process data (like accessing a database).
4. The Model sends the processed data (like user info) back to the Controller.
5. The Controller forwards data to the View (JSP) for displaying the result to the user.

MVC Flow Diagram

[Browser]

   ↓

[Controller - Servlet] ←→ [Model - Java + JDBC + Database]

   ↓

[View - JSP]

Model

* Represents the data and business logic of the application.
* It interacts with the database (JDBC, Hibernate, etc.).
* Usually implemented as JavaBeans or Plain Java Classes.

Example:

// User.java

*package com.demo.model;*

*public class User {*

*private int id;*

*private String name;*

*private String email;*

*// Getters & Setters*

*public int getId() { return id; }*

*public void setId(int 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; }*

*}*

View

* Represents the presentation layer (UI).
* Typically implemented using JSP, HTML, and CSS.
* Should not contain business logic.
* Displays only data provided by the Controller.

Example:

<!-- dashboard.jsp -->

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

*<html>*

*<head><title>User Dashboard</title></head>*

*<body>*

*<h2>Welcome, ${username}</h2>*

*<table border="1">*

*<tr><th>ID</th><th>Name</th><th>Email</th></tr>*

*<c:forEach var="u" items="${userList}">*

*<tr>*

*<td>${u.id}</td>*

*<td>${u.name}</td>*

*<td>${u.email}</td>*

*</tr>*

*</c:forEach>*

*</table>*

*</body>*

*</html>*

Controller

* Acts as the link between View and Model.
* Handles client requests, calls Model for data processing, and selects the appropriate View.
* Typically implemented using Servlets.

Example:

// UserController.java

*package com.demo.controller;*

*import com.demo.model.User;*

*import com.demo.dao.UserDAO;*

*import jakarta.servlet.\*;*

*import jakarta.servlet.http.\*;*

*import java.io.IOException;*

*import java.util.List;*

*public class UserController extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response)*

*throws ServletException, IOException {*

*UserDAO dao = new UserDAO();*

*List<User> list = dao.getAllUsers();*

*request.setAttribute("userList", list);*

*RequestDispatcher rd = request.getRequestDispatcher("dashboard.jsp");*

*rd.forward(request, response);*

*}*

*}*

Session Management in JSP

A session is a way to store information (data) about a user across multiple requests while they interact with a web application.

* HTTP is a stateless protocol — meaning, each page request is independent.
* To remember the user (like login state, shopping cart, etc.), we use Session Management.

How Sessions Work in Web Applications

* A user logs into a website — the server creates a session object for that user.
* A unique Session ID is generated and stored in the browser (usually as a cookie).
* On every request, the browser sends this Session ID back to the server.
* The server uses this ID to retrieve user-specific data from memory.

User Login  →  Server Creates Session (sessionId = X123)

                ↓

Browser stores sessionId in Cookie

                ↓

Next request → Server reads sessionId (X123)

                ↓

Identifies user → Returns personalized response

Why Use Sessions?

|  |  |
| --- | --- |
| Purpose | Example |
| Maintain user login | Remember who is logged in |
| Store temporary data | Shopping cart, preferences |
| Control user access | Prevent unauthorized access |
| Personalize content | Display “Welcome, John!” |

Session Management Techniques

|  |  |  |
| --- | --- | --- |
| Technique | Description | Use Case |
| Cookies | Data stored in browser | Small, non-sensitive info |
| URL Rewriting | Session ID passed in URL | When cookies disabled |
| Hidden Form Fields | Store session info in forms | Simple data persistence |
| HttpSession Object | Server-side session tracking | Most secure & recommended |

Common Methods

|  |  |
| --- | --- |
| Method | Description |
| session.setAttribute(name, value) | Store data in session |
| session.getAttribute(name) | Retrieve data from session |
| session.removeAttribute(name) | Remove a specific attribute |
| session.invalidate() | Destroy the entire session |
| session.getId() | Returns session ID |
| session.isNew() | Checks if it’s a new session |

Example 1: Create and Access Session in JSP

login.jsp

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

*<html>*

*<body>*

*<form action="welcome.jsp" method="post">*

*Username: <input type="text" name="user"><br>*

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

*</form>*

*</body>*

*</html>*

welcome.jsp

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

*<%*

*String user = request.getParameter("user");*

*session.setAttribute("username", user);*

*%>*

*<h2>Welcome, <%= session.getAttribute("username") %></h2>*

*<a href="profile.jsp">Go to Profile</a>*

profile.jsp

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

*<%*

*String user = (String) session.getAttribute("username");*

*if (user == null) {*

*response.sendRedirect("login.jsp");*

*}*

*%>*

*<h2>Hello again, <%= user %>!</h2>*

*<a href="logout.jsp">Logout</a>*

logout.jsp

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

*<%*

*session.invalidate();*

*response.sendRedirect("login.jsp");*

*%>*

Project: User Management System (Login, Register, Dashboard with CRUD, Logout)

* Frontend: HTML, CSS, JSP (View)
* Backend Logic: Java Servlet (Controller)
* Database: MySQL (Model Layer)
* Server: Apache Tomcat
* Architecture: MVC (Model-View-Controller)

Project Folder Structure

UserManagementMVC/

│

├── src/

│   ├── model/

│   │   ├── User.java

│   │   ├── UserDAO.java

│   ├── controller/

│   │   ├── UserServlet.java

│   │   ├── LoginServlet.java

│   │   ├── LogoutServlet.java

│

├── WebContent/

│   ├── index.jsp

│   ├── login.jsp

│   ├── register.jsp

│   ├── dashboard.jsp

│   ├── edit.jsp

│   ├── logout.jsp

│   ├── error.jsp

│   ├── WEB-INF/

│       └── web.xml

│

└── lib/ (MySQL Connector JAR)

⚙️ Tools and Software Required

|  |  |  |
| --- | --- | --- |
| Tool | Purpose | Download Link |
| VS Code / Eclipse IDE | For writing code | [**Eclipse**](https://www.eclipse.org/downloads/) |
| Tomcat 11 | Web server to run JSP/Servlet | [**Tomcat Download**](https://tomcat.apache.org/download-11.cgi) |
| Java 17+ | JDK to compile Java code | [**Java Downloads**](https://www.oracle.com/java/technologies/downloads/) |
| MySQL Community Server | Database | [**MySQL Download**](https://dev.mysql.com/downloads/mysql/) |
| MySQL Workbench | GUI for database | [**Workbench Download**](https://dev.mysql.com/downloads/workbench/) |
| MySQL Connector/J | JDBC driver | [**Connector/J Download**](https://dev.mysql.com/downloads/connector/j/) |

1. Database Setup (MySQL)

CREATE DATABASE userdb;

USE userdb;

CREATE TABLE users (

  id INT AUTO\_INCREMENT PRIMARY KEY,

  name VARCHAR(100),

  email VARCHAR(100),

  password VARCHAR(100)

);

INSERT INTO users (name, email, password) VALUES ('Admin', 'admin@gmail.com', 'admin123');

2. Model Layer

**User.java**

*package model;*

*public class User {*

*private int id;*

*private String name;*

*private String email;*

*private String password;*

*// Getters and Setters*

*public int getId() { return id; }*

*public void setId(int 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 String getPassword() { return password; }*

*public void setPassword(String password) { this.password = password; }*

*}*

*UserDAO.java*

*package model;*

*import java.sql.\*;*

*import java.util.\*;*

*public class UserDAO {*

*private String jdbcURL = "jdbc:mysql://localhost:3306/userdb";*

*private String jdbcUsername = "root";*

*private String jdbcPassword = "root"; // change your password*

*private String jdbcDriver = "com.mysql.cj.jdbc.Driver";*

*private static final String INSERT\_USER\_SQL = "INSERT INTO users (name,email,password) VALUES (?, ?, ?)";*

*private static final String SELECT\_USER\_BY\_ID = "SELECT id,name,email,password FROM users WHERE id=?";*

*private static final String SELECT\_ALL\_USERS = "SELECT \* FROM users";*

*private static final String DELETE\_USER\_SQL = "DELETE FROM users WHERE id=?";*

*private static final String UPDATE\_USER\_SQL = "UPDATE users SET name=?, email=?, password=? WHERE id=?";*

*protected Connection getConnection() {*

*Connection connection = null;*

*try {*

*Class.forName(jdbcDriver);*

*connection = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);*

*} catch (Exception e) { e.printStackTrace(); }*

*return connection;*

*}*

*public void insertUser(User user) {*

*try (Connection conn = getConnection();*

*PreparedStatement ps = conn.prepareStatement(INSERT\_USER\_SQL)) {*

*ps.setString(1, user.getName());*

*ps.setString(2, user.getEmail());*

*ps.setString(3, user.getPassword());*

*ps.executeUpdate();*

*} catch (Exception e) { e.printStackTrace(); }*

*}*

*public User validateUser(String email, String password) {*

*User user = null;*

*String sql = "SELECT \* FROM users WHERE email=? AND password=?";*

*try (Connection conn = getConnection();*

*PreparedStatement ps = conn.prepareStatement(sql)) {*

*ps.setString(1, email);*

*ps.setString(2, password);*

*ResultSet rs = ps.executeQuery();*

*if (rs.next()) {*

*user = new User();*

*user.setId(rs.getInt("id"));*

*user.setName(rs.getString("name"));*

*user.setEmail(rs.getString("email"));*

*}*

*} catch (Exception e) {*

*e.printStackTrace();*

*}*

*return user;*

*}*

*public User selectUser(int id) {*

*User user = null;*

*try (Connection conn = getConnection();*

*PreparedStatement ps = conn.prepareStatement(SELECT\_USER\_BY\_ID)) {*

*ps.setInt(1, id);*

*ResultSet rs = ps.executeQuery();*

*while (rs.next()) {*

*String name = rs.getString("name");*

*String email = rs.getString("email");*

*String password = rs.getString("password");*

*user = new User();*

*user.setId(id);*

*user.setName(name);*

*user.setEmail(email);*

*user.setPassword(password);*

*}*

*} catch (Exception e) { e.printStackTrace(); }*

*return user;*

*}*

*public List<User> selectAllUsers() {*

*List<User> users = new ArrayList<>();*

*try (Connection conn = getConnection();*

*PreparedStatement ps = conn.prepareStatement(SELECT\_ALL\_USERS)) {*

*ResultSet rs = ps.executeQuery();*

*while (rs.next()) {*

*User u = new User();*

*u.setId(rs.getInt("id"));*

*u.setName(rs.getString("name"));*

*u.setEmail(rs.getString("email"));*

*u.setPassword(rs.getString("password"));*

*users.add(u);*

*}*

*} catch (Exception e) { e.printStackTrace(); }*

*return users;*

*}*

*public boolean deleteUser(int id) throws SQLException {*

*boolean rowDeleted;*

*try (Connection conn = getConnection();*

*PreparedStatement ps = conn.prepareStatement(DELETE\_USER\_SQL)) {*

*ps.setInt(1, id);*

*rowDeleted = ps.executeUpdate() > 0;*

*}*

*return rowDeleted;*

*}*

*public boolean updateUser(User user) throws SQLException {*

*boolean rowUpdated;*

*try (Connection conn = getConnection();*

*PreparedStatement ps = conn.prepareStatement(UPDATE\_USER\_SQL)) {*

*ps.setString(1, user.getName());*

*ps.setString(2, user.getEmail());*

*ps.setString(3, user.getPassword());*

*ps.setInt(4, user.getId());*

*rowUpdated = ps.executeUpdate() > 0;*

*}*

*return rowUpdated;*

*}*

*}*

3. Controller Layer

*package controller;*

*import jakarta.servlet.\*;*

*import jakarta.servlet.http.\*;*

*import java.io.IOException;*

*import model.User;*

*import model.UserDAO;*

*public class LoginServlet extends HttpServlet {*

*private UserDAO userDAO;*

*public void init() {*

*userDAO = new UserDAO();*

*}*

*protected void doPost(HttpServletRequest request, HttpServletResponse response)*

*throws ServletException, IOException {*

*String email = request.getParameter("email");*

*String password = request.getParameter("password");*

*User user = userDAO.validateUser(email, password);*

*if (user != null) {*

*HttpSession session = request.getSession();*

*session.setAttribute("loggedUser", user);*

*response.sendRedirect("list");*

*} else {*

*request.setAttribute("errorMessage", "Invalid email or password!");*

*RequestDispatcher dispatcher = request.getRequestDispatcher("login.jsp");*

*dispatcher.forward(request, response);*

*}*

*}*

*}*

*package controller;*

*import jakarta.servlet.\*;*

*import jakarta.servlet.http.\*;*

*import java.io.IOException;*

*public class LogoutServlet extends HttpServlet {*

*protected void doGet(HttpServletRequest request, HttpServletResponse response)*

*throws ServletException, IOException {*

*HttpSession session = request.getSession(false);*

*if (session != null) {*

*session.invalidate();*

*}*

*response.sendRedirect("login.jsp");*

*}*

*}*

*UserServlet.java*

*package controller;*

*import model.\*;*

*import jakarta.servlet.\*;*

*import jakarta.servlet.http.\*;*

*import java.io.\*;*

*import java.sql.SQLException;*

*import java.util.\*;*

*public class UserServlet extends HttpServlet {*

*private UserDAO userDAO;*

*public void init() {*

*userDAO = new UserDAO();*

*}*

*protected void doPost(HttpServletRequest request, HttpServletResponse response)*

*throws ServletException, IOException {*

*doGet(request, response);*

*}*

*protected void doGet(HttpServletRequest request, HttpServletResponse response)*

*throws ServletException, IOException {*

*String action = request.getServletPath();*

*try {*

*switch (action) {*

*case "/new":*

*showNewForm(request, response);*

*break;*

*case "/insert":*

*insertUser(request, response);*

*break;*

*case "/delete":*

*deleteUser(request, response);*

*break;*

*case "/edit":*

*showEditForm(request, response);*

*break;*

*case "/update":*

*updateUser(request, response);*

*break;*

*default:*

*listUser(request, response);*

*break;*

*}*

*} catch (SQLException ex) {*

*throw new ServletException(ex);*

*}*

*}*

*private void listUser(HttpServletRequest request, HttpServletResponse response)*

*throws SQLException, IOException, ServletException {*

*List<User> listUser = userDAO.selectAllUsers();*

*request.setAttribute("listUser", listUser);*

*RequestDispatcher dispatcher = request.getRequestDispatcher("dashboard.jsp");*

*dispatcher.forward(request, response);*

*}*

*private void showNewForm(HttpServletRequest request, HttpServletResponse response)*

*throws ServletException, IOException {*

*RequestDispatcher dispatcher = request.getRequestDispatcher("register.jsp");*

*dispatcher.forward(request, response);*

*}*

*private void insertUser(HttpServletRequest request, HttpServletResponse response)*

*throws SQLException, IOException {*

*String name = request.getParameter("name");*

*String email = request.getParameter("email");*

*String password = request.getParameter("password");*

*User newUser = new User();*

*newUser.setName(name);*

*newUser.setEmail(email);*

*newUser.setPassword(password);*

*userDAO.insertUser(newUser);*

*response.sendRedirect("list");*

*}*

*private void showEditForm(HttpServletRequest request, HttpServletResponse response)*

*throws SQLException, ServletException, IOException {*

*int id = Integer.parseInt(request.getParameter("id"));*

*User existingUser = userDAO.selectUser(id);*

*RequestDispatcher dispatcher = request.getRequestDispatcher("edit.jsp");*

*request.setAttribute("user", existingUser);*

*dispatcher.forward(request, response);*

*}*

*private void updateUser(HttpServletRequest request, HttpServletResponse response)*

*throws SQLException, IOException {*

*int id = Integer.parseInt(request.getParameter("id"));*

*String name = request.getParameter("name");*

*String email = request.getParameter("email");*

*String password = request.getParameter("password");*

*User user = new User();*

*user.setId(id);*

*user.setName(name);*

*user.setEmail(email);*

*user.setPassword(password);*

*userDAO.updateUser(user);*

*response.sendRedirect("list");*

*}*

*private void deleteUser(HttpServletRequest request, HttpServletResponse response)*

*throws SQLException, IOException {*

*int id = Integer.parseInt(request.getParameter("id"));*

*userDAO.deleteUser(id);*

*response.sendRedirect("list");*

*}*

*}*

4. JSP Pages (View Layer)

**register.jsp**

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

*<h2>Register</h2>*

*Name: <input type="text" name="name"><br>*

*Email: <input type="email" name="email"><br>*

*Password: <input type="password" name="password"><br>*

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

*</form>*

*<a href="list">View Users</a>*

**index.jsp**

*<!DOCTYPE html>*

*<html>*

*<head><title>Home - User Management</title></head>*

*<body>*

*<h2>Welcome to User Management System</h2>*

*<a href="login.jsp">Login</a> |*

*<a href="register.jsp">Register</a>*

*</body>*

*</html>*

**login.jsp**

*<!DOCTYPE html>*

*<html>*

*<head><title>Login</title></head>*

*<body>*

*<h2>User Login</h2>*

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

*Email: <input type="email" name="email" required><br><br>*

*Password: <input type="password" name="password" required><br><br>*

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

*</form>*

*<p style="color:red;">*

*${errorMessage}*

*</p>*

*<a href="register.jsp">New user? Register here</a>*

*</body>*

*</html>*

**dashboard.jsp** (Add session check + Logout)

*<%@ page import="java.util.List" %>*

*<%@ page import="model.User" %>*

*<%*

*User loggedUser = (User) session.getAttribute("loggedUser");*

*if (loggedUser == null) {*

*response.sendRedirect("login.jsp");*

*return;*

*}*

*%>*

*<h2>Welcome, <%= loggedUser.getName() %>!</h2>*

*<a href="new">Add New User</a> |*

*<a href="logout">Logout</a>*

*<table border="1" cellpadding="5" cellspacing="0">*

*<tr><th>ID</th><th>Name</th><th>Email</th><th>Actions</th></tr>*

*<%*

*List<User> listUser = (List<User>) request.getAttribute("listUser");*

*if (listUser != null) {*

*for (User u : listUser) {*

*%>*

*<tr>*

*<td><%=u.getId()%></td>*

*<td><%=u.getName()%></td>*

*<td><%=u.getEmail()%></td>*

*<td>*

*<a href="edit?id=<%=u.getId()%>">Edit</a> |*

*<a href="delete?id=<%=u.getId()%>">Delete</a>*

*</td>*

*</tr>*

*<% } } %>*

*</table>*

**logout.jsp**

*<%*

*session.invalidate();*

*response.sendRedirect("login.jsp");*

*%>*

**edit.jsp**

*<h2>Edit User</h2>*

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

*<input type="hidden" name="id" value="${user.id}">*

*Name: <input type="text" name="name" value="${user.name}"><br>*

*Email: <input type="email" name="email" value="${user.email}"><br>*

*Password: <input type="password" name="password" value="${user.password}"><br>*

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

*</form>*

5. web.xml Configuration

*<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee" version="5.0">*

*<servlet>*

*<servlet-name>UserServlet</servlet-name>*

*<servlet-class>controller.UserServlet</servlet-class>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>UserServlet</servlet-name>*

*<url-pattern>/</url-pattern>*

*</servlet-mapping>*

*<servlet>*

*<servlet-name>LoginServlet</servlet-name>*

*<servlet-class>controller.LoginServlet</servlet-class>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>LoginServlet</servlet-name>*

*<url-pattern>/login</url-pattern>*

*</servlet-mapping>*

*<servlet>*

*<servlet-name>LogoutServlet</servlet-name>*

*<servlet-class>controller.LogoutServlet</servlet-class>*

*</servlet>*

*<servlet-mapping>*

*<servlet-name>LogoutServlet</servlet-name>*

*<url-pattern>/logout</url-pattern>*

*</servlet-mapping>*

*</web-app>*