**Practical No. 1**

**Title -: Understand about different Website design issues.**

**Code -:**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Website Evaluation Case Study</title>**

**<style>**

**body {**

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

**margin: 20px;**

**}**

**h1 {**

**color: #333;**

**}**

**table {**

**width: 100%;**

**border-collapse: collapse;**

**margin-top: 20px;**

**}**

**th, td {**

**border: 1px solid #ddd;**

**padding: 8px;**

**text-align: left;**

**}**

**th {**

**background-color: #f2f2f2;**

**}**

**tr:nth-child(even) {**

**background-color: #f9f9f9;**

**}**

**.good {**

**background-color: #e6ffe6;**

**}**

**.bad {**

**background-color: #ffe6e6;**

**}**

**</style>**

**</head>**

**<body>**

**<h1>Website Evaluation Case Study</h1>**

**<table>**

**<thead>**

**<tr>**

**<th>Sr. No.</th>**

**<th>Website</th>**

**<th>URL</th>**

**<th>Purpose of Website</th>**

**<th>Things Liked in the Website</th>**

**<th>Things Disliked in the Website</th>**

**<th>Overall Evaluation</th>**

**</tr>**

**</thead>**

**<tbody>**

**<tr class="good">**

**<td>1</td>**

**<td>Amazon</td>**

**<td>amazon.com</td>**

**<td>E-commerce & product sales</td>**

**<td>Huge selection, fast checkout</td>**

**<td>Cluttered homepage, too many ads</td>**

**<td>Good</td>**

**</tr>**

**<tr class="good">**

**<td>2</td>**

**<td>BBC News</td>**

**<td>bbc.com/news</td>**

**<td>News & information</td>**

**<td>Clean layout, reliable content</td>**

**<td>Autoplay videos, paywall for some articles</td>**

**<td>Good</td>**

**</tr>**

**<tr class="bad">**

**<td>3</td>**

**<td>Craigslist</td>**

**<td>craigslist.org</td>**

**<td>Classified ads & local listings</td>**

**<td>Simple, free to use</td>**

**<td>Outdated design, hard to navigate</td>**

**<td>Bad</td>**

**</tr>**

**<tr class="good">**

**<td>4</td>**

**<td>Tesla</td>**

**<td>tesla.com</td>**

**<td>Car sales & brand promotion</td>**

**<td>Sleek design, interactive features</td>**

**<td>Slow loading on some pages</td>**

**<td>Good</td>**

**</tr>**

**<tr class="bad">**

**<td>5</td>**

**<td>Geocities (Archive)</td>**

**<td>geocities.restorativland.org</td>**

**<td>Retro web design archive</td>**

**<td>Nostalgic, historical value</td>**

**<td>Extremely outdated, broken layouts</td>**

**<td>Bad (but intentional)</td>**

**</tr>**

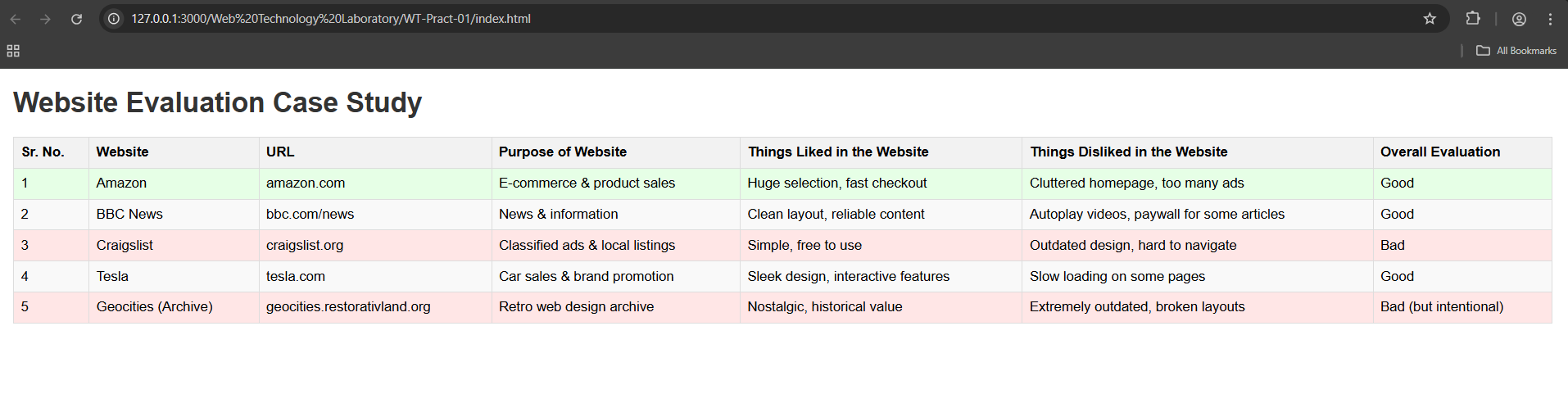
**</tbody>**

**</table>**

**</body>**

**</html>**

**Output -:**

****

**Practical No. 2**

**Title -: Implement a web page index.htm for any client website (e.g., a restaurant website**

**Project) using following: a. HTML syntax: heading tags, basic tags and attributes,**

**Frames, tables, images, lists, links for text and images, forms etc. b. Use of Internal**

**CSS, Inline CSS, External CSS**

**Code -:**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Tasty Bites</title>**

**<style>**

**body {font-family:Arial; margin:0; padding:0;}**

**header {background:#8B0000; color:white; padding:20px; text-align:center;}**

**nav {background:#333; overflow:hidden;}**

**nav a {float:left; color:white; padding:14px 16px; text-decoration:none;}**

**.main {float:left; width:70%; padding:20px;}**

**.sidebar {float:right; width:25%; padding:20px; background:#e9e9e9;}**

**footer {background:#333; color:white; text-align:center; padding:10px; clear:both;}**

**@media (max-width: 600px) {**

**.sidebar {display:left;}**

**}**

**table {width:100%; border-collapse:collapse;}**

**th, td {border:1px solid #ddd; padding:8px;}**

**</style>**

**</head>**

**<body>**

**<header>**

**<h1>Tasty Bites</h1>**

**<h2>Local Family Restaurant</h2>**

**</header>**

**<nav>**

**<a href="#home">Home</a>**

**<a href="#menu">Menu</a>**

**<a href="#about">About</a>**

**<a href="#contact">Contact</a>**

**</nav>**

**<h3>Our Menu</h3>**

**<table>**

**<tr><th>Item</th><th>Price</th></tr>**

**<tr><td>Burger</td><td>$9.99</td></tr>**

**<tr><td>Pizza</td><td>$12.99</td></tr>**

**<tr><td>Salad</td><td>$7.99</td></tr>**

**</table>**

**</div>**

**<footer>**

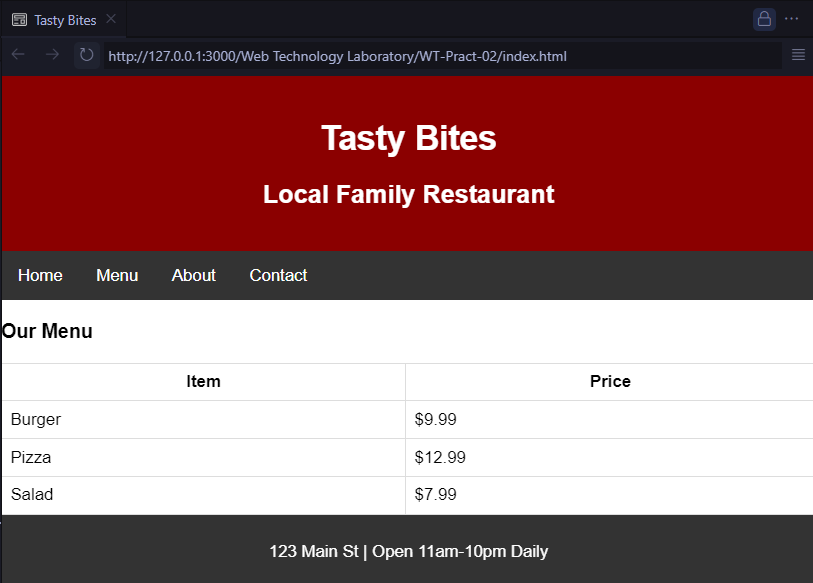
**<p>123 Main St | Open 11am-10pm Daily</p>**

**</footer>**

**</body>**

**</html>**

**Output -:**

****

**Practical No. 3**

**Title -: Design the XML document to store the information of the employees of any business organization and demonstrate the use of:**

**a) DTD b) XML Schema And display the content in (e.g., tabular format) by using CSS/XSL**

**Code -:**

**<?xml version="1.0" encoding="UTF-8"?>**

**<!DOCTYPE employees SYSTEM "employees.dtd">**

**<?xml-stylesheet type="text/xsl" href="employees.xsl"?>**

**<?xml-stylesheet type="text/css" href="employees.css"?>**

**<employees xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

**xsi:noNamespaceSchemaLocation="employees.xsd">**

**<employee id="E1001">**

**<name>Akhilesh Jadhav</name>**

**<position>Software Developer</position>**

**<department>IT</department>**

**<salary>75000</salary>**

**<hire\_date>2018-06-15</hire\_date>**

**</employee>**

**<employee id="E1002">**

**<name>Sarah Johnson</name>**

**<position>HR Manager</position>**

**<department>Human Resources</department>**

**<salary>68000</salary>**

**<hire\_date>2019-03-22</hire\_date>**

**</employee>**

**<employee id="E1003">**

**<name>Vishwajit Kogade</name>**

**<position>Sales Executive</position>**

**<department>Marketing</department>**

**<salary>62000</salary>**

**<hire\_date>2020-11-05</hire\_date>**

**</employee>**

**</employees>**

**<?xml version="1.0" encoding="UTF-8"?>**

**<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">**

**<xsl:template match="/">**

**<html>**

**<head>**

**<title>Employee Information</title>**

**<style>**

**table { width:100%; border-collapse:collapse; }**

**th, td { border:1px solid #ddd; padding:8px; text-align:left; }**

**th { background-color:#8B0000; color:white; }**

**tr:nth-child(even) { background-color:#f2f2f2; }**

**</style>**

**</head>**

**<body>**

**<h2>Employee Directory</h2>**

**<table>**

**<tr>**

**<th>ID</th>**

**<th>Name</th>**

**<th>Position</th>**

**<th>Department</th>**

**<th>Salary</th>**

**<th>Hire Date</th>**

**</tr>**

**<xsl:for-each select="employees/employee">**

**<tr>**

**<td><xsl:value-of select="@id"/></td>**

**<td><xsl:value-of select="name"/></td>**

**<td><xsl:value-of select="position"/></td>**

**<td><xsl:value-of select="department"/></td>**

**<td>$<xsl:value-of select="salary"/></td>**

**<td><xsl:value-of select="hire\_date"/></td>**

**</tr>**

**</xsl:for-each>**

**</table>**

**</body>**

**</html>**

**</xsl:template>**

**</xsl:stylesheet>**

**<?xml version="1.0" encoding="UTF-8"?>**

**<!DOCTYPE employees SYSTEM "employees.dtd">**

**<?xml-stylesheet type="text/xsl" href="employees.xsl"?>**

**<?xml-stylesheet type="text/css" href="employees.css"?>**

**<employees xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

**xsi:noNamespaceSchemaLocation="employees.xsd">**

**<employee id="E1001">**

**<name>John Smith</name>**

**<position>Software Developer</position>**

**<department>IT</department>**

**<salary>75000</salary>**

**<hire\_date>2018-06-15</hire\_date>**

**</employee>**

**<employee id="E1002">**

**<name>Sarah Johnson</name>**

**<position>HR Manager</position>**

**<department>Human Resources</department>**

**<salary>68000</salary>**

**<hire\_date>2019-03-22</hire\_date>**

**</employee>**

**<employee id="E1003">**

**<name>Michael Brown</name>**

**<position>Sales Executive</position>**

**<department>Marketing</department>**

**<salary>62000</salary>**

**<hire\_date>2020-11-05</hire\_date>**

**</employee>**

**</employees>**

**<?xml version="1.0" encoding="UTF-8"?>**

**<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">**

**<xs:element name="employees">**

**<xs:complexType>**

**<xs:sequence>**

**<xs:element name="employee" maxOccurs="unbounded">**

**<xs:complexType>**

**<xs:sequence>**

**<xs:element name="name" type="xs:string"/>**

**<xs:element name="position" type="xs:string"/>**

**<xs:element name="department" type="xs:string"/>**

**<xs:element name="salary" type="xs:integer"/>**

**<xs:element name="hire\_date" type="xs:date"/>**

**</xs:sequence>**

**<xs:attribute name="id" type="xs:string" use="required"/>**

**</xs:complexType>**

**</xs:element>**

**</xs:sequence>**

**</xs:complexType>**

**</xs:element>**

**</xs:schema>**

**<!ELEMENT employees (employee+)>**

**<!ATTLIST employees**

**xmlns:xsi CDATA #IMPLIED**

**xsi:noNamespaceSchemaLocation CDATA #IMPLIED>**

**<!ELEMENT employee (name, position, department, salary, hire\_date)>**

**<!ATTLIST employee id CDATA #REQUIRED>**

**<!ELEMENT name (#PCDATA)>**

**<!ELEMENT position (#PCDATA)>**

**<!ELEMENT department (#PCDATA)>**

**<!ELEMENT salary (#PCDATA)>**

**<!ELEMENT hire\_date (#PCDATA)>**

**Output -:**

****

**Practical No. 4**

**Title -: Implement an application in Java Script using following:**

**a) Design UI of application using HTML, CSS etc. b) Include JavaScript validation**

**number etc. c) Use of prompt and alert window using JavaScript e.g., Design and implement a simple calculator using JavaScript for operations like addition, Multiplication, subtraction, division, square of b) Validate input values a) Design calculator interface like text field for input and output, buttons for numbers and Operators etc. c) Promptalerts for invalid values etc.**

**Code -:**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Calculator</title>**

**<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">**

**<style>**

**button{**

**height: 50px;**

**width: 50px;**

**}**

**</style>**

**</head>**

**<body>**

**<h1 class="text-center">Calculator</h1>**

**<main style="  max-width: 300px; margin: auto; padding: 10px;">**

**<div class="container">**

**<input id="display" class="w-100 bg-dark text-white" readonly>**

**<div class="d-flex">**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('1')">1</button>**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('2')">2</button>**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('3')">3</button>**

**<button class="btn bg-success text-white m-2" onclick="apDisplay('+')">+</button>**

**</div>**

**<div class="d-flex">**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('4')">4</button>**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('5')">5</button>**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('6')">6</button>**

**<button class="btn bg-success text-white m-2" onclick="apDisplay('-')">-</button>**

**</div>**

**<div class="d-flex">**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('7')">7</button>**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('8')">8</button>**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('9')">9</button>**

**<button class="btn bg-success text-white m-2" onclick="apDisplay('\*')">\*</button>**

**</div>**

**<div class="d-flex">**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('.')">.</button>**

**<button class="btn bg-dark text-white m-2" onclick="apDisplay('0')">0</button>**

**<button class="btn bg-dark text-white m-2" onclick="calculate()">=</button>**

**<button class="btn bg-success text-white m-2" onclick="apDisplay('/')">/</button>**

**</div>**

**<button class="btn bg-dark text-white m-2 w-50" onclick="clearDisplay()">Clear</button>**

**</div>**

**</main>**

**<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>**

**<script>**

**document.addEventListener('keydown', function(event) {**

**if (event.key === 'Delete') {**

**clearDisplay();**

**}**

**if (event.key === 'Backspace') {**

**minusDisplay();**

**}**

**if (event.key === 'Enter') {**

**calculate();**

**}**

**if (event.key >= '0' && event.key <= '9') {**

**apDisplay(event.key);**

**}**

**if (event.key === '+' || event.key === '-' || event.key === '\*' || event.key === '/') {**

**apDisplay(event.key);**

**}**

**if (event.key === '.') {**

**apDisplay(event.key);**

**}**

**});**

***const* display = document.getElementById('display');**

**function apDisplay(input){**

**display.value += input;**

**}**

**function clearDisplay(){**

**display.value = "";**

**}**

**function minusDisplay(){**

**display.value = display.value.slice(0, -1);**

**}**

**function calculate(){**

**try{**

**display.value = eval(display.value);**

**}**

**catch(error){**

**display.value = "Error";**

**}**

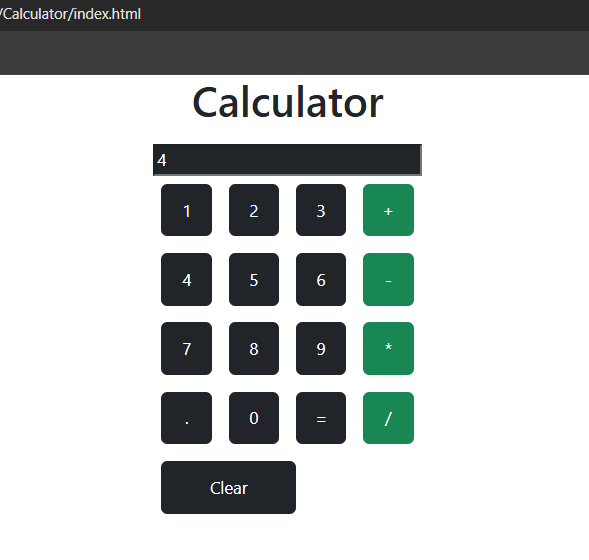
**}**

**</script>**

**</body>**

**</html>**

**Output -:**

****

**Practical No. 5**

**Title -: Implement the sample program demonstrating the use of Servlet. e.g., create a database table eBook shop (book\_id, book\_ title, book\_author, book\_price, quantity) using database like Oracle/MySQL etc. and display (use SQL select query) the table content using servlet.**

**Code -:**

**import *java.io.\**;**

**import *java.sql.\**;**

**import *javax.servlet.\**;**

**import *javax.servlet.http.\**;**

***public* *class* BookServlet *extends* HttpServlet {**

***private* *static* *final* String JDBC\_URL = "jdbc:mysql://localhost:3306/ebookshop";**

***private* *static* *final* String JDBC\_USER = "your\_username";**

***private* *static* *final* String JDBC\_PASSWORD = "your\_password";**

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

***throws* ServletException, IOException {**

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

**PrintWriter out = response.getWriter();**

**try {**

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

**Connection conn = DriverManager.getConnection(JDBC\_URL, JDBC\_USER, JDBC\_PASSWORD);**

**Statement stmt = conn.createStatement();**

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

**out.println("<html><head><title>eBook Shop</title>");**

**out.println("<style>");**

**out.println("table { border-collapse: collapse; width: 80%; margin: 20px auto; }");**

**out.println("th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }");**

**out.println("th { background-color: #4CAF50; color: white; }");**

**out.println("tr:nth-child(even) { background-color: #f2f2f2; }");**

**out.println("h1 { text-align: center; color: #333; }");**

**out.println("</style></head>");**

**out.println("<body><h1>eBook Shop Inventory</h1>");**

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

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

**while (rs.next()) {**

**int id = rs.getInt("book\_id");**

**String title = rs.getString("book\_title");**

**String author = rs.getString("book\_author");**

**double price = rs.getDouble("book\_price");**

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

**out.println("<tr>");**

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

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

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

**out.println("<td>$" + price + "</td>");**

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

**out.println("</tr>");**

**}**

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

**rs.close();**

**stmt.close();**

**conn.close();**

**} catch (ClassNotFoundException e) {**

**out.println("<p>Error: JDBC driver not found</p>");**

**e.printStackTrace(out);**

**} catch (SQLException e) {**

**out.println("<p>Error: Database connection problem</p>");**

**e.printStackTrace(out);**

**}**

**}**

**}**

**<?xml version="1.0" encoding="UTF-8"?>**

**<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"**

**xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

**xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee**

**http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"**

**version="4.0">**

**<servlet>**

**<servlet-name>BookServlet</servlet-name>**

**<servlet-class>BookServlet</servlet-class>**

**</servlet>**

**<servlet-mapping>**

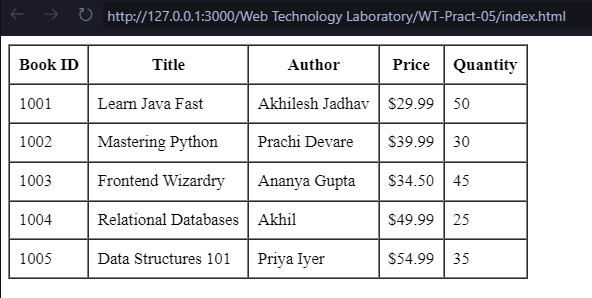
**<servlet-name>BookServlet</servlet-name>**

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

**</servlet-mapping>**

**</web-app>**

**Output -:**

****

**Practical No. 6**

**Title -:** **Implement the program demonstrating the use of JSP. e.g., Create a database table students\_info (stud\_id, stud \_name, class, division, city) using database like Oracle/MySQL etc. and display (use SQL select query) the table content using JSP.**

**Code -:**

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

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

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Student Information System</title>**

**<style>**

**body {**

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

**margin: 0;**

**padding: 20px;**

**background-color: #f5f5f5;**

**}**

**.container {**

**max-width: 1000px;**

**margin: 0 auto;**

**background-color: white;**

**padding: 20px;**

**border-radius: 8px;**

**box-shadow: 0 0 10px rgba(0,0,0,0.1);**

**}**

**h1 {**

**color: #333;**

**text-align: center;**

**margin-bottom: 30px;**

**}**

**table {**

**width: 100%;**

**border-collapse: collapse;**

**margin-top: 20px;**

**}**

**th, td {**

**border: 1px solid #ddd;**

**padding: 12px;**

**text-align: left;**

**}**

**th {**

**background-color: #4CAF50;**

**color: white;**

**}**

**tr:nth-child(even) {**

**background-color: #f2f2f2;**

**}**

**.error {**

**color: red;**

**text-align: center;**

**margin: 20px 0;**

**}**

**</style>**

**</head>**

**<body>**

**<div class="container">**

**<h1>Student Information</h1>**

**<%**

**// Database connection details**

**String jdbcUrl = "jdbc:mysql://localhost:3306/student\_db";**

**String dbUser = "your\_username";**

**String dbPassword = "your\_password";**

**Connection conn = null;**

**Statement stmt = null;**

**ResultSet rs = null;**

**try {**

**// Load JDBC driver**

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

**// Establish connection**

**conn = DriverManager.getConnection(jdbcUrl, dbUser, dbPassword);**

**// Execute query**

**stmt = conn.createStatement();**

**String sql = "SELECT \* FROM students\_info";**

**rs = stmt.executeQuery(sql);**

**%>**

**<table>**

**<tr>**

**<th>Student ID</th>**

**<th>Name</th>**

**<th>Class</th>**

**<th>Division</th>**

**<th>City</th>**

**</tr>**

**<%**

**// Process result set**

**while (rs.next()) {**

**int studId = rs.getInt("stud\_id");**

**String studName = rs.getString("stud\_name");**

**String studClass = rs.getString("class");**

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

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

**%>**

**<tr>**

**<td><%= studId %></td>**

**<td><%= studName %></td>**

**<td><%= studClass %></td>**

**<td><%= division %></td>**

**<td><%= city %></td>**

**</tr>**

**<%**

**}**

**%>**

**</table>**

**<%**

**} catch (ClassNotFoundException e) {**

**%>**

**<p class="error">Error: MySQL JDBC Driver not found!</p>**

**<%**

**e.printStackTrace();**

**} catch (SQLException e) {**

**%>**

**<p class="error">Error: Database connection problem!</p>**

**<%**

**e.printStackTrace();**

**} finally {**

**// Close resources**

**try {**

**if (rs != null) rs.close();**

**if (stmt != null) stmt.close();**

**if (conn != null) conn.close();**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

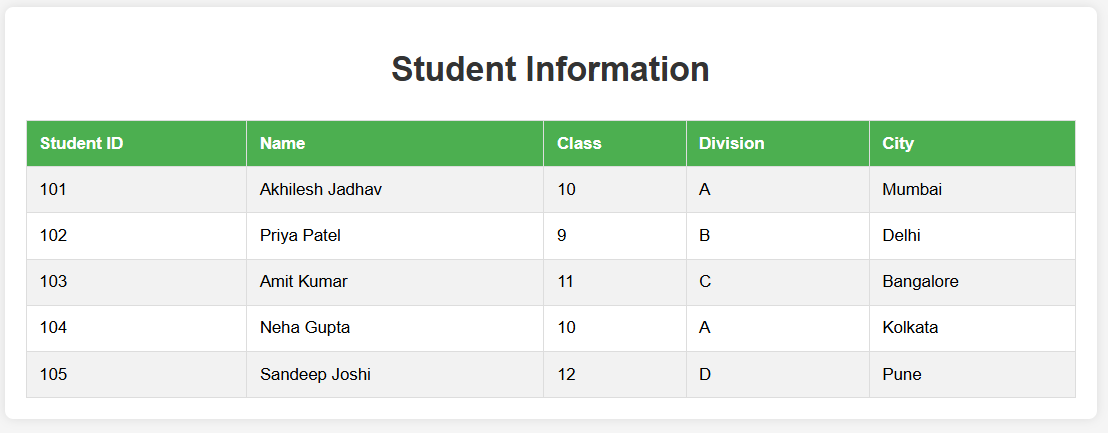
**}**

**%>**

**</div>**

**</body>**

**</html>**

**Output -:**

**Practical No. 7**

**Title -:** **Build a dynamic web application using PHP and MySQL.**

**a. Create database tables in MySQL and create connection with PHP.**

**b. Create the add, update, delete and retrieve functions in the PHP web app interacting with MySQL**

**Code -:**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="utf-8">**

**<title>Register</title>**

**<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.min.css" integrity="sha512-xh6O/CkQoPOWDdYTDqeRdPCVd1SpvCA9XXcUnZS2FmJNp1coAFzvtCN9BmamE+4aHK8yyUHUSCcJHgXloTyT2A==" crossorigin="anonymous" referrerpolicy="no-referrer">**

**<link href="style.css" rel="stylesheet" type="text/css">**

**</head>**

**<body>**

**<div class="register">**

**<h1>Register</h1>**

**<form action="register.php" method="post" autocomplete="off">**

**<label for="username">**

**<i class="fas fa-user"></i>**

**</label>**

**<input type="text" name="username" placeholder="Username" id="username" required>**

**<label for="password">**

**<i class="fas fa-lock"></i>**

**</label>**

**<input type="password" name="password" placeholder="Password" id="password" required>**

**<label for="email">**

**<i class="fas fa-envelope"></i>**

**</label>**

**<input type="email" name="email" placeholder="Email" id="email" required>**

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

**</form>**

**</div>**

**</body>**

**</html>**

**<?php**

**require\_once 'connection.php';**

**if (!isset($\_POST['username'], $\_POST['password'], $\_POST['email'])) {**

**exit('Please complete the registration form!');**

**}**

**if (empty($\_POST['username']) || empty($\_POST['password']) || empty($\_POST['email'])) {**

**exit('Please complete the registration form');**

**}**

**if ($stmt = $con->prepare('INSERT INTO accounts (username, password, email) VALUES (?, ?, ?)')) {**

**$stmt->bind\_param('sss', $\_POST['username'], $\_POST['password'], $\_POST['email']);**

**$stmt->execute();**

**echo 'You have successfully registered! You can now login!';**

**} else {**

**echo 'Could not prepare statement!';**

**}**

**<?php**

**$DATABASE\_HOST = 'localhost';**

**$DATABASE\_USER = 'root';**

**$DATABASE\_PASS = '';**

**$DATABASE\_NAME = 'phplogin';**

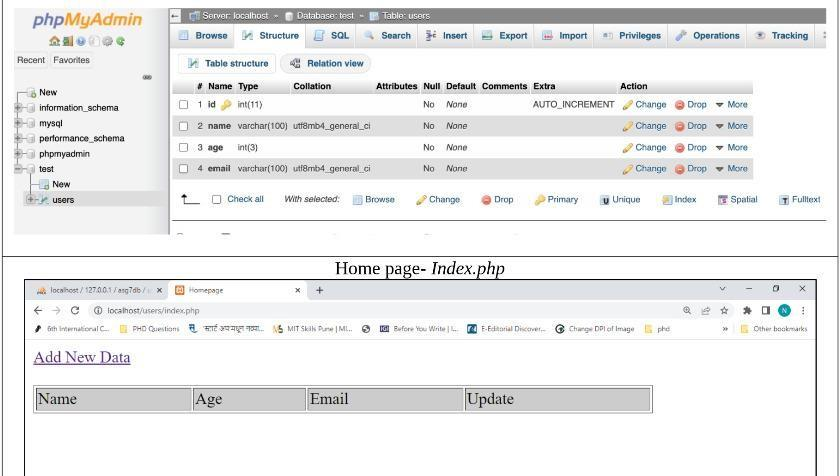
**$con = mysqli\_connect($DATABASE\_HOST, $DATABASE\_USER, $DATABASE\_PASS, $DATABASE\_NAME);**

**if (mysqli\_connect\_errno()) {**

**exit('Failed to connect to MySQL: ' . mysqli\_connect\_error());**

**}**

**Output -:**





**Practical No. 8**

**Title -:** **Design a login page with entries for name, mobile number email id and login button. Use struts and pertom tollowing validations b. Validation for mobile numbers c. Validation for enmail id d. Validation if no cntered any value e. Re-display for wrongly entered values with message f. Congratulations and welcome page upon successful entries**

**Code -:**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>JavaScript Form Validation Demo</title>**

**<style>**

**.container {**

**max-width: 400px;**

**margin: 0 auto;**

**padding: 20px;**

**}**

**.form-field {**

**margin-bottom: 15px;**

**}**

**.form-field label {**

**display: block;**

**margin-bottom: 5px;**

**}**

**.form-field input {**

**width: 100%;**

**padding: 8px;**

**}**

**.form-field small {**

**color: red;**

**}**

**.error input {**

**border-color: red;**

**}**

**.success input {**

**border-color: green;**

**}**

**</style>**

**</head>**

**<body>**

**<div class="container">**

**<form id="signup" class="form" action='Hello.jsp'>**

**<h1>Sign Up</h1>**

**<div class="form-field">**

**<label for="username">Username:</label>**

**<input type="text" name="username" id="username" autocomplete="off">**

**<small></small>**

**</div>**

**<div class="form-field">**

**<label for="email">Email:</label>**

**<input type="email" name="email" id="email" autocomplete="off">**

**<small></small>**

**</div>**

**<div class="form-field">**

**<label for="mobile">Mobile</label>**

**<input type="number" name="mobile" id="mobile" autocomplete="off">**

**<small></small>**

**</div>**

**<input type="submit" value="Sign Up">**

**</form>**

**</div>**

**<script>**

***const* usernameEl = document.querySelector('#username');**

***const* emailEl = document.querySelector('#email');**

***const* mobileEl = document.querySelector('#mobile');**

***const* form = document.querySelector('#signup');**

***const* checkUsername = () => {**

***let* valid = false;**

***const* min = 3, max = 25;**

***const* username = usernameEl.value.trim();**

**if (!isRequired(username)) {**

**showError(usernameEl, 'Username cannot be blank.');**

**} else if (!isBetween(username.length, min, max)) {**

**showError(usernameEl, `Username must be between ${min} and ${max} characters.`);**

**} else {**

**showSuccess(usernameEl);**

**valid = true;**

**}**

***return* valid;**

**};**

***const* checkEmail = () => {**

***let* valid = false;**

***const* email = emailEl.value.trim();**

**if (!isRequired(email)) {**

**showError(emailEl, 'Email cannot be blank.');**

**} else if (!isEmailValid(email)) {**

**showError(emailEl, 'Email is not valid.');**

**} else {**

**showSuccess(emailEl);**

**valid = true;**

**}**

***return* valid;**

**};**

***const* checkMobile = () => {**

***let* valid = false;**

***const* mobile = mobileEl.value.trim();**

**if (!isRequired(mobile)) {**

**showError(mobileEl, 'Mobile cannot be blank.');**

**} else if (!isMobileValid(mobile)) {**

**showError(mobileEl, 'Mobile must be 10 digits.');**

**} else {**

**showSuccess(mobileEl);**

**valid = true;**

**}**

***return* valid;**

**};**

***const* isEmailValid = (email) => {**

***const* re = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;**

***return* re.test(email);**

**};**

***const* isMobileValid = (mobile) => {**

***const* re = /^[0-9]{10}$/;**

***return* re.test(mobile);**

**};**

***const* isRequired = value => value === '' ? false : true;**

***const* isBetween = (length, min, max) => length < min || length > max ? false : true;**

***const* showError = (input, message) => {**

***const* formField = input.parentElement;**

**formField.classList.remove('success');**

**formField.classList.add('error');**

***const* error = formField.querySelector('small');**

**error.textContent = message;**

**};**

***const* showSuccess = (input) => {**

***const* formField = input.parentElement;**

**formField.classList.remove('error');**

**formField.classList.add('success');**

***const* error = formField.querySelector('small');**

**error.textContent = '';**

**};**

**form.addEventListener('submit', function(e) {**

**e.preventDefault();**

***let* isUsernameValid = checkUsername(),**

**isEmailValid = checkEmail(),**

**isMobileValid = checkMobile();**

***let* isFormValid = isUsernameValid && isEmailValid && isMobileValid;**

**if (isFormValid) {**

**form.submit();**

**}**

**});**

***const* debounce = (fn, delay = 500) => {**

***let* timeoutId;**

***return* (...args) => {**

**if (timeoutId) {**

**clearTimeout(timeoutId);**

**}**

**timeoutId = setTimeout(() => {**

**fn.apply(null, args)**

**}, delay);**

**};**

**};**

**form.addEventListener('input', debounce(function(e) {**

**switch (e.target.id) {**

**case 'username':**

**checkUsername();**

**break;**

**case 'email':**

**checkEmail();**

**break;**

**case 'mobile':**

**checkMobile();**

**break;**

**}**

**}));**

**</script>**

**</body>**

**</html>**

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

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Hello World</title>**

**</head>**

**<body>**

**<%**

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

**String Mobile = request.getParameter("mobile");**

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

**session.setAttribute("name", name);**

**session.setAttribute("mobile", Mobile);**

**session.setAttribute("email", Email);**

**%>**

**Hi!!! My name is <% out.println(session.getAttribute("name")); %><br /><br>**

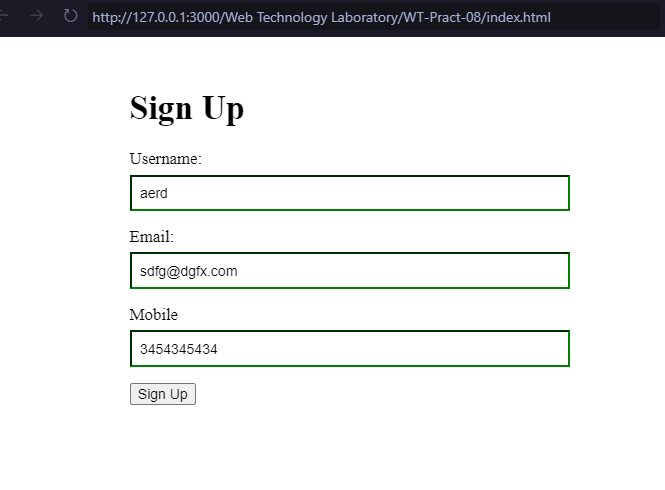
**Mobile Number is:- <% out.println(session.getAttribute("mobile")); %><br><br>**

**Email id:- <% out.println(session.getAttribute("email")); %><br><br>**

**</body>**

**</html>**

**Output -:**



**Practical No. 9**

**Title -:** **Design an application using Angular JS. e-g., Design registration (first name, last name, username, password) and login page using Angular JS.**

**Code -:**

**<!DOCTYPE html>**

**<html ng-app="myApp">**

**<head>**

**<meta charset="utf-8">**

**<title>AngularJS Registration & Login</title>**

**<link rel="stylesheet" href="style.css">**

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>**

**<script src="app.js"></script>**

**</head>**

**<body ng-controller="myCtrl">**

**<h1><b>REGISTRATION & LOGIN</b></h1>**

**<h2>Register</h2>**

**<form ng-submit="registerUser()">**

**<label for="firstName">First Name:</label>**

**<input type="text" ng-model="firstName" required><br>**

**<br>**

**<label for="lastName">Last Name:</label>**

**<input type="text" ng-model="lastName" required><br>**

**<br>**

**<label for="username">Username:</label>**

**<input type="text" ng-model="username" required><br>**

**<br>**

**<label for="password">Password:</label>**

**<input type="password" ng-model="password" required><br>**

**<br>**

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

**</form>**

**<h2>Login</h2>**

**<form ng-submit="loginUser()">**

**<label for="username">Username:</label>**

**<input type="text" ng-model="username" required><br>**

**<br>**

**<label for="password">Password:</label>**

**<input type="password" ng-model="password" required><br>**

**<br>**

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

**</form>**

**<div ng-if="errorMessage">{{errorMessage}}</div>**

**<div ng-if="successMessage">{{successMessage}}</div>**

**</body>**

**</html>**

***var* app = angular.module("myApp", []);**

**app.controller("myCtrl", function($scope) {**

**$scope.registerUser = function() {**

***var* user = {**

**firstName: $scope.firstName,**

**lastName: $scope.lastName,**

**username: $scope.username,**

**password: $scope.password**

**};**

**localStorage.setItem($scope.username, JSON.stringify(user));**

**$scope.firstName = "";**

**$scope.lastName = "";**

**$scope.username = "";**

**$scope.password = "";**

**$scope.successMessage = "Registration successful!";**

**};**

**$scope.loginUser = function() {**

***var* user = JSON.parse(localStorage.getItem($scope.username));**

**if (user && user.password === $scope.password) {**

**$scope.username = "";**

**$scope.password = "";**

**$scope.successMessage = "Login successful!";**

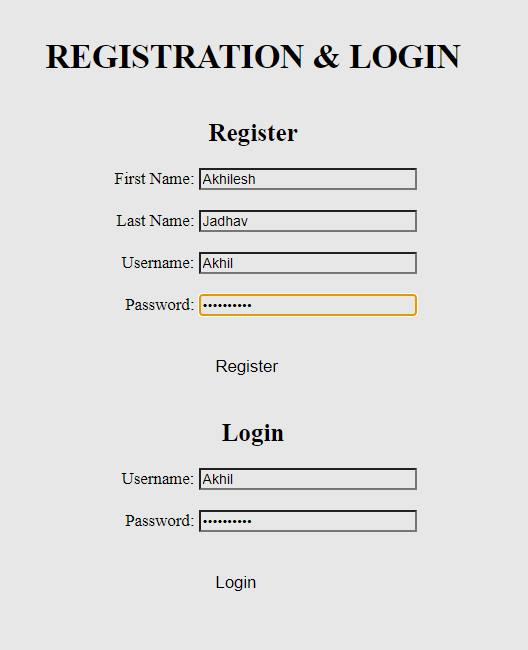
**} else {**

**$scope.errorMessage = "Invalid username or password";**

**}**

**};**

**});**

**Output -:**

**Practical No. 10**

**Title -:** **Design and implement a business interface with necessary business logic for any web application using EJB. e.g., Design and implement the web application logic for deposit and withdraw amount transactions using EJB.**

**Code -:**

**package com.banking.ejb;**

**import com.banking.model.Account;**

**import javax.ejb.Stateless;**

**import javax.persistence.EntityManager;**

**import javax.persistence.PersistenceContext;**

**@Stateless**

**public class AccountServiceImpl implements AccountService {**

**@PersistenceContext(unitName = "BankingPU")**

**private EntityManager em;**

**@Override**

**public Account createAccount(String accountNumber, String accountHolder, double initialDeposit) {**

**Account account = new Account(accountNumber, accountHolder, initialDeposit);**

**em.persist(account);**

**return account;**

**}**

**@Override**

**public Account findAccount(String accountNumber) {**

**return em.find(Account.class, accountNumber);**

**}**

**@Override**

**public Account deposit(String accountNumber, double amount) throws TransactionException {**

**if (amount <= 0) {**

**throw new TransactionException("Deposit amount must be positive");**

**}**

**Account account = findAccount(accountNumber);**

**if (account == null) {**

**throw new TransactionException("Account not found");**

**}**

**account.setBalance(account.getBalance() + amount);**

**return em.merge(account);**

**}**

**@Override**

**public Account withdraw(String accountNumber, double amount) throws TransactionException {**

**if (amount <= 0) {**

**throw new TransactionException("Withdrawal amount must be positive");**

**}**

**Account account = findAccount(accountNumber);**

**if (account == null) {**

**throw new TransactionException("Account not found");**

**}**

**if (account.getBalance() < amount) {**

**throw new TransactionException("Insufficient funds");**

**}**

**account.setBalance(account.getBalance() - amount);**

**return em.merge(account);**

**}**

**}**

**package com.banking.web;**

**import com.banking.ejb.AccountService;**

**import com.banking.ejb.TransactionException;**

**import com.banking.model.Account;**

**import java.io.IOException;**

**import javax.ejb.EJB;**

**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("/account")**

**public class AccountServlet extends HttpServlet {**

**@EJB**

**private AccountService accountService;**

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

**throws ServletException, IOException {**

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

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

**try {**

**switch (action) {**

**case "deposit":**

**double depositAmount = Double.parseDouble(request.getParameter("amount"));**

**Account depositedAccount = accountService.deposit(accountNumber, depositAmount);**

**request.setAttribute("account", depositedAccount);**

**request.setAttribute("message", "Deposit successful!");**

**break;**

**case "withdraw":**

**double withdrawAmount = Double.parseDouble(request.getParameter("amount"));**

**Account withdrawnAccount = accountService.withdraw(accountNumber, withdrawAmount);**

**request.setAttribute("account", withdrawnAccount);**

**request.setAttribute("message", "Withdrawal successful!");**

**break;**

**case "find":**

**Account foundAccount = accountService.findAccount(accountNumber);**

**if (foundAccount != null) {**

**request.setAttribute("account", foundAccount);**

**} else {**

**request.setAttribute("error", "Account not found");**

**}**

**break;**

**}**

**} catch (TransactionException e) {**

**request.setAttribute("error", e.getMessage());**

**} catch (NumberFormatException e) {**

**request.setAttribute("error", "Invalid amount format");**

**}**

**request.getRequestDispatcher("/index.jsp").forward(request, response);**

**}**

**}**

**<?xml version="1.0" encoding="UTF-8"?>**

**<project xmlns="http://maven.apache.org/POM/4.0.0"**

**xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

**xsi:schemaLocation="http://maven.apache.org/POM/4.0.0**

**http://maven.apache.org/xsd/maven-4.0.0.xsd">**

**<modelVersion>4.0.0</modelVersion>**

**<groupId>com.banking</groupId>**

**<artifactId>BankingApp</artifactId>**

**<version>1.0</version>**

**<packaging>war</packaging>**

**<properties>**

**<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>**

**<maven.compiler.source>1.8</maven.compiler.source>**

**<maven.compiler.target>1.8</maven.compiler.target>**

**</properties>**

**<dependencies>**

**<!-- Java EE 8 API -->**

**<dependency>**

**<groupId>javax</groupId>**

**<artifactId>javaee-api</artifactId>**

**<version>8.0</version>**

**<scope>provided</scope>**

**</dependency>**

**<!-- JSTL -->**

**<dependency>**

**<groupId>javax.servlet</groupId>**

**<artifactId>jstl</artifactId>**

**<version>1.2</version>**

**</dependency>**

**</dependencies>**

**<build>**

**<finalName>BankingApp</finalName>**

**</build>**

**</project>**

**Output -:**

