UNIVERSITY SCHOOL OF INFORMATION COMMUNICATION AND TECHNOLOGY

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY 2024



# Next Generation Web Programming ( ICT-423P )

**Lab File**

**Submitted to**: **Submitted by:**

Ms. Yashima Hooda Name – Pranav Jain

Course – B.Tech. (CSE)

Enrollment No. – 00516403221

**Index**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Experiment** | **Remarks** |
| 1) | To create an html le to link to different html page which contains images, tables, and also link within a page. |  |
| 2) | Write a program to create an html le by applying the different styles using inline, external & internal style sheets. |  |
| 3) | Write a program to display the calendar using javascript code by getting the year from the user |  |
| 4) | Write a program to create a html registration form and to validate the form using javascript code. |  |
| 5) | Write a program for implementing XML document for STUDENTDETAILS. |  |
| 6) | Write a program to display contents of XML file in a table using Extensible Style Sheets. |  |
| 7) | Write a servlet that reads parameters from employee login page |  |
| 8) | Write a program to create a single page website using Node.js and Express.js. |  |

**Experiment-1**

**Aim**:. To create an html file to link to different html page which contains images, tables, and also link within a page.

**Code**

HTML

// index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Main Page</title>

</head>

<body>

<h1>Welcome to My Web Page</h1>

<nav>

<ul>

<li><a href="images.html">Go to Images Page</a></li>

<li><a href="table.html">Go to Table Page</a></li>

<li><a href="#section2">Go to Section 2 (Internal Link)</a></li>

</ul>

</nav>

<section id="section1">

<h2>Section 1</h2>

<p>This is the first section of the main page.</p>

</section>

<section id="section2">

<h2>Section 2</h2>

<p>This is the second section of the main page. It contains an internal link.</p>

</section>

</body>

</html>

//images.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Images Page</title>

</head>

<body>

<h1>Images Page</h1>

<p><a href="index.html">Back to Main Page</a></p>

<section>

<h2>Image Gallery</h2>

<img src="image1.jpg" alt="Sample Image 1" width="200">

<img src="image1.jpg" alt="Sample Image 2" width="200">

<img src="image1.jpg" alt="Sample Image 3" width="200">

</section>

<p><a href="#top">Back to Top</a></p>

</body>

</html>

//table.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Table Page</title>

</head>

<body>

<h1>Table Page</h1>

<p><a href="index.html">Back to Main Page</a></p>

<section>

<h2>Sample Table</h2>

<table border="1">

<tr>

<th>Name</th>

<th>Age</th>

<th>City</th>

</tr>

<tr>

<td>Alice</td>

<td>24</td>

<td>New York</td>

</tr>

<tr>

<td>Bob</td>

<td>30</td>

<td>Los Angeles</td>

</tr>

<tr>

<td>Charlie</td>

<td>28</td>

<td>Chicago</td>

</tr>

</table>

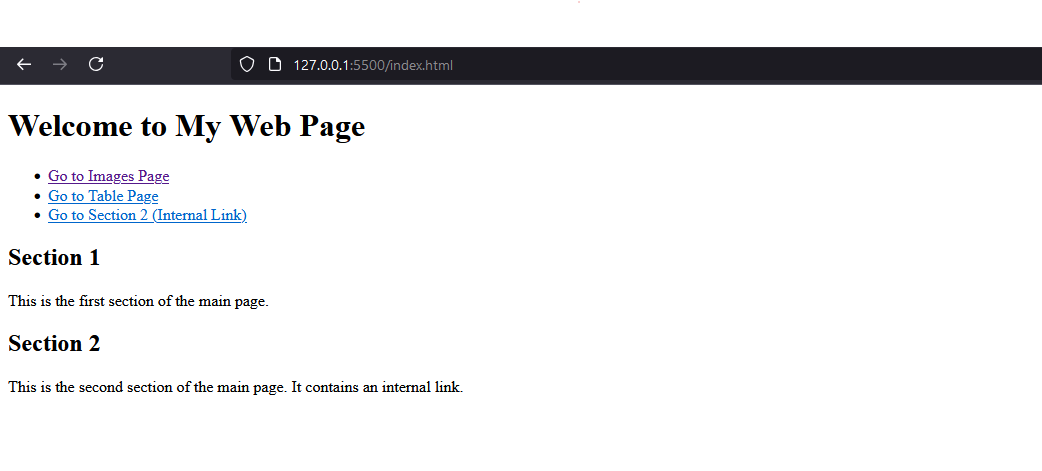
</section>

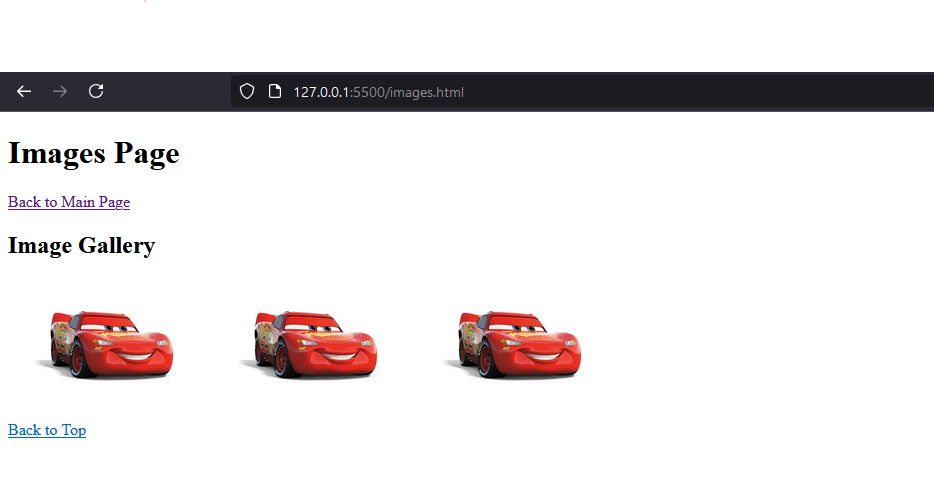
<p><a href="#top">Back to Top</a></p>

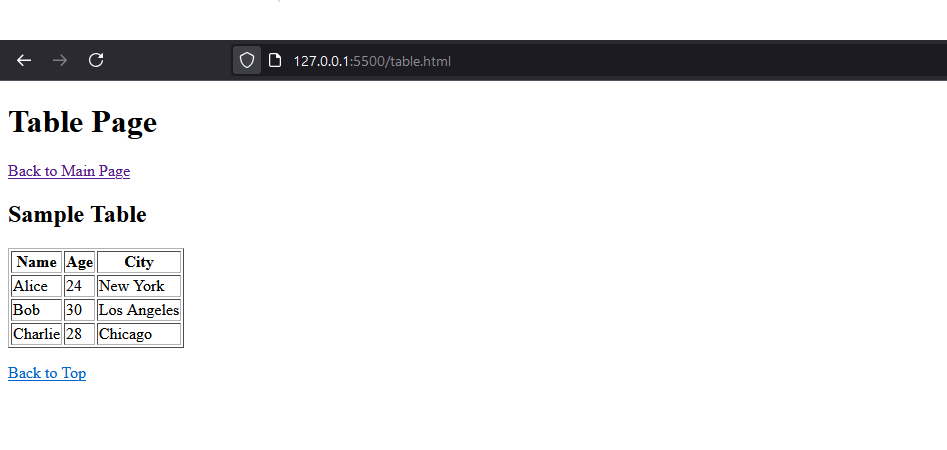
</body>

</html>

**Output**







**Experiment-2**

**Aim**:. Write a program to create an html file by applying the different styles using inline, external & internal style sheets.

**Code**

HTML

// index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Styled Page</title>

<!-- Internal Styles -->

<style>

body {

font-family: Arial, sans-serif;

background-color: #f0f8ff;

}

.container {

max-width: 800px;

margin: auto;

padding: 20px;

}

h1 {

color: #4682b4;

text-align: center;

}

.internal-paragraph {

color: #2e8b57;

font-size: 1.1em;

text-align: justify;

}

</style>

<!-- Link to External CSS -->

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="container">

<!-- Inline Style Example -->

<h1 style="font-size: 2em; text-shadow: 1px 1px 2px #aaa;">Welcome to the Styled Page</h1>

<p class="internal-paragraph">

This paragraph is styled using internal CSS. Internal CSS allows us to define styles for the entire HTML document within the &lt;style&gt; tag inside the &lt;head&gt; section.

</p>

<p class="external-paragraph">

This paragraph is styled using external CSS. External styles are applied from a separate `.css` file and can be reused across multiple HTML documents.

</p>

<p style="color: #ff4500; font-weight: bold;">

This paragraph uses inline styling. Inline styles are directly applied to elements and override both internal and external styles if there are conflicts.

</p>

</div>

</body>

</html>

//style.css

/\* External Styles for styled\_page.html \*/

body {

margin: 0;

padding: 0;

line-height: 1.6;

}

.external-paragraph {

color: #8b0000;

font-size: 1.2em;

margin-top: 15px;

text-align: justify;

}

.container {

border: 2px solid #4682b4;

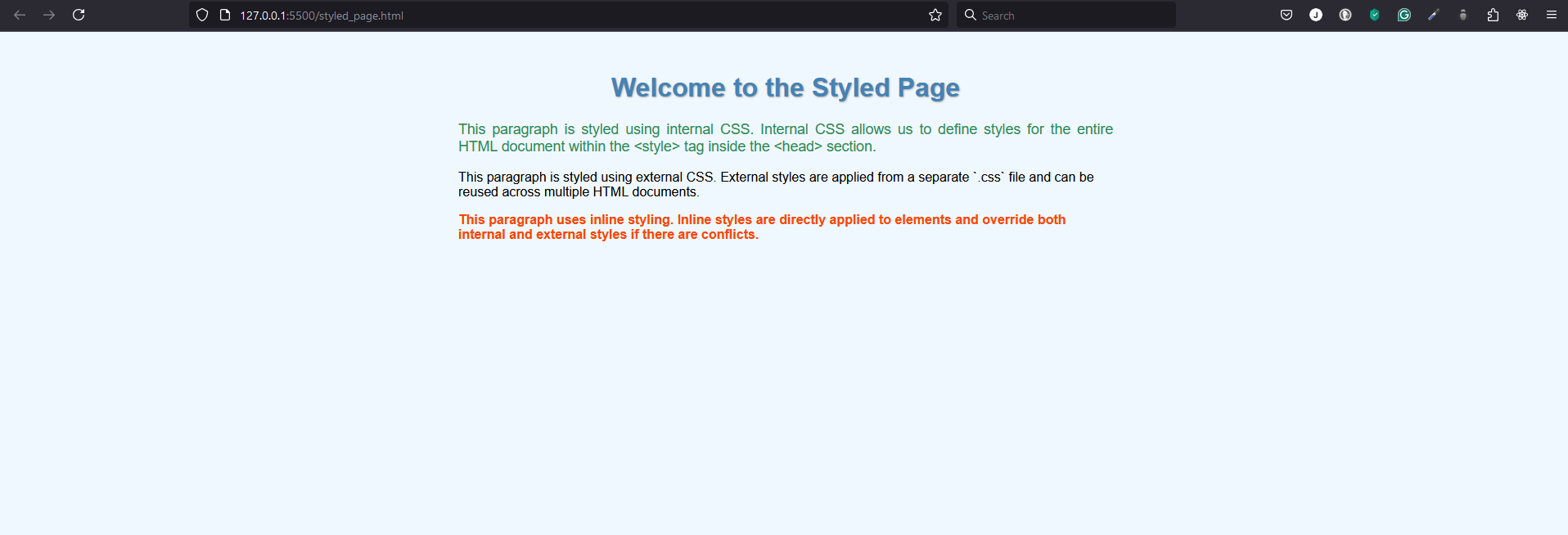
border-radius: 10px;

background-color: #ffffff;

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

}

**Output**



**Experiment-3**

**Aim**:. Write a program to display the calendar using javascript code by getting the year from the user.

**Code**

function isLeapYear(year) {

return (year % 4 === 0 && year % 100 !== 0) || (year % 400 === 0);

}

function getDaysInMonth(year, month) {

return new Date(year, month + 1, 0).getDate();

}

function generateCalendar(year) {

const months = [

"January", "February", "March", "April", "May", "June",

"July", "August", "September", "October", "November", "December"

];

const daysOfWeek = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"];

console.log(`Calendar for the year ${year}\n`);

months.forEach((month, monthIndex) => {

console.log(`\n${month}`);

console.log(daysOfWeek.join(" "));

const daysInMonth = getDaysInMonth(year, monthIndex);

let firstDay = new Date(year, monthIndex, 1).getDay();

// Print initial spaces for the first row

let calendarRow = "";

for (let i = 0; i < firstDay; i++) {

calendarRow += " ";

}

// Print days of the month

for (let day = 1; day <= daysInMonth; day++) {

calendarRow += day.toString().padStart(3, " ") + " ";

if ((firstDay + day) % 7 === 0 || day === daysInMonth) {

console.log(calendarRow);

calendarRow = "";

}

}

});

}

// Get year from user

const year = parseInt(prompt("Enter a year:"), 10);

if (!isNaN(year)) {

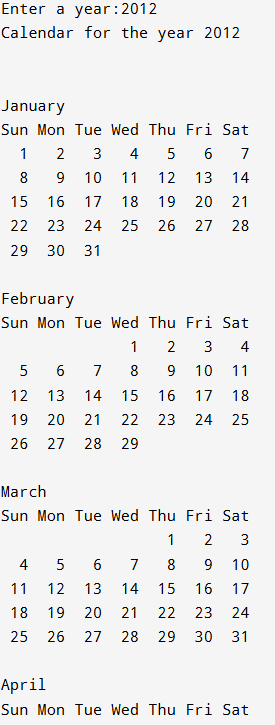
generateCalendar(year);

} else {

console.log("Invalid year input");

}

**Output**

**A calendar with numbers and letters

Description automatically generated**

**Experiment-4**

**Aim** Write a program to create a html registration form and to validate the form using javascript code.

**Code**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Registration Form</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.form-container {

max-width: 400px;

padding: 20px;

border: 1px solid #ccc;

border-radius: 8px;

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

}

.form-container h2 {

text-align: center;

}

.form-group {

margin-bottom: 15px;

}

.form-group label {

display: block;

margin-bottom: 5px;

}

.form-group input {

width: 100%;

padding: 8px;

border: 1px solid #ccc;

border-radius: 4px;

}

.error-message {

color: red;

font-size: 0.9em;

}

button {

width: 100%;

padding: 10px;

background-color: #28a745;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 16px;

}

button:hover {

background-color: #218838;

}

</style>

</head>

<body>

<div class="form-container">

<h2>Registration Form</h2>

<form id="registrationForm" onsubmit="return validateForm()">

<div class="form-group">

<label for="name">Full Name</label>

<input type="text" id="name" name="name">

<span class="error-message" id="nameError"></span>

</div>

<div class="form-group">

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

<input type="email" id="email" name="email">

<span class="error-message" id="emailError"></span>

</div>

<div class="form-group">

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

<input type="password" id="password" name="password">

<span class="error-message" id="passwordError"></span>

</div>

<div class="form-group">

<label for="confirmPassword">Confirm Password</label>

<input type="password" id="confirmPassword" name="confirmPassword">

<span class="error-message" id="confirmPasswordError"></span>

</div>

<button type="submit">Register</button>

</form>

</div>

<script>

function validateForm() {

// Clear any previous error messages

document.getElementById("nameError").innerText = "";

document.getElementById("emailError").innerText = "";

document.getElementById("passwordError").innerText = "";

document.getElementById("confirmPasswordError").innerText = "";

// Get form values

const name = document.getElementById("name").value.trim();

const email = document.getElementById("email").value.trim();

const password = document.getElementById("password").value.trim();

const confirmPassword = document.getElementById("confirmPassword").value.trim();

let isValid = true;

// Validate name

if (name === "") {

document.getElementById("nameError").innerText = "Name is required";

isValid = false;

}

// Validate email format

const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (email === "") {

document.getElementById("emailError").innerText = "Email is required";

isValid = false;

} else if (!emailPattern.test(email)) {

document.getElementById("emailError").innerText = "Please enter a valid email";

isValid = false;

}

// Validate password length

if (password === "") {

document.getElementById("passwordError").innerText = "Password is required";

isValid = false;

} else if (password.length < 6) {

document.getElementById("passwordError").innerText = "Password must be at least 6 characters long";

isValid = false;

}

// Validate confirm password matches password

if (confirmPassword === "") {

document.getElementById("confirmPasswordError").innerText = "Please confirm your password";

isValid = false;

} else if (password !== confirmPassword) {

document.getElementById("confirmPasswordError").innerText = "Passwords do not match";

isValid = false;

}

return isValid; // Form will not submit if `isValid` is false

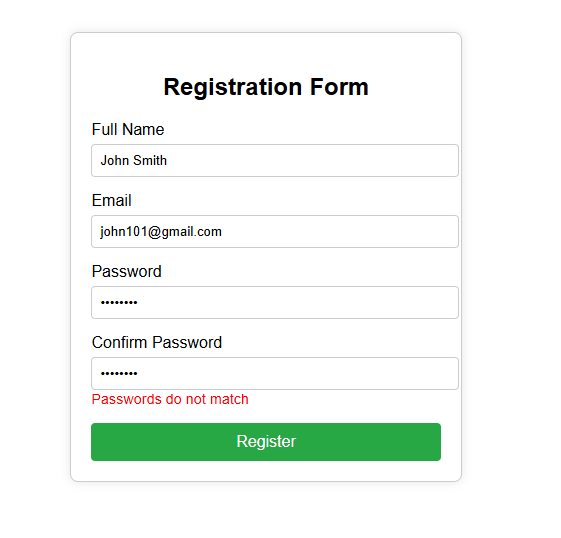
}

</script>

</body>

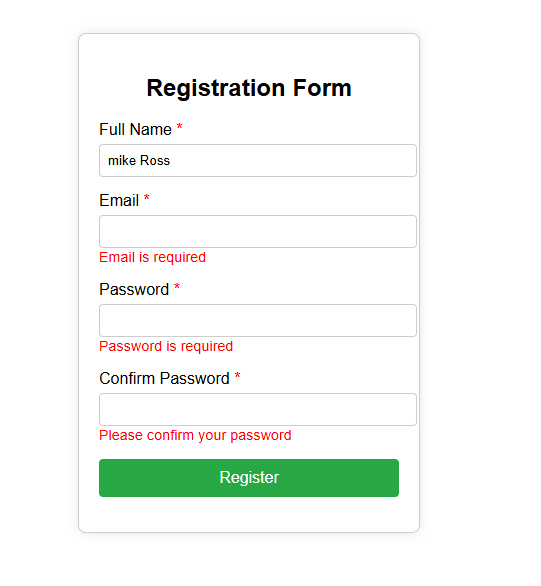
</html>

**Output**

** A screen shot of a registration form

Description automatically generated**

**A registration form with green and white text

Description automatically generated**

**Experiment-5**

**Aim** Write a program for implementing XML document for STUDENTDETAILS

**Code**

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

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

<Students>

<Student>

<RollNumber>101</RollNumber>

<Name>John Doe</Name>

<Department>Computer Science</Department>

<Year>3</Year>

<Subjects>

<Subject>Data Structures</Subject>

<Subject>Operating Systems</Subject>

<Subject>Database Management</Subject>

</Subjects>

</Student>

<Student>

<RollNumber>102</RollNumber>

<Name>Jane Smith</Name>

<Department>Electrical Engineering</Department>

<Year>2</Year>

<Subjects>

<Subject>Circuits</Subject>

<Subject>Electromagnetics</Subject>

<Subject>Control Systems</Subject>

</Subjects>

</Student>

<Student>

<RollNumber>103</RollNumber>

<Name>Michael Brown</Name>

<Department>Mechanical Engineering</Department>

<Year>4</Year>

<Subjects>

<Subject>Thermodynamics</Subject>

<Subject>Fluid Mechanics</Subject>

<Subject>Heat Transfer</Subject>

</Subjects>

</Student>

</Students>

**Experiment-6**

**Aim** Write a program to display contents of XML file in a table using Extensible Style Sheets

**Code**

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

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

<xsl:output method="html" indent="yes"/>

<xsl:template match="/">

<html>

<head>

<title>Student Details</title>

<style>

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

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

th { background-color: #f2f2f2; }

</style>

</head>

<body>

<h2>Student Details</h2>

<table>

<tr>

<th>Roll Number</th>

<th>Name</th>

<th>Department</th>

<th>Year</th>

<th>Subjects</th>

</tr>

<xsl:for-each select="Students/Student">

<tr>

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

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

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

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

<td>

<xsl:for-each select="Subjects/Subject">

<xsl:value-of select="."/>

<xsl:if test="position() != last()">, </xsl:if>

</xsl:for-each>

</td>

</tr>

</xsl:for-each>

</table>

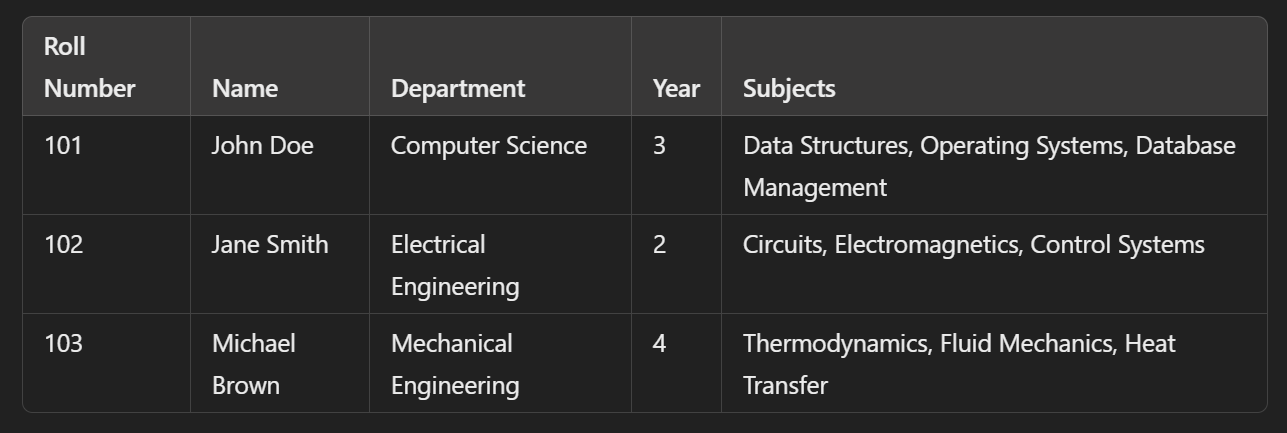
</body>

</html>

</xsl:template>

</xsl:stylesheet>

**Output**

****

**Experiment-7**

**Aim** Write a servlet that reads parameters from employee login page.

**Code**

// html

<!DOCTYPE html>

<html>

<head>

<title>Employee Login</title>

</head>

<body>

<h2>Employee Login Form</h2>

<form action="EmployeeLoginServlet" method="post">

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

<input type="text" id="username" name="username" required><br><br>

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

<input type="password" id="password" name="password" required><br><br>

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

</form>

</body>

</html>

// server.java

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

// Annotation to define the servlet URL pattern

@WebServlet("/EmployeeLoginServlet")

public class EmployeeLoginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

// Handles POST requests for employee login

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

// Set the response content type

response.setContentType("text/html");

// Retrieve the 'username' and 'password' parameters from the request

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

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

// Obtain a PrintWriter to send response data to the client

PrintWriter out = response.getWriter();

// Simple validation logic for demonstration purposes

if (username != null && password != null && !username.isEmpty() && !password.isEmpty()) {

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

out.println("<h2>Login Successful!</h2>");

out.println("<p>Welcome, " + username + "!</p>");

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

} else {

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

out.println("<h2>Login Failed</h2>");

out.println("<p>Invalid username or password. Please try again.</p>");

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

}

// Close the PrintWriter

out.close();

}

}

**Output**

**A screenshot of a login form

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Experiment-8**

**Aim** Write a program to create a single page website using Node.js and Express.js.

**Code**

// index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Single Page Website</title>

<link rel="stylesheet" href="/styles.css">

</head>

<body>

<h1>Welcome to My Single Page Website</h1>

<p>This is a simple website built with Node.js and Express.js.</p>

</body>

</html>

// style.css

body {

font-family: Arial, sans-serif;

text-align: center;

padding: 20px;

}

h1 {

color: #333;

}

p {

font-size: 18px;

color: #555;

}

// server.js

// Import required modules

const express = require('express');

const path = require('path');

// Create an instance of an Express app

const app = express();

// Define a port to run the server

const PORT = process.env.PORT || 3000;

// Set up a simple route for the homepage

app.get('/', (req, res) => {

  res.sendFile(path.join(\_\_dirname, 'public/index.html'));

});

// Serve static files (optional, e.g., CSS, images)

app.use(express.static(path.join(\_\_dirname, 'public')));

// Start the server

app.listen(PORT, () => {

  console.log(`Server is running on http://localhost:${PORT}`);

});

**Output**

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated