1. Develop an HTML page to display student details using a table and appropriate CSS styles. (CO1)

**Aim:** To develop an HTML webpage that displays student details in a tabular format using appropriate CSS styles for better presentation, readability, and visual appearance.

**Algorithm / Procedure:**

1. **Start the program.**
2. **Create a new HTML file**
   * Open your text editor and create a new file named student\_details.html.
3. **Define the HTML structure**
   * Use the <html>, <head>, and <body> tags to structure the document.
4. **Add a title to the page**
   * Within the <head> section, add a <title> tag for the webpage title.
5. **Write internal CSS styles**
   * Inside the <style> tag, define styles for:
     + body → to set font, background, and margins.
     + h2 → to align the main heading and set color.
     + table, th, td → to format borders, padding, alignment, and colors.
     + tr:hover → to add hover effects for interactivity.
6. **Create a heading in the body section**
   * Use <h2> tag for “Student Details”.
7. **Create a table structure**
   * Use <table> tag to define the table.
   * Use <tr> to create table rows.
   * Use <th> for header cells (Roll No, Name, Department, Email, Marks).
   * Use <td> for student data entries.
8. **Add multiple rows for student details**
   * Enter sample data for each student (minimum four records).
9. **Save the file**
   * Save the file with .html extension (e.g., student\_details.html).
10. **Execute the program**
    * Open the saved file in any web browser.
11. **Observe the output**
    * Verify that the table appears with proper styling, background color, and hover effects.
12. **End the program.**

Program:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Student Details</title>

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background-color: #f8f9fa;

margin: 50px;

}

h2 {

text-align: center;

color: #2c3e50;

text-transform: uppercase;

letter-spacing: 1px;

}

table {

border-collapse: collapse;

width: 80%;

margin: 30px auto;

box-shadow: 0 4px 8px rgba(0,0,0,0.1);

background-color: #ffffff;

}

th, td {

text-align: center;

padding: 12px 15px;

border-bottom: 1px solid #dddddd;

}

th {

background-color: #34495e;

color: #ffffff;

}

tr:hover {

background-color: #f1f1f1;

transition: 0.3s;

}

caption {

caption-side: top;

margin-bottom: 10px;

font-weight: bold;

font-size: 18px;

color: #2c3e50;

}

</style>

</head>

<body>

<h2>Student Details</h2>

<table>

<caption>Class: CSE - AI & DS | Semester: III</caption>

<tr>

<th>Roll No</th>

<th>Name</th>

<th>Department</th>

<th>Email</th>

<th>Marks</th>

</tr>

<tr>

<td>21A31A0501</td>

<td>seetha</td>

<td>CSE - AI&DS</td>

<td>pravallika@gmail.com</td>

<td>92%</td>

</tr>

<tr>

<td>21A31A0502</td>

<td>geetha</td>

<td>CSE - AI&DS</td>

<td>saiteja@gmail.com</td>

<td>88%</td>

</tr>

<tr>

<td>21A31A0503</td>

<td>akila</td>

<td>CSE - AI&DS</td>

<td>harika@gmail.com</td>

<td>90%</td>

</tr>

<tr>

<td>21A31A0504</td>

<td>Akash</td>

<td>CSE - AI&DS</td>

<td>akash@gmail.com</td>

<td>85%</td>

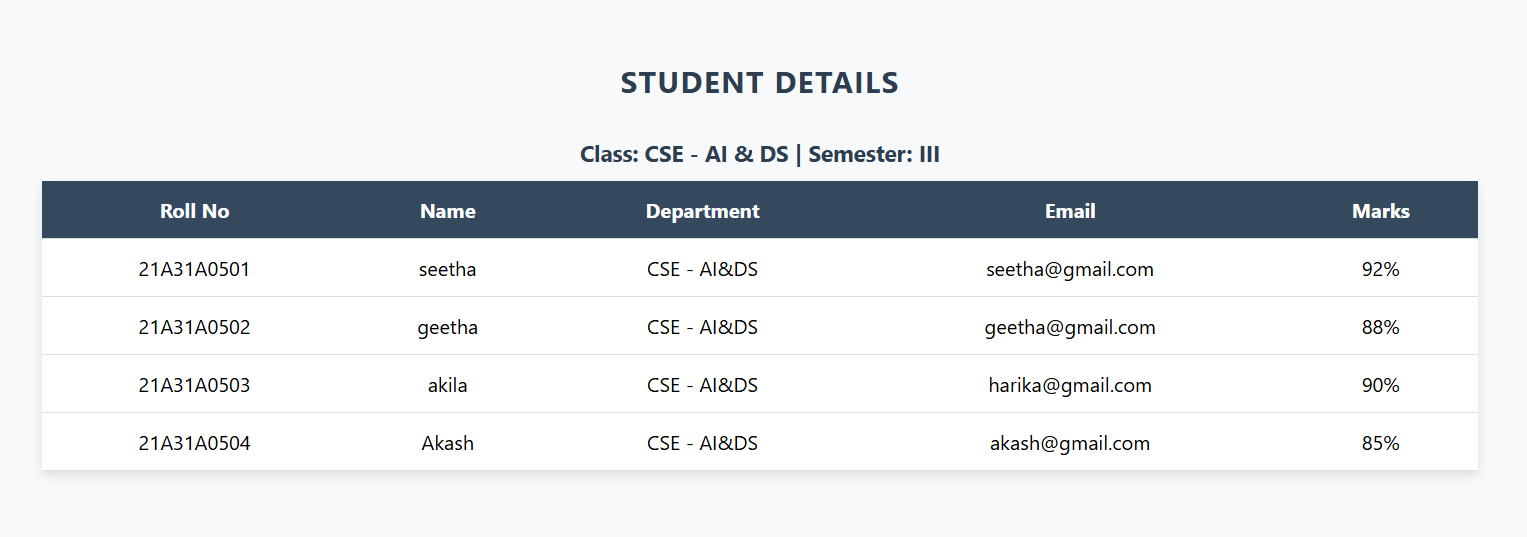
</tr>

</table>

</body>

</html>

Output:



1. Develop a responsive webpage using CSS Flexbox or Grid layout using media queries(CO1)

Aim: To design and develop a **responsive webpage** using **CSS Flexbox/Grid layout** that automatically adjusts its structure based on the screen size (desktop, tablet, and mobile view).

**Algorithm / Procedure:**

1. **Start the program.**
2. **Create a new HTML file**
   * Open any text editor and create a new file named responsive\_webpage.html.
3. **Define the HTML structure**
   * Use <!DOCTYPE html>, <html>, <head>, and <body> tags.
4. **Add metadata and title**
   * Inside <head>, include <meta name="viewport" content="width=device-width, initial-scale=1.0"> for responsiveness.
   * Add <title> tag for webpage title.
5. **Write CSS styles using Flexbox or Grid**
   * Use either:
     + display: flex; for **Flexbox** layout, or
     + display: grid; for **Grid** layout.
   * Apply responsive properties using **media queries** (@media) for different screen widths.
6. **Create main webpage structure**
   * Include the following sections:
     + Header
     + Navigation bar
     + Main content area
     + Sidebar
     + Footer
7. **Style the layout for desktop and mobile**
   * For desktop: show sections side-by-side.
   * For mobile: stack sections vertically.
8. **Save and execute the program**
   * Save as responsive\_webpage.html and open in a web browser.
   * Resize the window to check responsiveness.
9. **End the program.**

Program:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Responsive Webpage</title>

<style>

  body {

    font-family: Arial, sans-serif;

    margin: 0;

    background: #f4f4f4;

  }

  header, footer {

    background: #2c3e50;

    color: white;

    text-align: center;

    padding: 15px;

  }

  nav {

    background: #34495e;

    color: white;

    text-align: center;

    padding: 10px;

  }

  .container {

    display: flex;

    flex-wrap: wrap;

    padding: 10px;

  }

  .main {

    flex: 3;

    background: white;

    margin: 5px;

    padding: 15px;

  }

  .aside {

    flex: 1;

    background: #ecf0f1;

    margin: 5px;

    padding: 15px;

  }

  @media (max-width: 768px) {

    .container {

      flex-direction: column;

    }

  }

</style>

</head>

<body>

  <header><h2>Responsive Webpage</h2></header>

  <nav>Home | About | Services | Contact</nav>

  <div class="container">

    <div class="main">

      <h3>Main Content</h3>

      <p>This is the main section of the webpage. It adjusts automatically with screen size.</p>

    </div>

    <div class="aside">

      <h3>Sidebar</h3>

      <p>Additional info or links appear here.</p>

    </div>

  </div>

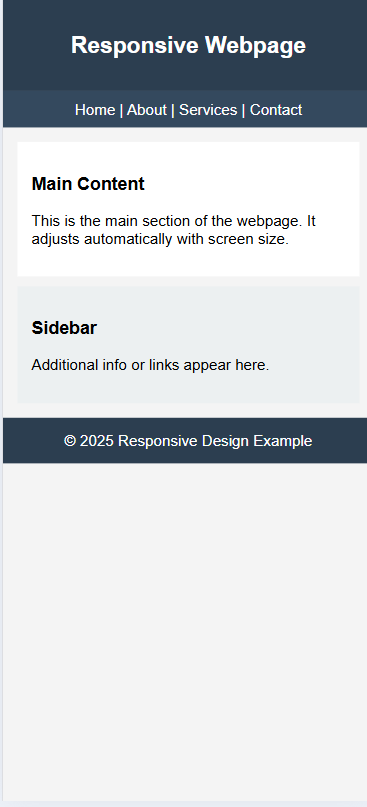
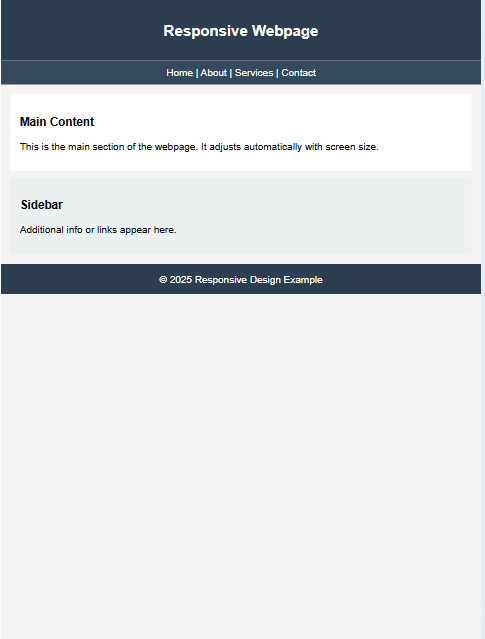
  <footer>&copy; 2025 Responsive Design Example</footer>

</body>

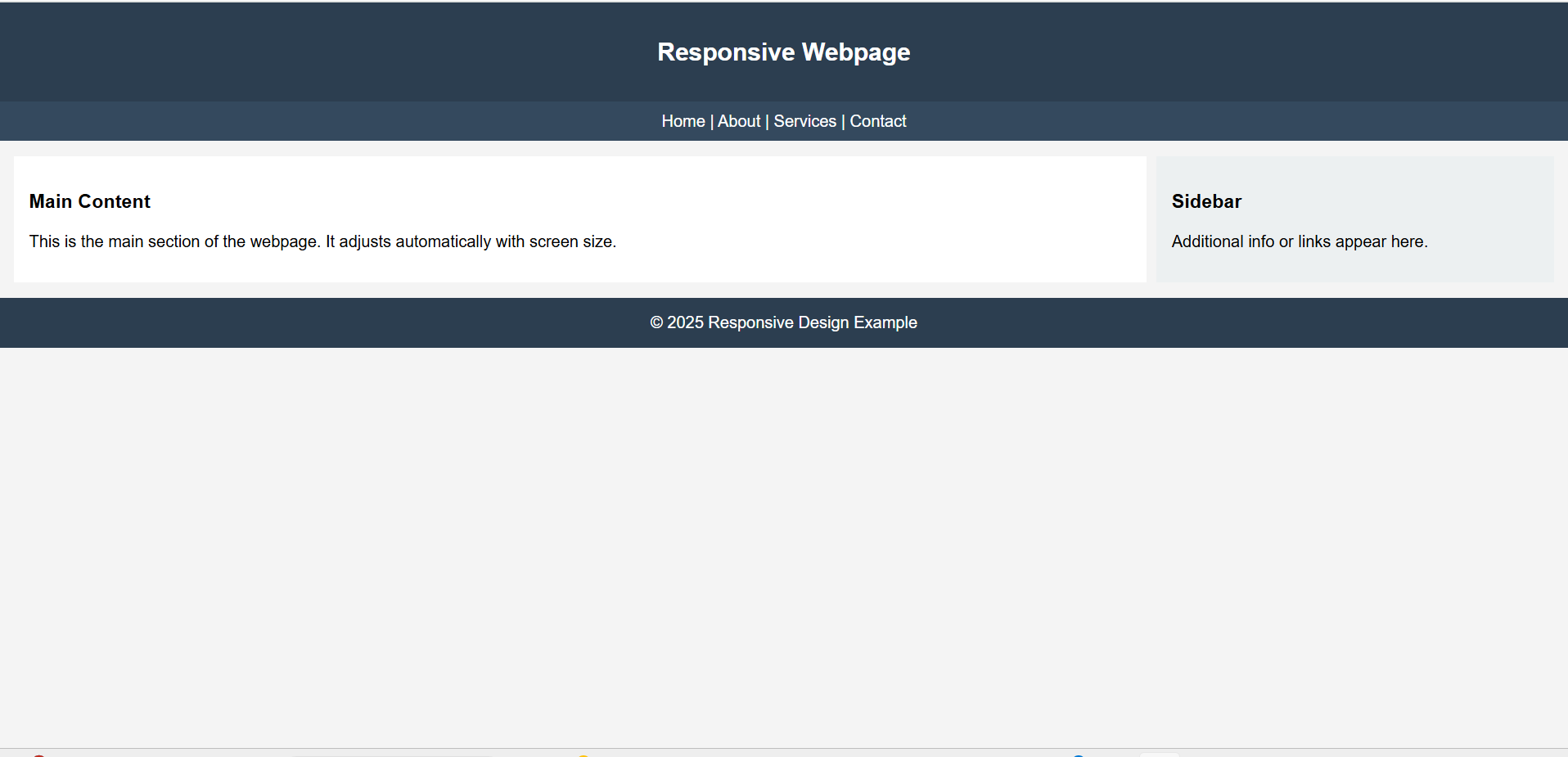
</html>

Output:

Mobile view: Tab view:

MOBILE VIEW TABLET view:

Desktop view:



1. Develop a JavaScript-based calculator web page that performs basic arithmetic operation(CO1)

Aim: To design and implement a **web-based calculator** using **HTML, CSS, and JavaScript** that can perform **basic arithmetic operations** such as addition, subtraction, multiplication, and division.

Aim: To design and implement a **web-based calculator** using **HTML, CSS, and JavaScript** that can perform **basic arithmetic operations** such as addition, subtraction, multiplication, and division.

**Procedure:**

1. **Create a new HTML file**
   * Open any text editor (Notepad / VS Code) and create a file named calculator.html.
2. **Define the basic HTML structure**
   * Use <html>, <head>, and <body> tags.
   * Set the <title> as “Calculator”.
3. **Design the calculator layout**
   * Use an <input> field to display numbers and results.
   * Use <button> elements for numbers (0–9) and operations (+, -, \*, /, =).
   * Optionally include a **Clear (C)** button to reset the input.
4. **Style the calculator using CSS**
   * Apply CSS for proper spacing, colors, and layout.
   * Use a **grid or flex layout** for arranging buttons neatly.
5. **Write JavaScript functions**
   * Create functions to handle:
     + **Number input**: append numbers to the display.
     + **Operation input**: store the selected operator.
     + **Calculate**: evaluate the expression when = is pressed.
     + **Clear**: reset the display.
6. **Link JavaScript with HTML**
   * Either use a <script> tag inside the HTML file or link an external .js file.
7. **Save and execute the program**
   * Save the file with .html extension and open it in a browser.
8. **Test the calculator**
   * Perform addition, subtraction, multiplication, and division.
   * Verify that results are correct and buttons function properly.
9. **End the experiment.**

Program:

<!DOCTYPE html>

<html>

<head>

<title>Simple Calculator</title>

<style>

body {

font-family: Arial, sans-serif;

text-align: center;

margin-top: 50px;

}

.calc {

display: inline-block;

border: 2px solid #0077cc;

padding: 20px;

border-radius: 10px;

background-color: #f0f8ff;

}

input {

width: 200px;

height: 30px;

font-size: 18px;

margin-bottom: 10px;

text-align: right;

}

button {

width: 45px;

height: 45px;

margin: 5px;

font-size: 18px;

}

</style>

</head>

<body>

<h2>Simple Calculator</h2>

<div class="calc">

<input type="text" id="display" readonly><br>

<button onclick="append('7')">7</button>

<button onclick="append('8')">8</button>

<button onclick="append('9')">9</button>

<button onclick="append('/')">÷</button><br>

<button onclick="append('4')">4</button>

<button onclick="append('5')">5</button>

<button onclick="append('6')">6</button>

<button onclick="append('\*')">×</button><br>

<button onclick="append('1')">1</button>

<button onclick="append('2')">2</button>

<button onclick="append('3')">3</button>

<button onclick="append('-')">−</button><br>

<button onclick="append('0')">0</button>

<button onclick="append('.')">.</button>

<button onclick="calculate()">=</button>

<button onclick="append('+')">+</button><br>

<button onclick="clearDisplay()">C</button>

</div>

<script>

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

function append(value) {

display.value += value;

}

function calculate() {

try {

display.value = eval(display.value);

} catch {

display.value = 'Error';

}

}

function clearDisplay() {

display.value = '';

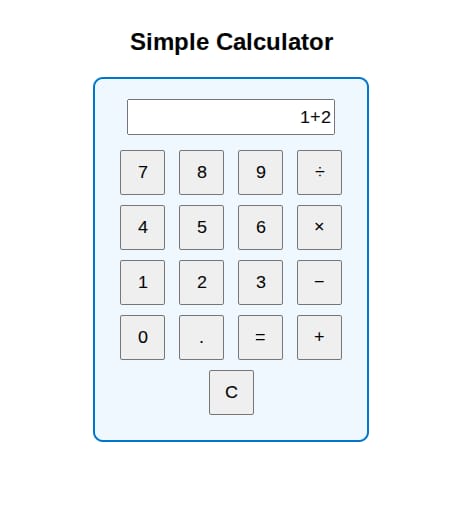
}

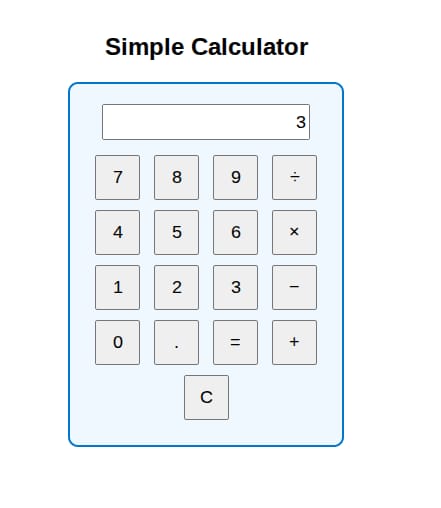
</script>

</body>

</html>

Output:



  
4. Write a JavaScript program to validate a login form (username and password).(CO2).

Aim: To develop a **web-based login form** using **HTML and JavaScript** that validates user input for **username and password**, and displays appropriate error messages for invalid input.

**Procedure:**

1. **Start the program.**
2. **Create a new HTML file**
   * Open a text editor (Notepad, VS Code, etc.) and create a file named login.html.
3. **Define HTML structure**
   * Add <html>, <head>, and <body> tags.
   * Set the <title> of the page.
4. **Add the heading for the form**
   * Use <h2> tag to display **“Login Form”** at the top of the page.
5. **Create input fields**
   * Add an <input> field for **username**.
   * Add an <input> field for **password**.
   * Add a **Login** <button> to submit the form.
6. **Add an area for error messages**
   * Use a <div> to display messages like "Invalid username or password".
7. **Write JavaScript validation function**
   * Retrieve values from the username and password fields.
   * Check if either field is empty and display a message.
   * Compare input values with **predefined valid credentials**.
   * If matched, display a **success message**; otherwise, show **error message**.
8. **Link JavaScript with HTML**
   * Include <script> at the end of the body for functionality.
9. **Save and execute the file**
   * Save the file as login.html and open it in a web browser.
10. **Test the form**
    * Enter valid credentials and check for **successful login alert**.
    * Enter invalid credentials and check for **error message**.
    * Leave fields empty and verify validation prompts.
11. **End the experiment.**

Program:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Login Form</title>

<style>

  body {

    font-family: Arial, sans-serif;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background: #f0f0f0;

  }

  .login-form {

    background: white;

    padding: 20px;

    border-radius: 5px;

    text-align: center;

  }

  h2 {

    margin-bottom: 15px;

    color: #2c3e50;

  }

  input {

    width: 100%;

    padding: 8px;

    margin: 5px 0;

    box-sizing: border-box;

  }

  button {

    width: 100%;

    padding: 8px;

    margin-top: 10px;

    background: #2c3e50;

    color: white;

    border: none;

    cursor: pointer;

  }

  .error {

    color: red;

    font-size: 14px;

    margin-bottom: 5px;

  }

</style>

</head>

<body>

<div class="login-form">

  <h2>Login Form</h2>

  <div class="error" id="error"></div>

  <input type="text" id="username" placeholder="Username">

  <input type="password" id="password" placeholder="Password">

  <button onclick="validate()">Login</button>

</div>

<script>

  const validUsers = [

    {username: "user1", password: "pwd1"},

    {username: "user2", password: "pwd2"},

    {username: "user3", password: "pwd3"},

    {username: "user4", password: "pwd4"}

  ];

  function validate() {

    const username = document.getElementById('username').value.trim();

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

    const errorDiv = document.getElementById('error');

    if(username === "" || password === "") {

      errorDiv.textContent = "Please enter both username and password.";

      return;

    }

    const isValid = validUsers.some(user => user.username === username && user.password === password);

    if(isValid) {

      alert("Login successful!");

      errorDiv.textContent = "";

    } else {

      errorDiv.textContent = "Invalid username or password.";

    }

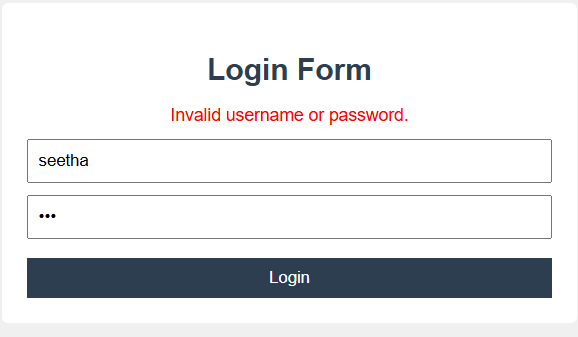
  }

</script>

</body>

</html>

Output:



5. Create a registration form using HTML and validate user input using JavaScript(CO2)

Aim: To design a **web-based registration form** using **HTML** and implement **JavaScript validation** to ensure that all user inputs are entered correctly, including required fields, email format, password rules, and other constraints.

**Procedure:**

1. **Start the program.**
2. **Create a new HTML file**
   * Open a text editor (Notepad, VS Code, etc.) and create a file named registration.html.
3. **Define HTML structure**
   * Use <html>, <head>, and <body> tags.
   * Set the <title> of the page.
4. **Add a heading for the form**
   * Use <h2> to display **“Registration Form”** at the top.
5. **Create input fields**
   * Include fields such as:
     + Full Name
     + Username
     + Email
     + Password
     + Confirm Password
     + Phone Number (optional)
   * Add a **Submit** button to send the form.
6. **Add an area for error messages**
   * Use a <div> or <span> to display validation errors for each field or collectively.
7. **Write JavaScript validation function**
   * Ensure required fields are **not empty**.
   * Validate **email format** using regex.
   * Check **password strength** and ensure **password and confirm password match**.
   * Optionally, validate **phone number** for numeric input and length.
   * Display appropriate error messages for invalid input.
   * If all inputs are valid, show a **success message** or proceed to submit.
8. **Link JavaScript with HTML**
   * Use a <script> tag at the bottom of the body to include validation code.
   * Call the validation function on **form submission** (onsubmit).
9. **Save and execute the file**
   * Save as registration.html and open it in a browser.
10. **Test the form**
    * Try submitting with empty fields → should show error.
    * Enter invalid email → should show error.
    * Enter mismatched passwords → should show error.
    * Enter valid inputs → should display **successful registration alert**.
11. **End the experiment.**

Program:

<!DOCTYPE html>

<html>

<head>

<title>Registration Form</title>

<style>

body {

font-family: Arial, sans-serif;

text-align: center;

margin-top: 40px;

}

.form-box {

display: inline-block;

background-color: #f0f8ff;

border: 2px solid #0077cc;

border-radius: 10px;

padding: 20px;

text-align: left;

}

input {

width: 250px;

height: 30px;

margin: 5px 0;

font-size: 15px;

padding-left: 5px;

}

button {

width: 120px;

height: 35px;

background-color: #0077cc;

color: white;

border: none;

border-radius: 5px;

font-size: 16px;

}

#message {

text-align: center;

font-weight: bold;

margin-top: 10px;

}

</style>

</head>

<body>

<h2>Registration Form</h2>

<div class="form-box">

<label>Full Name:</label><br>

<input type="text" id="name" placeholder="Enter your name"><br>

<label>Email:</label><br>

<input type="email" id="email" placeholder="Enter your email"><br>

<label>Password:</label><br>

<input type="password" id="password" placeholder="Enter password"><br>

<label>Confirm Password:</label><br>

<input type="password" id="confirm" placeholder="Confirm password"><br><br>

<button onclick="validateForm()">Register</button>

</div>

<p id="message"></p>

<script>

function validateForm() {

let name = document.getElementById("name").value;

let email = document.getElementById("email").value;

let password = document.getElementById("password").value;

let confirm = document.getElementById("confirm").value;

let message = document.getElementById("message");

if (name === "" || email === "" || password === "" || confirm === "") {

message.style.color = "red";

message.textContent = "All fields are required!";

}

else if (password !== confirm) {

message.style.color = "red";

message.textContent = "Passwords do not match!";

}

else {

message.style.color = "green";

message.textContent = "Registration successful!";

}

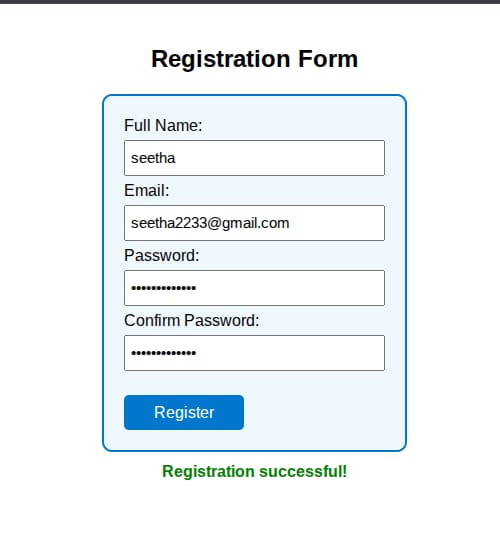
}

</script>

</body>

</html>

Output:



6. Design a web page that changes an background of the text when the mouse hovers over it.(CO2)

Aim:To design a **web page** in **HTML and CSS** that changes the **background color of text dynamically** when the user moves the mouse pointer over it, thereby demonstrating the use of the :hover pseudo-class in CSS.

**Procedure:**

1. **Start the program.**
2. **Create a new HTML file**
   * Open a text editor (Notepad, VS Code, etc.) and create a file named hover\_text.html.
3. **Define HTML structure**
   * Use <html>, <head>, and <body> tags.
   * Set the <title> of the webpage.
4. **Add heading and text**
   * Use <h2> for the heading (e.g., “Hover over the text”).
   * Use <p> or <span> tags for the text whose background will change on hover.
5. **Write CSS for hover effect**
   * Use p:hover or span:hover to define the **background color change**.
   * Optionally, change the **text color** for better visibility.
   * Use transition for a smooth effect.
6. **Save the file**
   * Save as hover\_text.html.
7. **Open in a browser**
   * Verify that moving the mouse pointer over the text **changes its background color**.
8. **Test with multiple text elements**
   * Add more <p> or <span> elements and ensure the hover effect works consistently.
9. **End the experiment.**

Program:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Text Hover Background</title>

<style>

  body {

    font-family: Arial, sans-serif;

    text-align: center;

    margin-top: 50px;

  }

  p {

    display: inline-block;

    padding: 10px 20px;

    font-size: 20px;

    transition: background-color 0.3s, color 0.3s;

  }

  /\* Change background color on hover \*/

  p:hover {

    background-color: yellow;

    color: black;

    cursor: pointer;

  }

</style>

</head>

<body>

<h2>Hover over the text</h2>

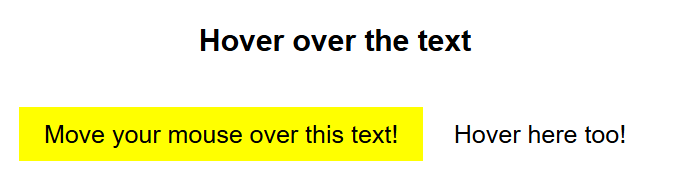
<p>Move your mouse over this text!</p>

<p>Hover here too!</p>

</body>

</html>

Output:



7.Make the static pages Responsive and attractive using Bootstrap components (\*Mobile View).(CO3)

Aim: To design a **responsive and attractive web page** using **Bootstrap components** such as navbar, buttons, cards, and footer, ensuring the page **adapts automatically to mobile, tablet, and desktop screens**.

**Procedure:**

1. **Start the program.**
2. **Create a new HTML file**
   * Open a text editor (Notepad, VS Code, etc.) and create a file named responsive.html.
3. **Include Bootstrap CSS and JS**
   * Add the Bootstrap CSS <link> in the <head> section.
   * Add the Bootstrap JS <script> before the closing </body> tag.
4. **Add a Navbar**
   * Use the navbar, navbar-expand-lg, and navbar-dark bg-primary classes.
   * Include links for Home, About, and Contact pages.
   * Make it collapsible for mobile screens using navbar-toggler.
5. **Create a Hero Section**
   * Add a heading (<h1>), paragraph (<p>), and a button (<a class="btn btn-primary">).
   * Use container and text-center classes for alignment and spacing.
6. **Add Cards Section**
   * Use row and col-md-4 classes to create a responsive grid of cards.
   * Include a card with card-body, card-title, card-text, and a small button.
7. **Add a Footer**
   * Use bg-primary text-white text-center py-3 classes to create a simple footer.
8. **Test Responsiveness**
   * Open the page in a browser.
   * Resize the window or view on a mobile device to ensure all components adjust correctly.
9. **End the experiment.**

Program:

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Bootstrap Components Example</title>

  <!-- Bootstrap CSS -->

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<!-- Navbar -->

<nav class="navbar navbar-expand-lg navbar-dark bg-primary">

  <div class="container-fluid">

    <a class="navbar-brand" href="#">MySite</a>

    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarNav">

      <ul class="navbar-nav ms-auto">

        <li class="nav-item"><a class="nav-link active" href="#">Home</a></li>

        <li class="nav-item"><a class="nav-link" href="#">About</a></li>

        <li class="nav-item"><a class="nav-link" href="#">Contact</a></li>

      </ul>

    </div>

  </div>

</nav>

<!-- Hero Section -->

<div class="container text-center my-5">

  <h1>Welcome to My Website</h1>

  <p class="lead">Responsive page using Bootstrap components</p>

  <a href="#" class="btn btn-primary">Get Started</a>

</div>

<!-- Card Component -->

<div class="container my-4">

  <div class="row">

    <div class="col-md-4 mb-3">

      <div class="card">

        <div class="card-body">

          <h5 class="card-title">Card 1</h5>

          <p class="card-text">Simple description here.</p>

          <a href="#" class="btn btn-primary btn-sm">Learn More</a>

        </div>

      </div>

    </div>

    <div class="col-md-4 mb-3">

      <div class="card">

        <div class="card-body">

          <h5 class="card-title">Card 2</h5>

          <p class="card-text">Simple description here.</p>

          <a href="#" class="btn btn-primary btn-sm">Learn More</a>

        </div>

      </div>

    </div>

  </div>

</div>

<!-- Footer -->

<footer class="bg-primary text-white text-center py-3">

  &copy; 2025 MySite. All rights reserved.

</footer>

<!-- Bootstrap JS -->

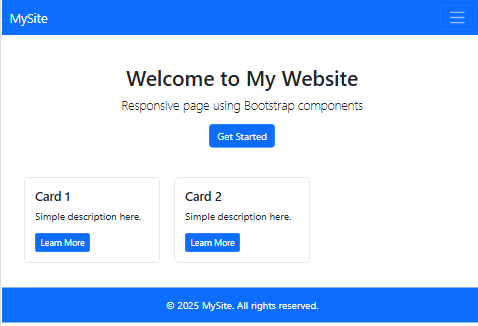
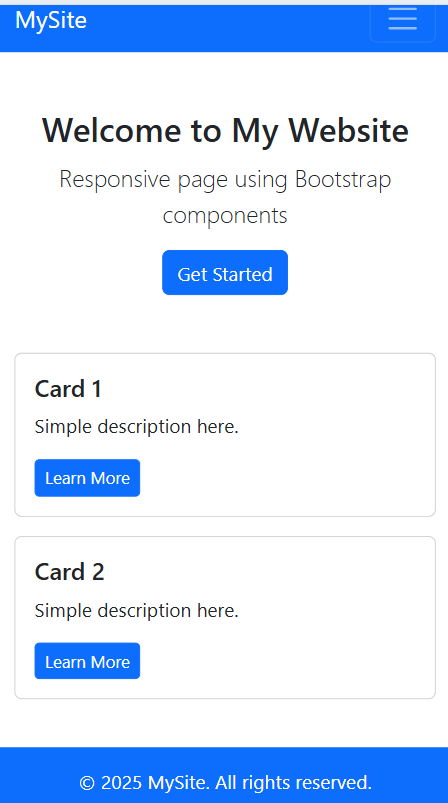
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

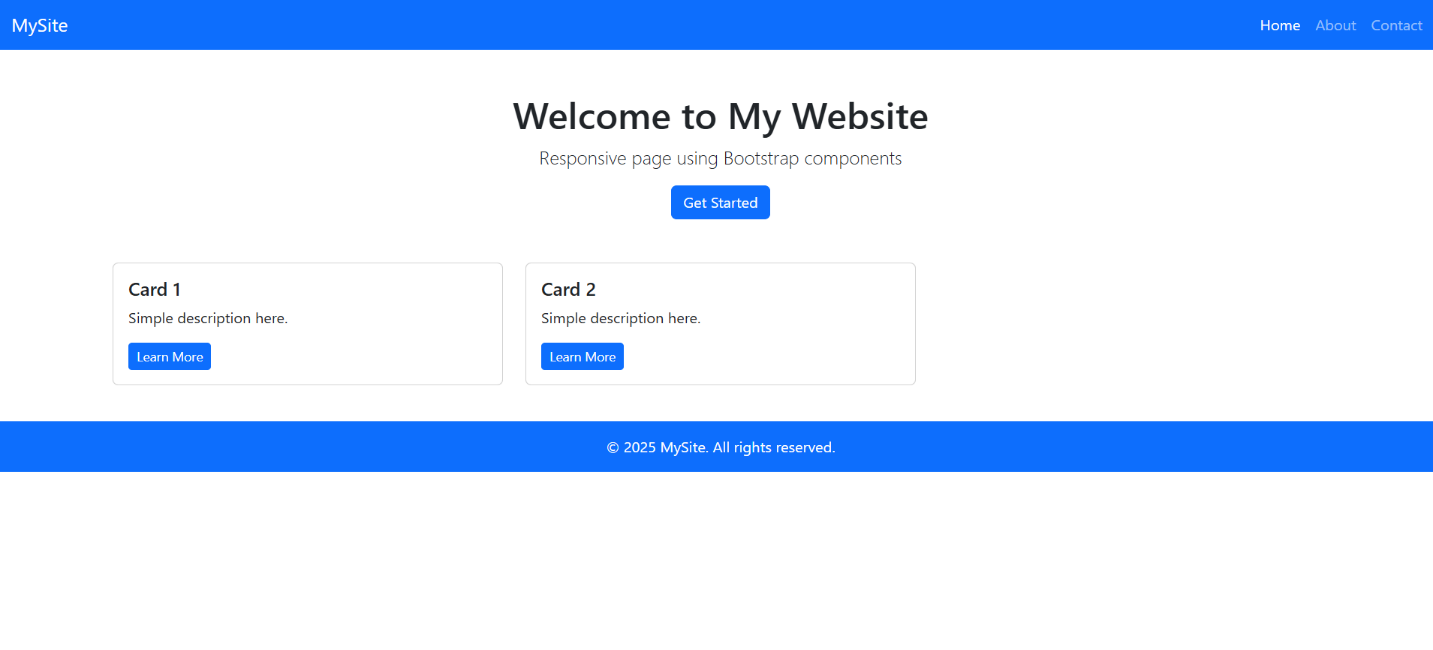
</body>

</html>

Output:

Mobile view: Tablet View

  
Desktop view:



8. Create a jQuery script to change the background color of a web page when a button is clicked.(CO3)

**Aim:**

To design a **web page** using **HTML and jQuery** that changes the **background color dynamically** when a button is clicked, demonstrating the use of **jQuery event handling and DOM manipulation**.

**Procedure:**

1. **Start the program.**
2. **Create a new HTML file**
   * Open a text editor (Notepad, VS Code, etc.) and create a file named change\_bg.html.
3. **Include jQuery library**
   * Add the jQuery <script> link in the <head> or before closing <body> tag:
   * <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
4. **Add HTML structure**
   * Include a heading <h2> to describe the page.
   * Add a <button> element that will trigger the background color change.
5. **Write jQuery script**
   * Use the $(document).ready() function to ensure the DOM is loaded.
   * Select the button using its id or class.
   * Attach a click event handler.
   * Inside the handler, use $("body").css("background-color", "color") to change the page’s background color.
6. **Save the file**
   * Save as change\_bg.html.
7. **Open the page in a browser**
   * Click the button and verify that the **background color changes** as expected.
8. **Test with multiple colors (optional)**
   * You can add more buttons or random color functionality to enhance interaction.
9. **End the experiment.**

**Program:**

<!DOCTYPE html>

<html>

<head>

<title>Change Background Color</title>

<!-- jQuery CDN -->

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

</head>

<body style="text-align:center; margin-top:50px; font-family:Arial;">

<h2>Click the button to change background color</h2>

<button id="btn">Change Color</button>

<script>

$(document).ready(function(){

$("#btn").click(function(){

$("body").css("background-color", "lightgreen");

});

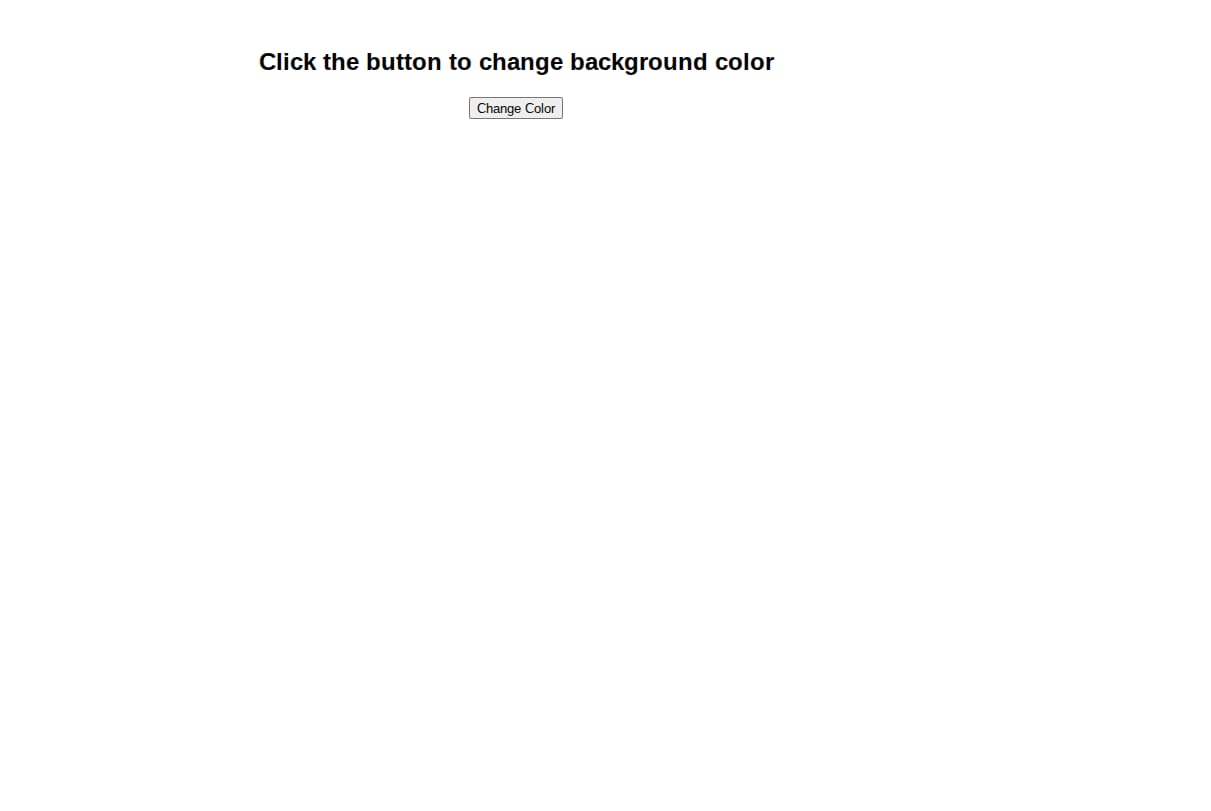
});

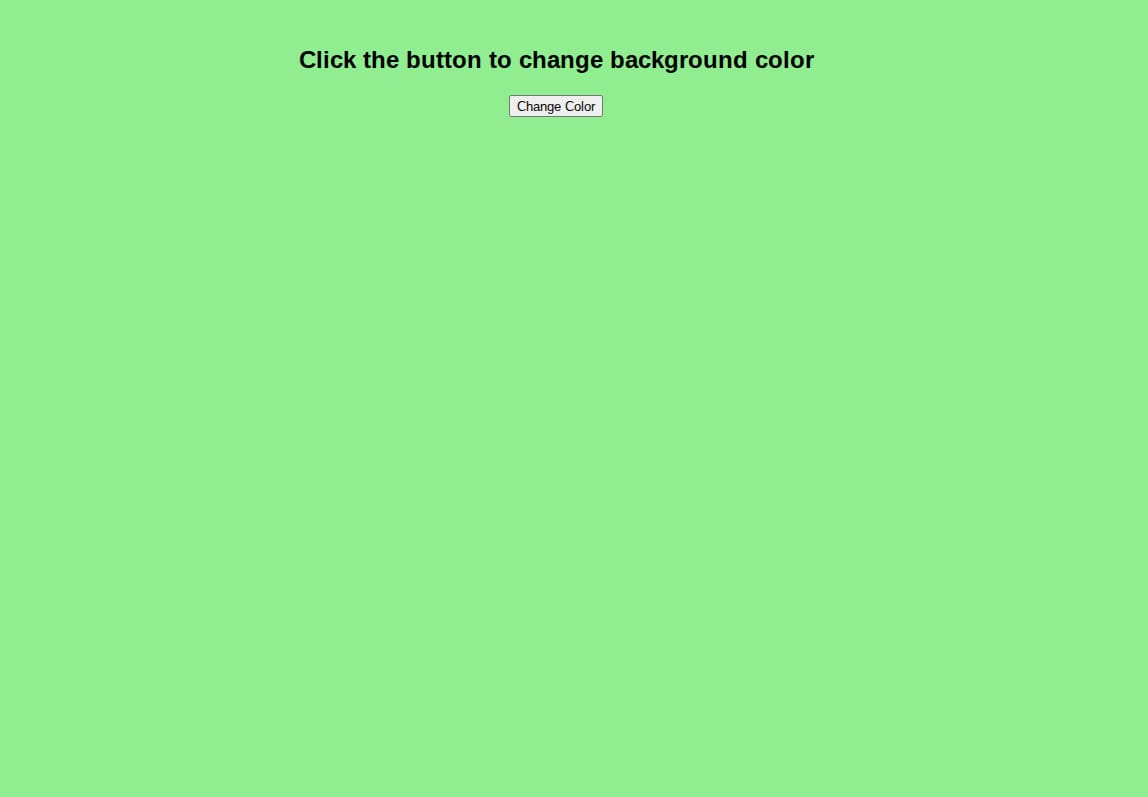
</script>

</body>

</html>

Output:

Before : After:

  
9. Write a JavaScript code to parse a JSON object containing student details and display them in an HTML table.(CO3)

Aim:To develop a **JavaScript program** that parses a **JSON object** containing student details and dynamically displays the data in an **HTML table** format on a web page.

**Procedure:**

1. **Start the program.**
2. **Create a new HTML file**
   * Open a text editor (Notepad, VS Code, etc.) and create a file named student\_json.html.
3. **Define JSON data**
   * Inside a <script> tag, define a JSON object or array that contains student details such as **name, roll number, and department**.

Example:

var students = [

{ "name": "vallika", "rollno": "21AD001", "dept": "AIDS" },

{ "name": "Kiranmay", "rollno": "21AD002", "dept": "AIDS" }

];

1. **Create HTML structure**
   * Add a <table> element with column headers (Name, Roll Number, Department).
   * Assign an id to the table to manipulate it using JavaScript.
2. **Parse and display JSON data**
   * Use JavaScript to **loop through the JSON array** and insert rows dynamically into the table using DOM methods like:
   * document.getElementById("tableId").innerHTML += ...
3. **Add basic CSS styles**
   * Apply simple styles (like borders and alignment) to make the table more readable.
4. **Save and run the file**
   * Save the file and open it in a web browser.
   * Verify that the **JSON data is displayed properly** in the HTML table.
5. **End the experiment.**

Program:

<!DOCTYPE html>

<html>

<head>

  <title>JSON Parsing</title>

</head>

<body>

  <h2>Student Details</h2>

  <table id="studentTable"></table>

  <script>

    // JSON string (could also come from API)

    let jsonString = `{

      "students": [

        { "id": 101, "name": "seetha", "age": 22, "course": "AI & DS" },

        { "id": 102, "name": "Rahul", "age": 23, "course": "CSE" },

        { "id": 103, "name": "latha", "age": 21, "course": "IT" }      ]    }`;

    // Parse JSON string → JavaScript object

    let obj = JSON.parse(jsonString);

    // Create table header

    let table = "<tr><th>ID</th><th>Name</th><th>Age</th><th>Course</th></tr>";

    // Loop through JSON array and add rows

    for (let s of obj.students) {

      table += "<tr>" +

                  "<td>" + s.id + "</td>" +

                  "<td>" + s.name + "</td>" +

                  "<td>" + s.age + "</td>" +

                  "<td>" + s.course + "</td>" +

               "</tr>";    }

    // Insert into HTML table

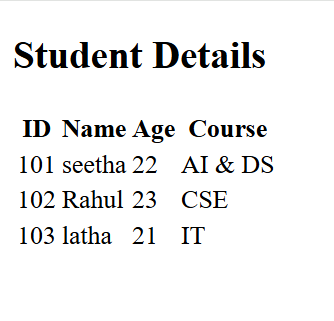
    document.getElementById("studentTable").innerHTML = table;

  </script>

</body>

</html>

Output:



10. Develop and demonstrate PHP Script for the following problems:

a. Write a PHP Script to find out the Sum of the Individual Digits.

b. Write a PHP Script to check whether the given number is Palindrome or not.

**a. sum of digits.**

**Aim:**

To write a PHP program that finds the sum of individual digits of a given number.

**Algorithm:**

1. Start the program.
2. Take input number from the user.
3. Initialize sum = 0.
4. Use a loop to extract digits using modulus (%) and divide by 10.
5. Add each digit to sum.
6. Display the result.

**Program:**

a. sum of digits

<!DOCTYPE html>

<html>

<body>

<form method="post">

Enter a number: <input type="text" name="num">

<input type="submit" name="submit" value="Find Sum">

</form>

<?php

if(isset($\_POST['submit'])) {

$num = $\_POST['num'];

$sum = 0;

$temp = $num;

while($temp > 0) {

$digit = $temp % 10;

$sum += $digit;

$temp = (int)($temp / 10);

}

echo "Sum of digits of $num is $sum";

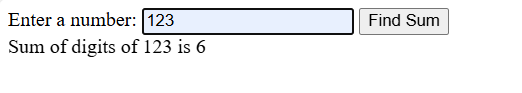
}

?>

</body>

</html>

**Output:**



**b. palindrome or not**

**Aim:**

To write a PHP script to check whether a number is a palindrome or not.

**Algorithm:**

1. Start the program.
2. Read input number.
3. Reverse the number.
4. Compare reversed number with original.
5. Display whether palindrome or not.

**Program:**

<!DOCTYPE html>

<html>

<body>

<form method="post">

Enter a number: <input type="text" name="num">

<input type="submit" name="check" value="Check Palindrome">

</form>

<?php

if(isset($\_POST['check'])) {

$num = $\_POST['num'];

$rev = strrev($num);

if($num == $rev)

echo "$num is a Palindrome number.";

else

echo "$num is not a Palindrome number.";

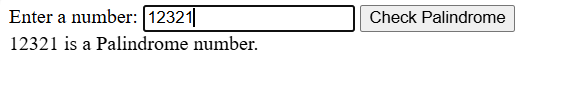
}

?>

</body>

</html>

**Output:**



11. Write a PHP script to connect with MySQL and insert as well as display student records.

**Aim:**

To connect PHP with MySQL and perform insert and display operations on student records.

**Algorithm:**

1. Connect to MySQL server.
2. Create a database and table if not exists.
3. Insert record using form.
4. Retrieve and display records from the database.

**Program:**

<?php

$servername = "localhost";

$username = "root";

$password = "";

// Create connection

$conn = new mysqli($servername, $username, $password);

if ($conn->connect\_error)

die("Connection failed: " . $conn->connect\_error);

// Create database if not exists

$conn->query("CREATE DATABASE IF NOT EXISTS studentdb");

// Select the database

$conn->select\_db("studentdb")

// Create table if not exists

$conn->query("CREATE TABLE IF NOT EXISTS students (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50),

rollno VARCHAR(20) UNIQUE,

branch VARCHAR(50)

)");

if(isset($\_POST['submit'])) {

$name = $\_POST['name'];

$rollno = $\_POST['rollno'];

$branch = $\_POST['branch'];

$insert = "INSERT INTO students (name, rollno, branch) VALUES ('$name', '$rollno', '$branch')";

if ($conn->query($insert)) {

echo "<p style='color:green;'>Record inserted successfully!</p>";

} else {

echo "<p style='color:red;'>Error: Could not insert record (Duplicate Roll No or SQL error)</p>";

}

}

?>

<!DOCTYPE html>

<html>

<body>

<h3>Insert and Display Student Records</h3>

<form method="post">

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

Roll No: <input type="text" name="rollno" required><br><br>

Branch: <input type="text" name="branch" required><br><br>

<input type="submit" name="submit" value="Insert Record">

</form>

<h3>Student Records:</h3>

<?php

$result = $conn->query("SELECT \* FROM students");

if ($result->num\_rows > 0) {

while($row = $result->fetch\_assoc()) {

echo $row['id'] . " | " . $row['name'] . " | " . $row['rollno'] . " | " . $row['branch'] . "<br>";

}

} else {

echo "No records found.";

}

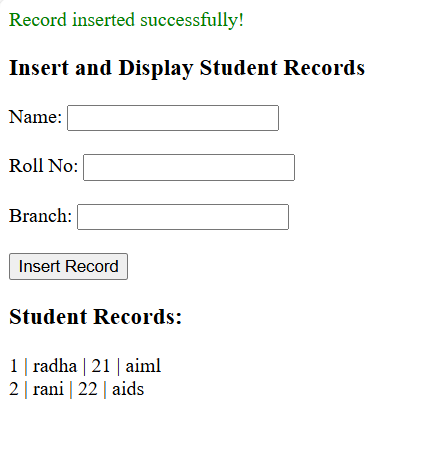
$conn->close();

?>

</body>

</html>

**Output:**



12. Develop a PHP form for student registration including photograph and display submitted data.

**Aim:**

To design a student registration form with photo upload and display submitted data.

**Algorithm:**

1. Create HTML form with name, roll, email, and photo upload.
2. On form submission, move uploaded file to uploads/ folder.
3. Display submitted data and photo.

**Program:**

<!DOCTYPE html>

<html>

<body>

<form method="post" enctype="multipart/form-data">

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

Roll No: <input type="text" name="roll"><br>

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

Photo: <input type="file" name="photo"><br>

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

</form>

<?php

if(isset($\_POST['submit'])) {

$name = $\_POST['name'];

$roll = $\_POST['roll'];

$email = $\_POST['email'];

$photo = "uploads/" . basename($\_FILES["photo"]["name"]);

move\_uploaded\_file($\_FILES["photo"]["tmp\_name"], $photo);

echo "<h3>Registration Details:</h3>";

echo "Name: $name <br> Roll No: $roll <br> Email: $email <br>";

echo "<img src='$photo' width='100'>";

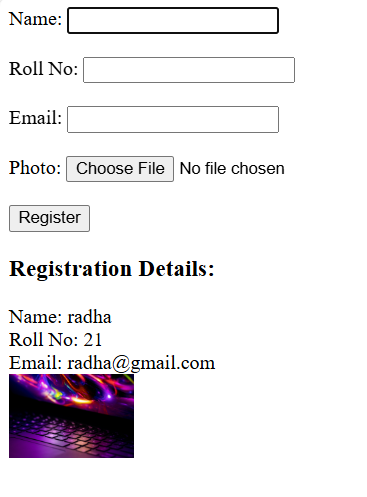
}

?>

</body>

</html>

**Output:**



13. write a PHP program ,assuming four users user1,user2,user3,user4 having pwd1,pwd2,pwd3,pwd4, respectively ,create a cookie and add these 4 users ID's and passwords to this cookie. Read the user id and password you entered in login form and authenticate it using JQUERY with the values available in cookies.

**Aim:**

To create cookies for four users with passwords and authenticate login using jQuery.

**Algorithm:**

1. Create cookie in PHP storing user IDs and passwords.
2. Create login form in HTML.
3. Use jQuery to validate entered credentials with cookie values.
4. Display success/failure message.

**Program:**

<?php

setcookie("user1", "pwd1");

setcookie("user2", "pwd2");

setcookie("user3", "pwd3");

setcookie("user4", "pwd4");

?>

<!DOCTYPE html>

<html>

<head>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

</head>

<body>

<h3>Login Form</h3>

User ID: <input type="text" id="uid"><br>

Password: <input type="password" id="pwd"><br>

<button id="loginBtn">Login</button>

<p id="result"></p>

<script>

$(document).ready(function(){

$("#loginBtn").click(function(){

let uid = $("#uid").val();

let pwd = $("#pwd").val();

let cookiePwd = getCookie(uid);

if(cookiePwd && cookiePwd === pwd)

$("#result").text("Login Successful!");

else

$("#result").text("Invalid Credentials!");

});

function getCookie(name) {

let value = "; " + document.cookie;

let parts = value.split("; " + name + "=");

if (parts.length === 2) return parts.pop().split(";").shift();

}

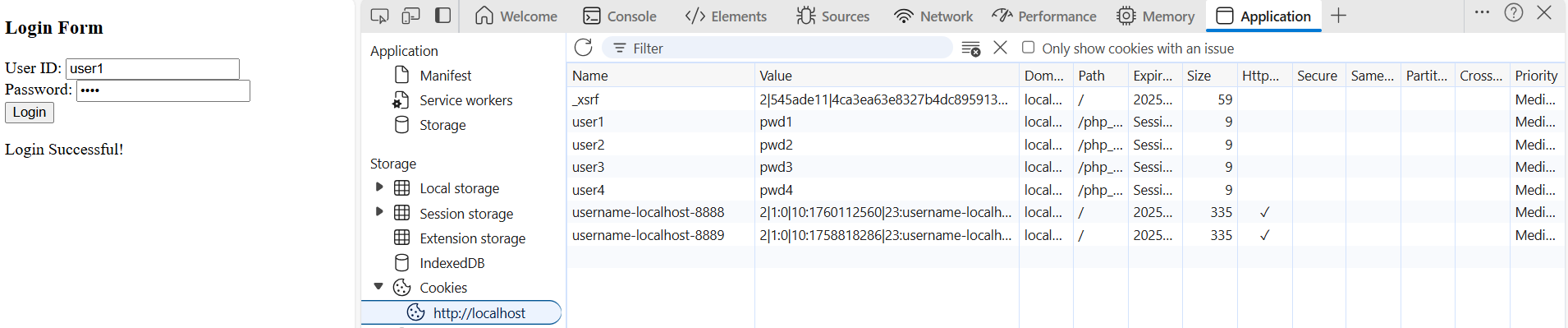
});

</script>

</body>

</html>

**Output:**



14. Write a PHP script to show an error message if the user tries to delete a record that does not exist in the database.

**Aim:**

To develop a PHP script that connects to a MySQL database and displays an error message when the user attempts to delete a record that does not exist.

**Algorithm:**

1. Start the program.
2. Establish a connection to the MySQL database using mysqli\_connect() or new mysqli().
3. Accept the roll number (or unique key) from the user through an HTML form.
4. When the user submits the form, execute a SELECT query to check if the record with the entered roll number exists in the database.
5. **If the record exists:**  
   Execute a DELETE query to remove it and display a success message.

**If the record does not exist:**  
Display an error message saying the record is not found.

1. Close the database connection.
2. End the program.

**Program:**

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "studentdb";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) die("Connection failed: " . $conn->connect\_error);

if(isset($\_POST['delete'])) {

$rollno = $\_POST['rollno'];

$check = $conn->query("SELECT \* FROM students WHERE rollno='$rollno'");

if($check->num\_rows > 0) {

$conn->query("DELETE FROM students WHERE rollno='$rollno'");

echo "<p style='color:green;'>Record deleted successfully!</p>";

} else {

echo "<p style='color:red;'>Error: Record with Roll No $rollno does not exist.</p>";

}

}

?>

<!DOCTYPE html>

<html>

<body>

<h3>Delete Student Record</h3>

<form method="post">

Enter Roll No: <input type="text" name="rollno" required>

<input type="submit" name="delete" value="Delete">

</form>

</body>

</html>

**Output:**

