ASSIGNMENT-14.1

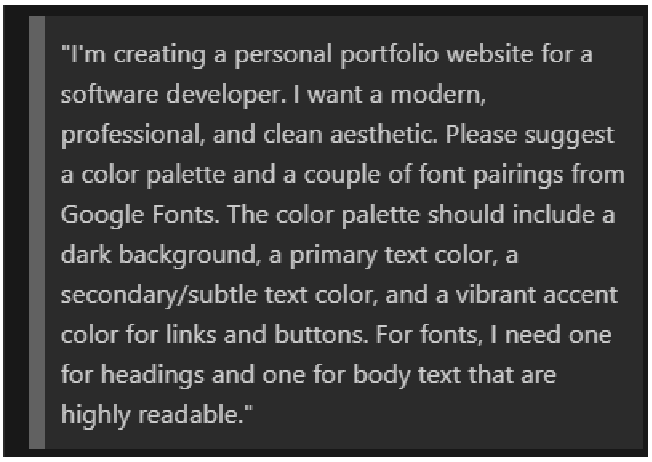
NAME:CHANDA HARINI

ROLL NO:2403A510E1

BATCH NO.:05

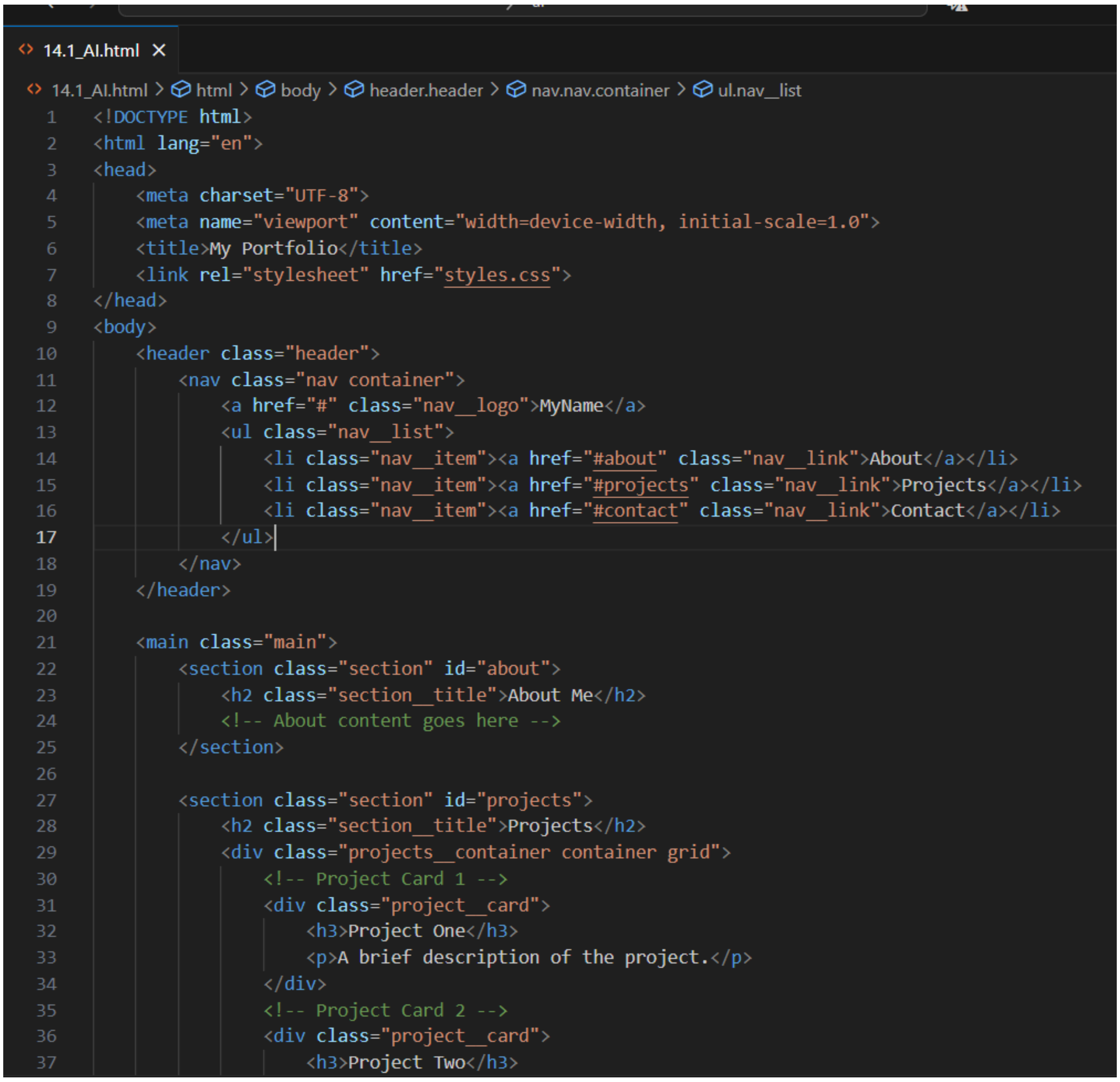
TASK-1: Portfolio Website Design

PROMPT:

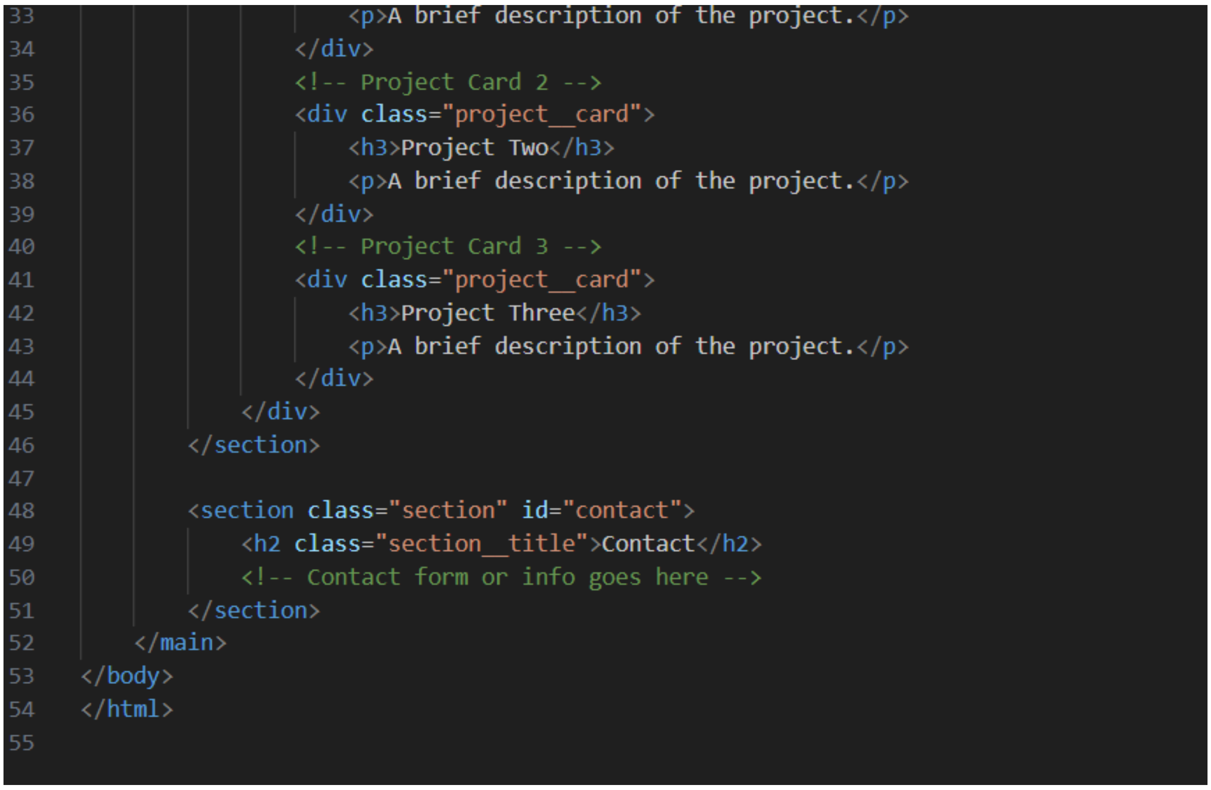


"I'm creating a personal portfolio website for a software developer. I want a modern, professional, and clean aesthetic. Please suggest a color palette and a couple of font pairings from Google Fonts. The color palette should include a dark background, a primary text color, a secondary/subtle text color, and a vibrant accent color for links and buttons. For fonts, I need one for headings and one for body text that are highly readable."

CODE:

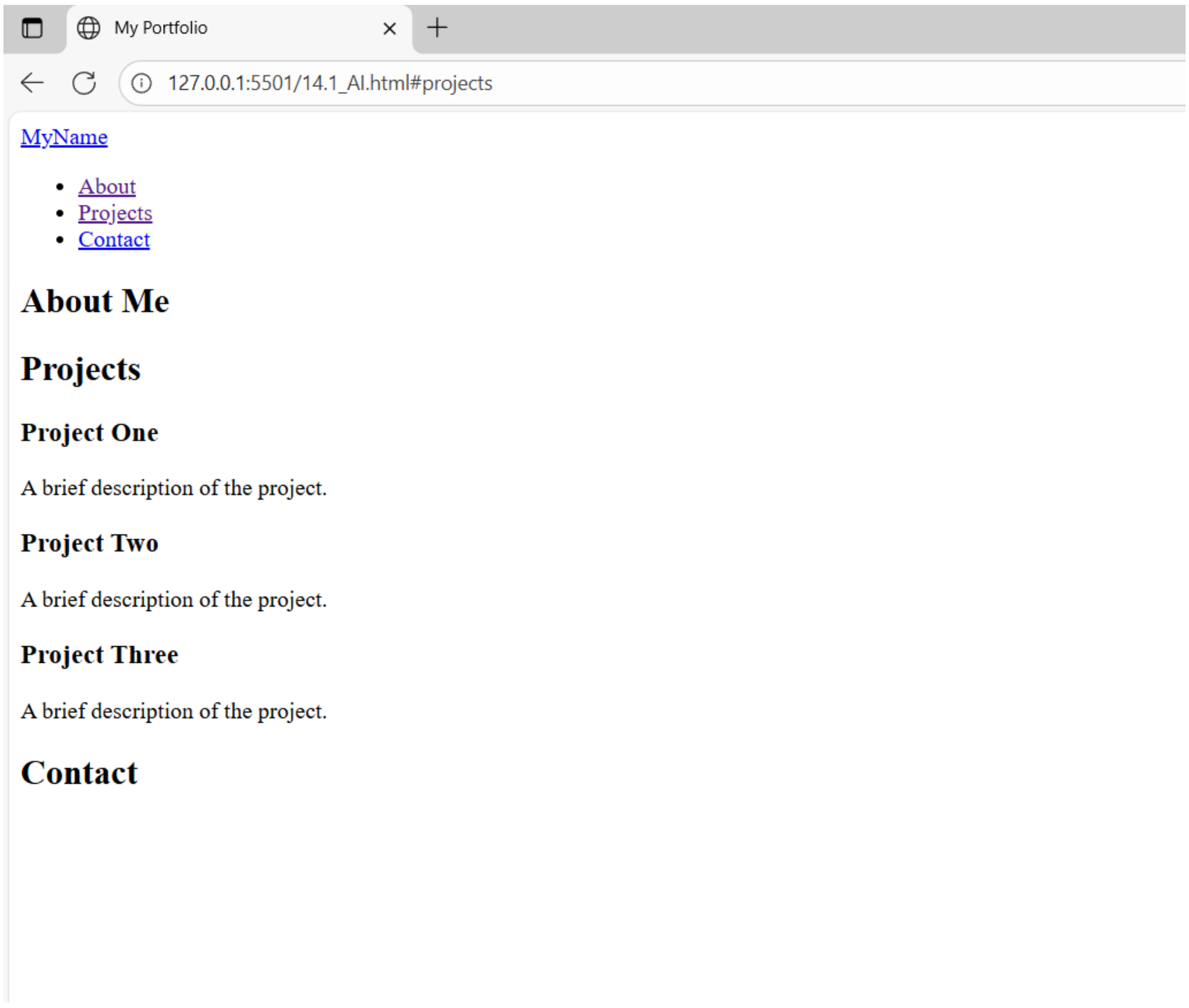

14.1_AI.html
14.1_AI.html > html > body > header.header > nav.nav.container > ul.nav__list

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Portfolio</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header class="header">
        <nav class="nav container">
            <a href="#" class="nav__logo">MyName</a>
            <ul class="nav__list">
                <li class="nav__item"><a href="#about" class="nav__link">About</a></li>
                <li class="nav__item"><a href="#projects" class="nav__link">Projects</a></li>
                <li class="nav__item"><a href="#contact" class="nav__link">Contact</a></li>
            </ul>
        </nav>
    </header>

    <main class="main">
        <section class="section" id="about">
            <h2 class="section__title">About Me</h2>
            <!-- About content goes here -->
        </section>

        <section class="section" id="projects">
            <h2 class="section__title">Projects</h2>
            <div class="projects__container container grid">
                <!-- Project Card 1 -->
                <div class="project__card">
                    <h3>Project One</h3>
                    <p>A brief description of the project.</p>
                </div>
                <!-- Project Card 2 -->
                <div class="project__card">
                    <h3>Project Two</h3>
```



```html
                    <p>A brief description of the project.</p>
                </div>
                <!-- Project Card 2 -->
                <div class="project__card">
                    <h3>Project Two</h3>
                    <p>A brief description of the project.</p>
                </div>
                <!-- Project Card 3 -->
                <div class="project__card">
                    <h3>Project Three</h3>
                    <p>A brief description of the project.</p>
                </div>
            </div>
        </section>

        <section class="section" id="contact">
            <h2 class="section__title">Contact</h2>
            <!-- Contact form or info goes here -->
        </section>
    </main>
</body>
</html>

```

OUTPUT:



MyName

- About
- Projects
- Contact

About Me

Projects

Project One

A brief description of the project.

Project Two

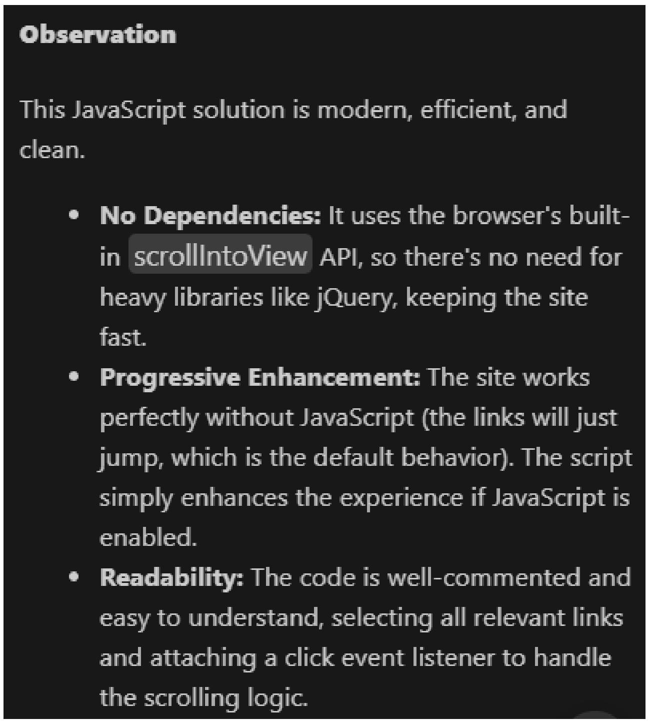
A brief description of the project.

Project Three

A brief description of the project.

Contact

OBSERVATION:



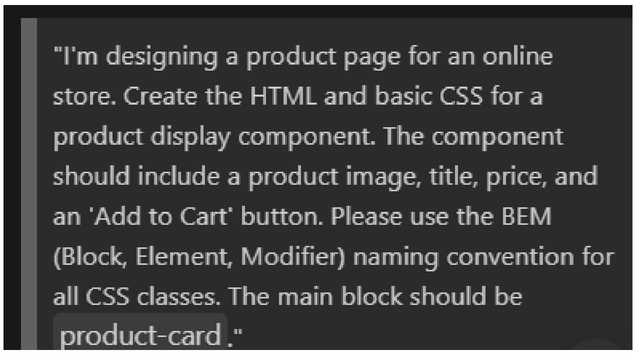
Observation

This JavaScript solution is modern, efficient, and clean.

- **No Dependencies:** It uses the browser's built-in `scrollIntoView` API, so there's no need for heavy libraries like jQuery, keeping the site fast.
- **Progressive Enhancement:** The site works perfectly without JavaScript (the links will just jump, which is the default behavior). The script simply enhances the experience if JavaScript is enabled.
- **Readability:** The code is well-commented and easy to understand, selecting all relevant links and attaching a click event listener to handle the scrolling logic.

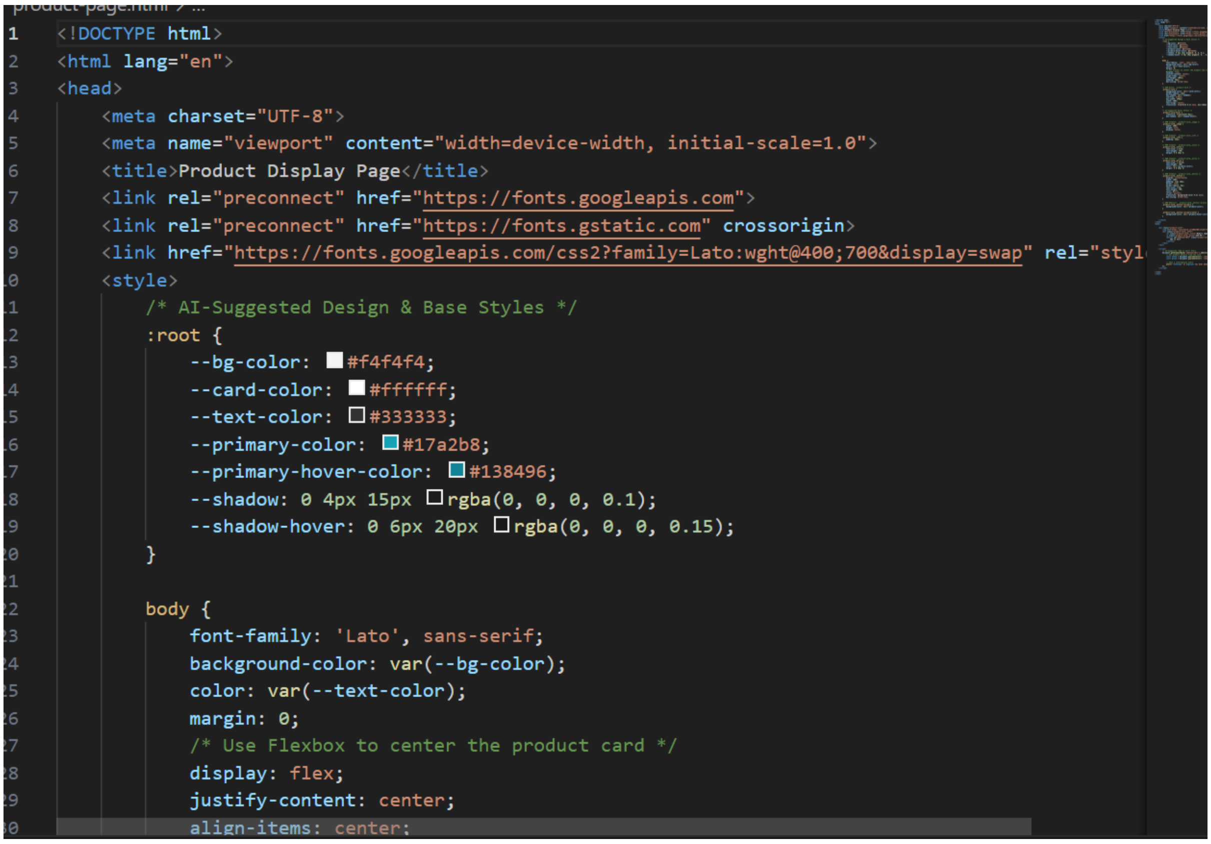
TASK-2: Online Store Product Page

PROMPT:



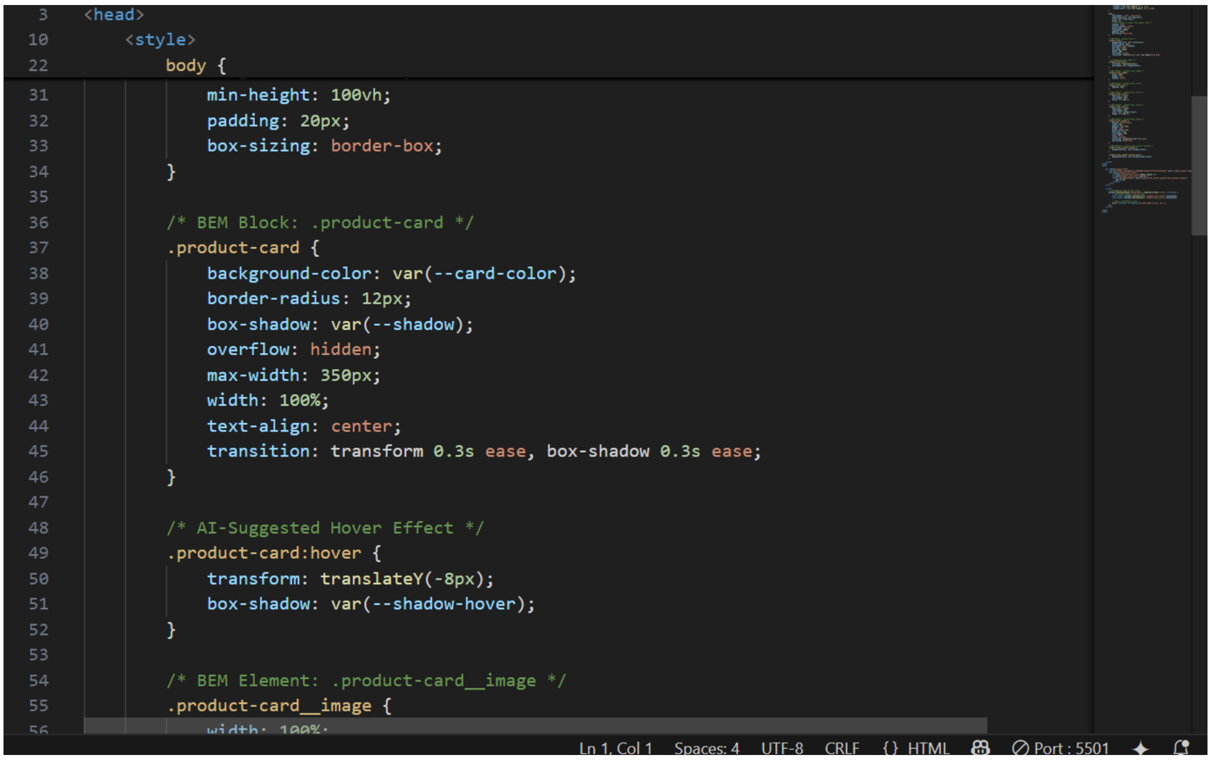
"I'm designing a product page for an online store. Create the HTML and basic CSS for a product display component. The component should include a product image, title, price, and an 'Add to Cart' button. Please use the BEM (Block, Element, Modifier) naming convention for all CSS classes. The main block should be `product-card`."

CODE:



product-page.html > ...

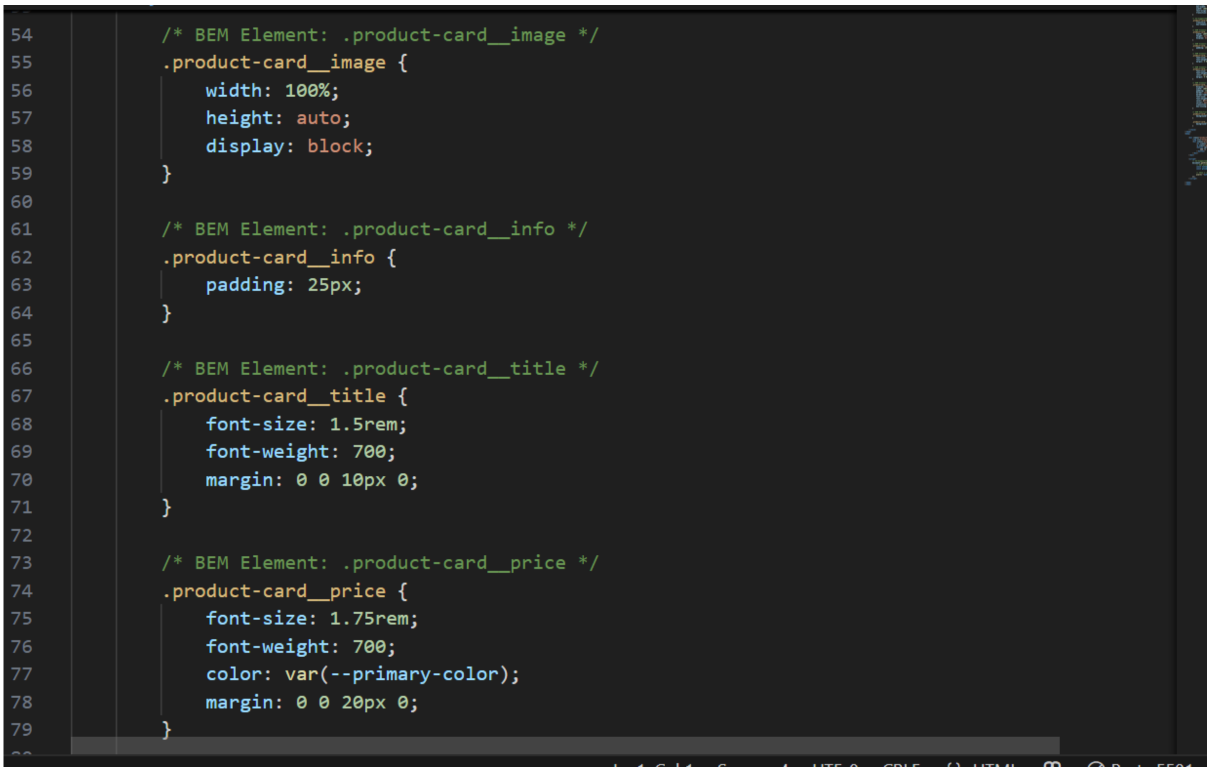
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Product Display Page</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Lato:wght@400;700&display=swap" rel="styl
    <style>
        /* AI-Suggested Design & Base Styles */
        :root {
            --bg-color: #f4f4f4;
            --card-color: #ffffff;
            --text-color: #333333;
            --primary-color: #17a2b8;
            --primary-hover-color: #138496;
            --shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
            --shadow-hover: 0 6px 20px rgba(0, 0, 0, 0.15);
        }

        body {
            font-family: 'Lato', sans-serif;
            background-color: var(--bg-color);
            color: var(--text-color);
            margin: 0;
            /* Use Flexbox to center the product card */
            display: flex;
            justify-content: center;
            align-items: center;
```



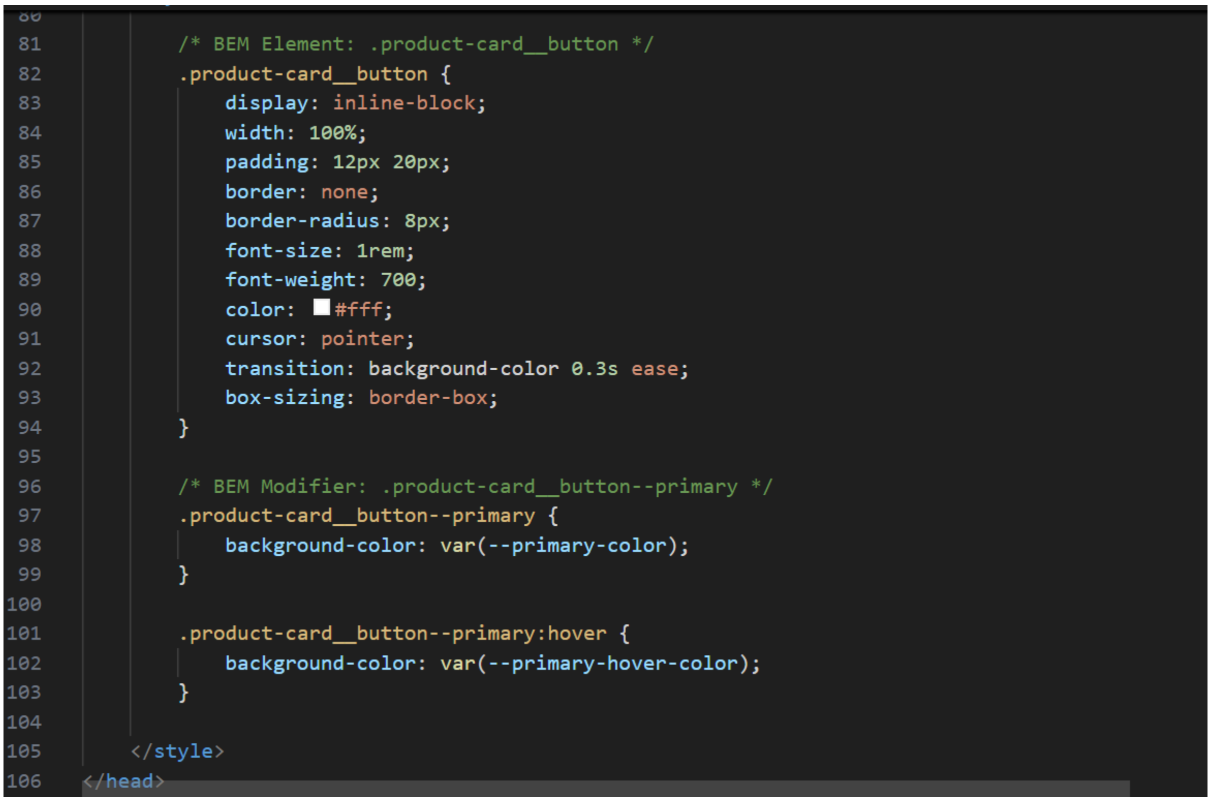
```html
<head>
    <style>
        body {
```

```css
            min-height: 100vh;
            padding: 20px;
            box-sizing: border-box;
        }

        /* BEM Block: .product-card */
        .product-card {
            background-color: var(--card-color);
            border-radius: 12px;
            box-shadow: var(--shadow);
            overflow: hidden;
            max-width: 350px;
            width: 100%;
            text-align: center;
            transition: transform 0.3s ease, box-shadow 0.3s ease;
        }

        /* AI-Suggested Hover Effect */
        .product-card:hover {
            transform: translateY(-8px);
            box-shadow: var(--shadow-hover);
        }

        /* BEM Element: .product-card__image */
        .product-card__image {
            width: 100%;
```

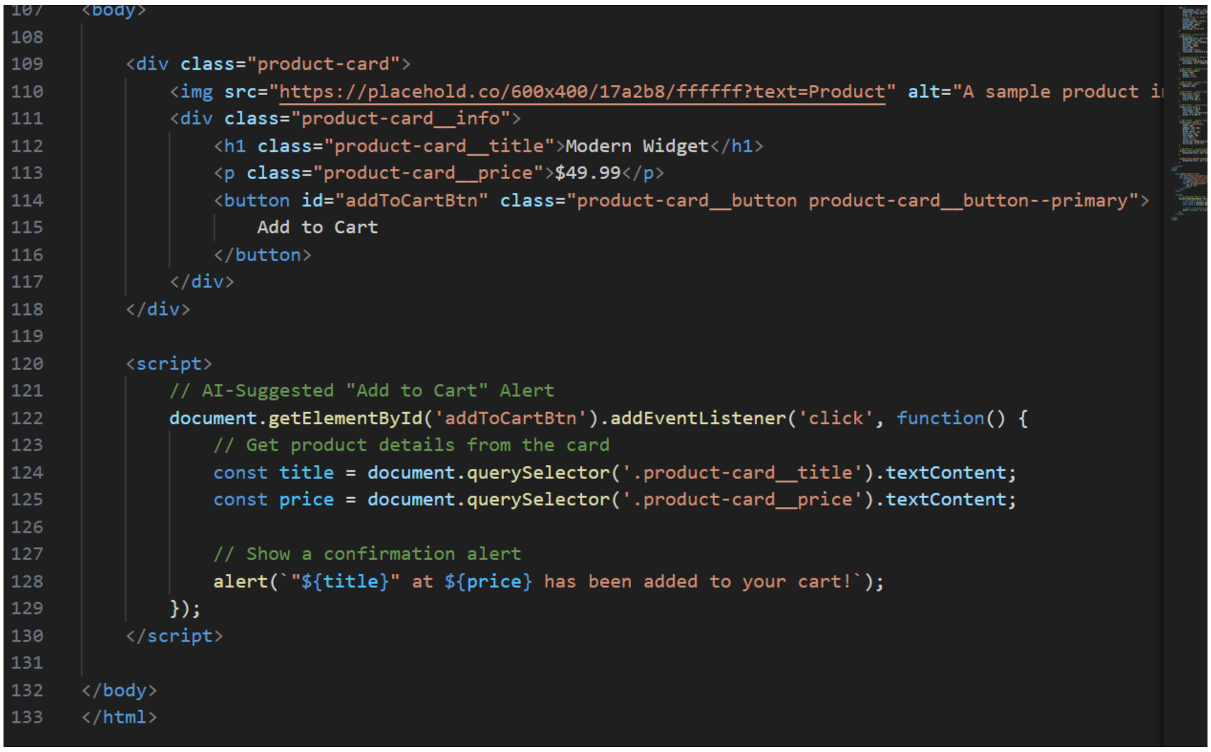
Ln 1, Col 1   Spaces: 4   UTF-8   CRLF   {} HTML   Port : 5501



```
        /* BEM Element: .product-card__image */
        .product-card__image {
            width: 100%;
            height: auto;
            display: block;
        }

        /* BEM Element: .product-card__info */
        .product-card__info {
            padding: 25px;
        }

        /* BEM Element: .product-card__title */
        .product-card__title {
            font-size: 1.5rem;
            font-weight: 700;
            margin: 0 0 10px 0;
        }

        /* BEM Element: .product-card__price */
        .product-card__price {
            font-size: 1.75rem;
            font-weight: 700;
            color: var(--primary-color);
            margin: 0 0 20px 0;
        }
```

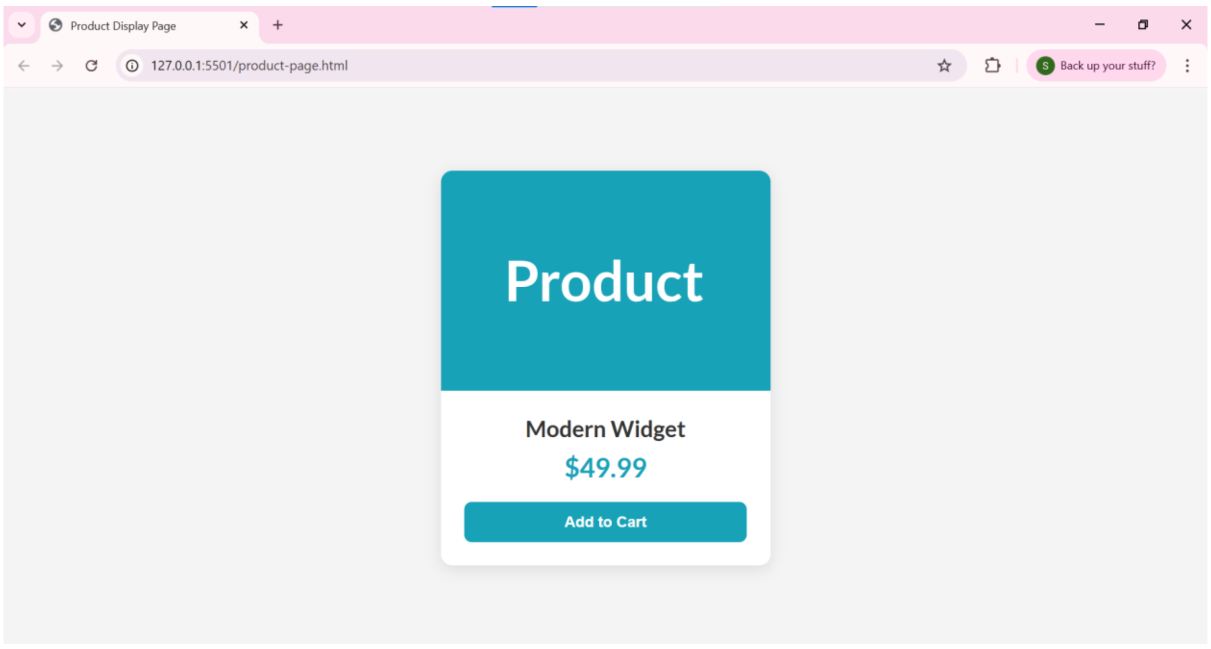


```
        /* BEM Element: .product-card__button */
        .product-card__button {
            display: inline-block;
            width: 100%;
            padding: 12px 20px;
            border: none;
            border-radius: 8px;
            font-size: 1rem;
            font-weight: 700;
            color: #fff;
            cursor: pointer;
            transition: background-color 0.3s ease;
            box-sizing: border-box;
        }

        /* BEM Modifier: .product-card__button--primary */
        .product-card__button--primary {
            background-color: var(--primary-color);
        }

        .product-card__button--primary:hover {
            background-color: var(--primary-hover-color);
        }

    </style>
</head>
```

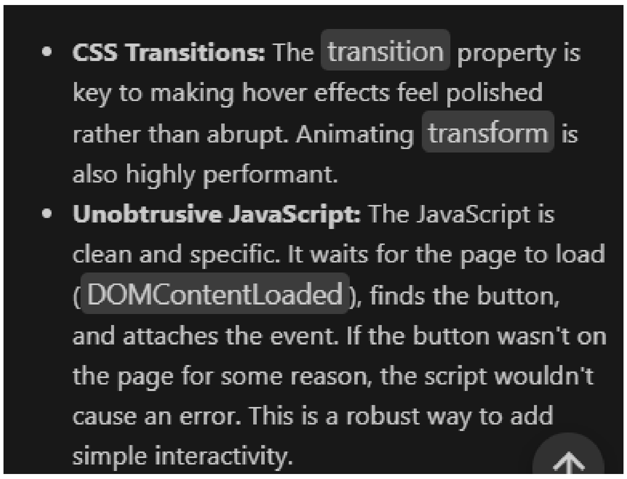


```html
<body>

    <div class="product-card">
        <img src="https://placehold.co/600x400/17a2b8/ffffff?text=Product" alt="A sample product i
        <div class="product-card__info">
            <h1 class="product-card__title">Modern Widget</h1>
            <p class="product-card__price">$49.99</p>
            <button id="addToCartBtn" class="product-card__button product-card__button--primary">
                Add to Cart
            </button>
        </div>
    </div>

    <script>
        // AI-Suggested "Add to Cart" Alert
        document.getElementById('addToCartBtn').addEventListener('click', function() {
            // Get product details from the card
            const title = document.querySelector('.product-card__title').textContent;
            const price = document.querySelector('.product-card__price').textContent;

            // Show a confirmation alert
            alert(`"${title}" at ${price} has been added to your cart!`);
        });
    </script>

</body>
</html>
```

OUTPUT:



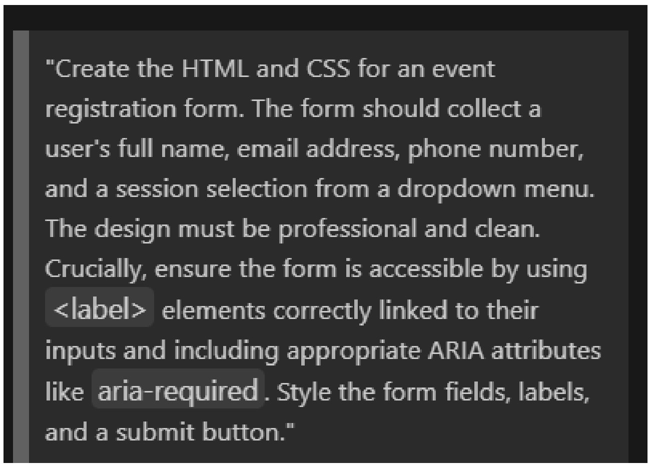
OBSERVATION:



- **CSS Transitions:** The `transition` property is key to making hover effects feel polished rather than abrupt. Animating `transform` is also highly performant.
- **Unobtrusive JavaScript:** The JavaScript is clean and specific. It waits for the page to load (`DOMContentLoaded`), finds the button, and attaches the event. If the button wasn't on the page for some reason, the script wouldn't cause an error. This is a robust way to add simple interactivity.

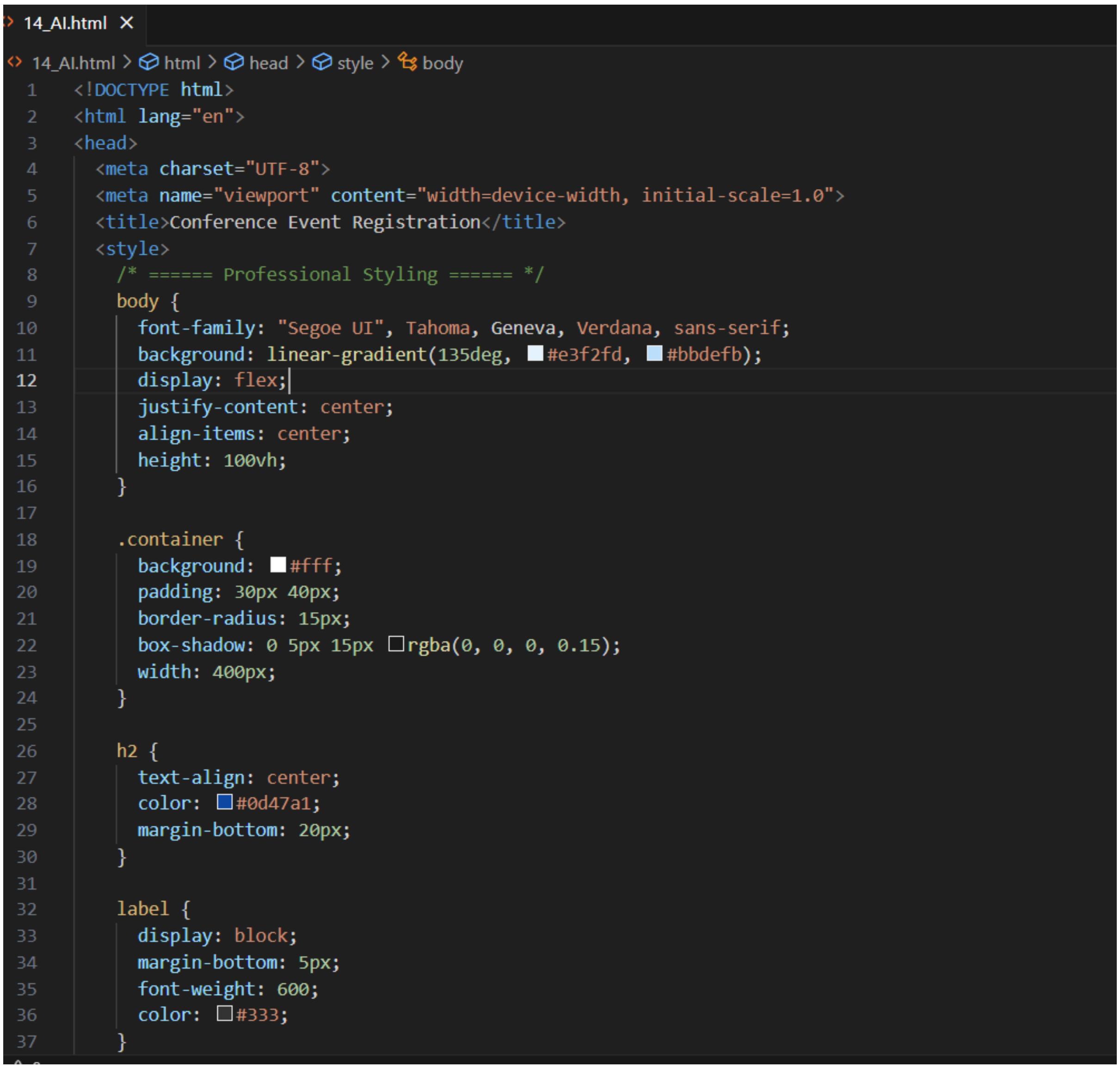
TASK-3: Event Registration Form

PROMPT:



"Create the HTML and CSS for an event registration form. The form should collect a user's full name, email address, phone number, and a session selection from a dropdown menu. The design must be professional and clean. Crucially, ensure the form is accessible by using `<label>` elements correctly linked to their inputs and including appropriate ARIA attributes like `aria-required`. Style the form fields, labels, and a submit button."

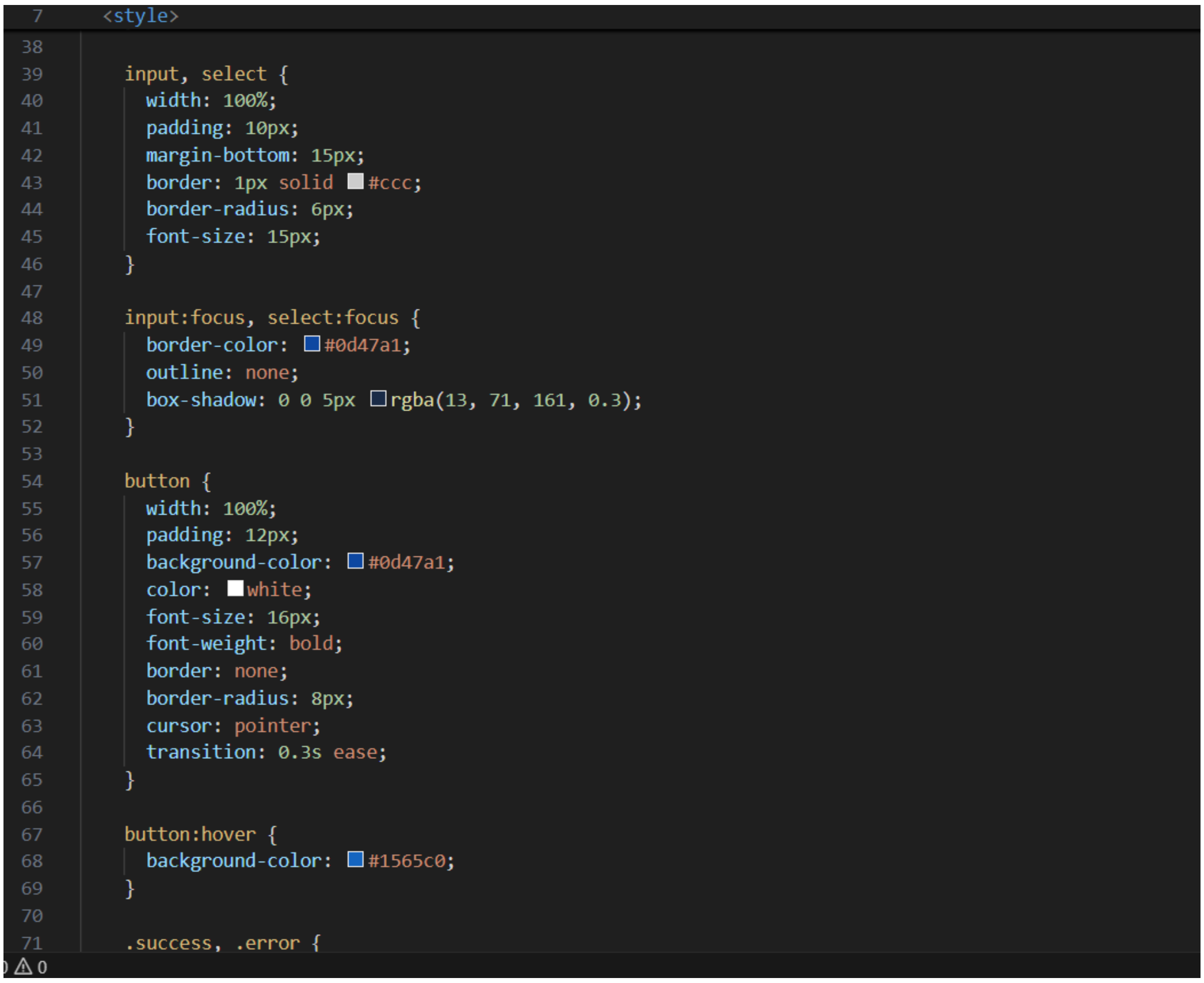
CODE:



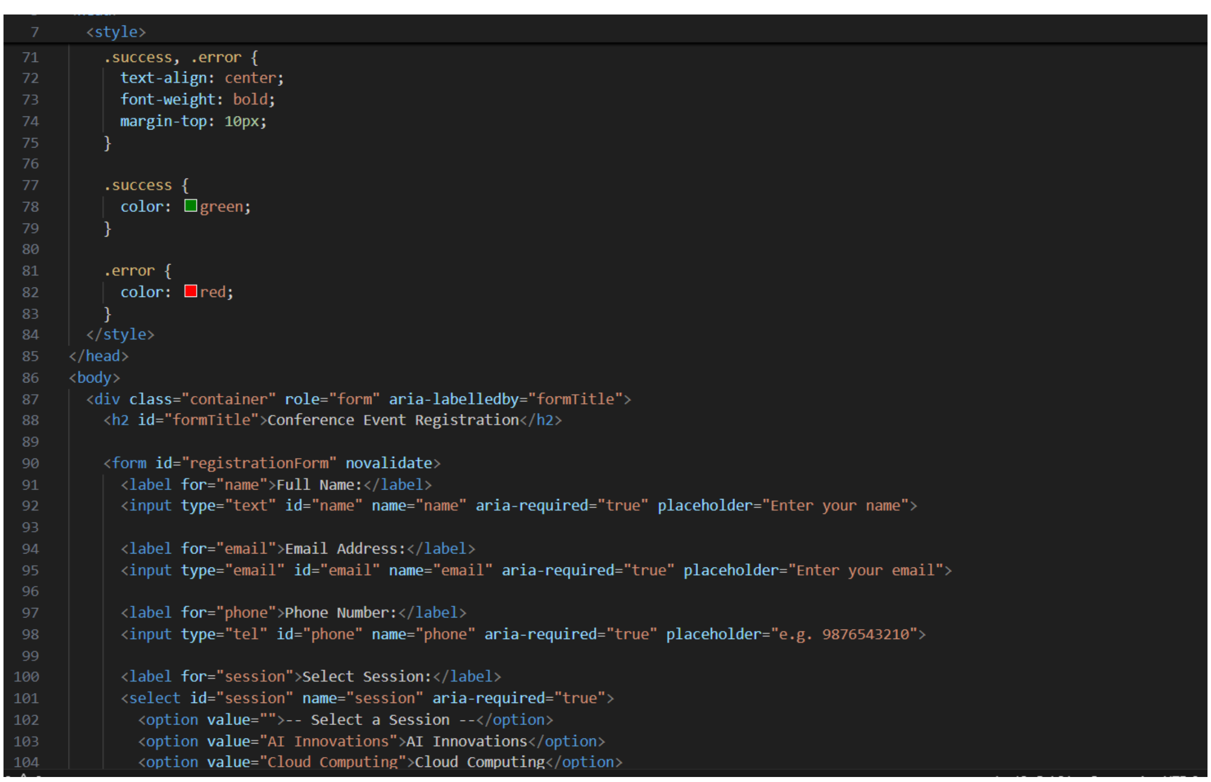
14_AI.html ×

14_AI.html > html > head > style > body

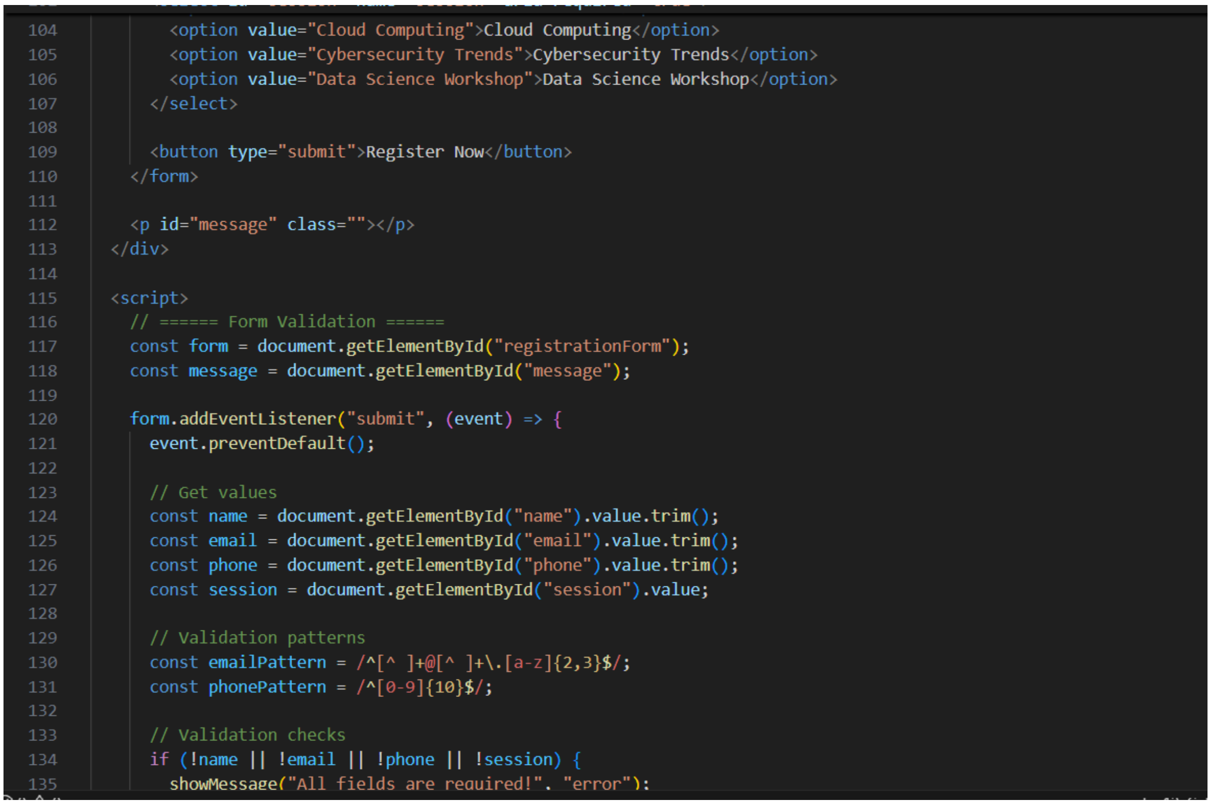
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Conference Event Registration</title>
  <style>
    /* ====== Professional Styling ====== */
    body {
      font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
      background: linear-gradient(135deg, #e3f2fd, #bbdefb);
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
    }

    .container {
      background: #fff;
      padding: 30px 40px;
      border-radius: 15px;
      box-shadow: 0 5px 15px rgba(0, 0, 0, 0.15);
      width: 400px;
    }

    h2 {
      text-align: center;
      color: #0d47a1;
      margin-bottom: 20px;
    }

    label {
      display: block;
      margin-bottom: 5px;
      font-weight: 600;
      color: #333;
    }
```



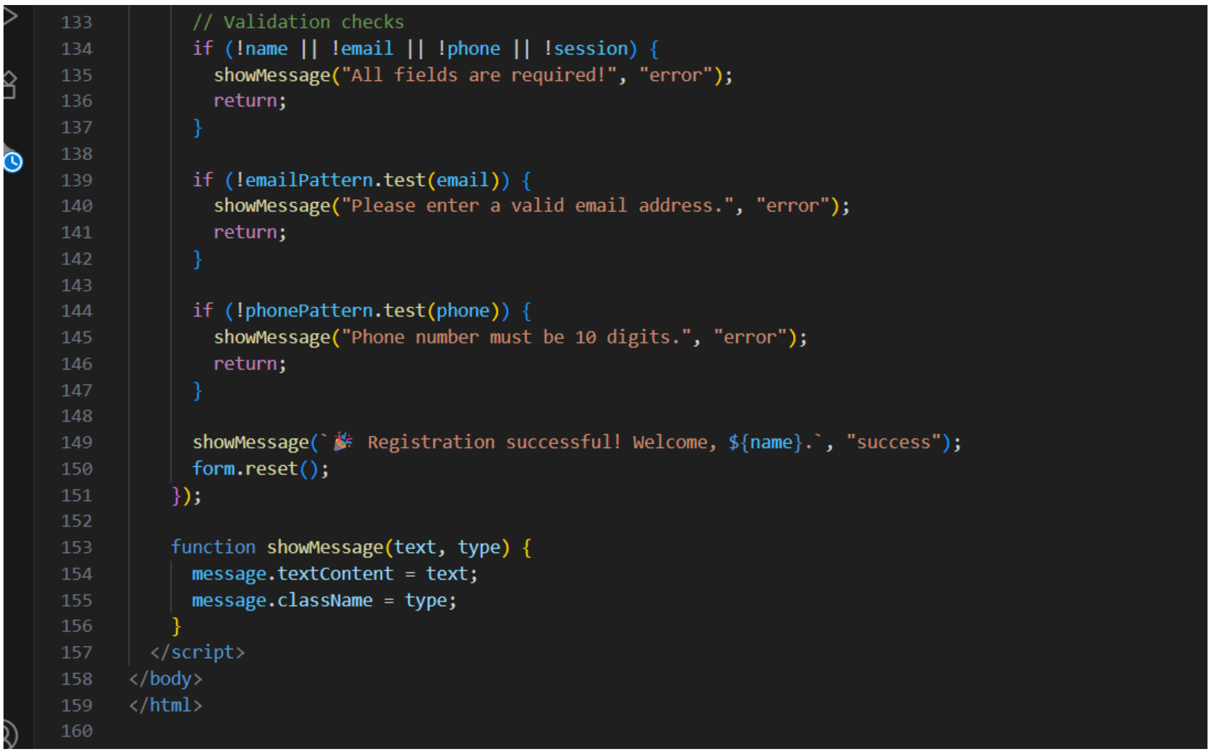
```
  <style>

    input, select {
      width: 100%;
      padding: 10px;
      margin-bottom: 15px;
      border: 1px solid #ccc;
      border-radius: 6px;
      font-size: 15px;
    }

    input:focus, select:focus {
      border-color: #0d47a1;
      outline: none;
      box-shadow: 0 0 5px rgba(13, 71, 161, 0.3);
    }

    button {
      width: 100%;
      padding: 12px;
      background-color: #0d47a1;
      color: white;
      font-size: 16px;
      font-weight: bold;
      border: none;
      border-radius: 8px;
      cursor: pointer;
      transition: 0.3s ease;
    }

    button:hover {
      background-color: #1565c0;
    }

    .success, .error {
```



```html
  <style>
    .success, .error {
      text-align: center;
      font-weight: bold;
      margin-top: 10px;
    }

    .success {
      color: 🟩green;
    }

    .error {
      color: 🟥red;
    }
  </style>
</head>
<body>
  <div class="container" role="form" aria-labelledby="formTitle">
    <h2 id="formTitle">Conference Event Registration</h2>

    <form id="registrationForm" novalidate>
      <label for="name">Full Name:</label>
      <input type="text" id="name" name="name" aria-required="true" placeholder="Enter your name">

      <label for="email">Email Address:</label>
      <input type="email" id="email" name="email" aria-required="true" placeholder="Enter your email">

      <label for="phone">Phone Number:</label>
      <input type="tel" id="phone" name="phone" aria-required="true" placeholder="e.g. 9876543210">

      <label for="session">Select Session:</label>
      <select id="session" name="session" aria-required="true">
        <option value="">-- Select a Session --</option>
        <option value="AI Innovations">AI Innovations</option>
        <option value="Cloud Computing">Cloud Computing</option>
```

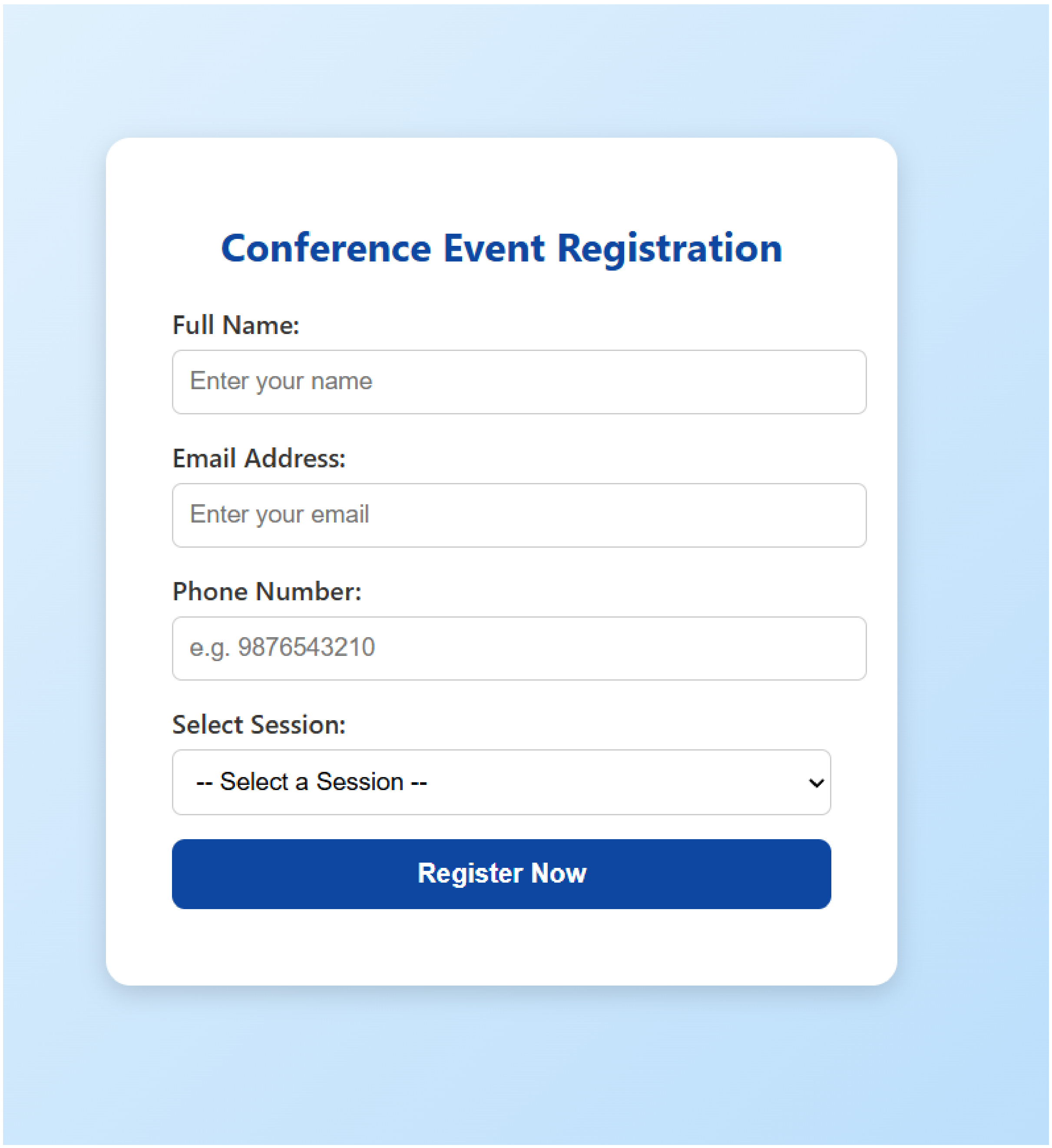


```html
        <option value="Cloud Computing">Cloud Computing</option>
        <option value="Cybersecurity Trends">Cybersecurity Trends</option>
        <option value="Data Science Workshop">Data Science Workshop</option>
      </select>

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

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

  <script>
    // ====== Form Validation ======
    const form = document.getElementById("registrationForm");
    const message = document.getElementById("message");

    form.addEventListener("submit", (event) => {
      event.preventDefault();

      // Get values
      const name = document.getElementById("name").value.trim();
      const email = document.getElementById("email").value.trim();
      const phone = document.getElementById("phone").value.trim();
      const session = document.getElementById("session").value;

      // Validation patterns
      const emailPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
      const phonePattern = /^[0-9]{10}$/;

      // Validation checks
      if (!name || !email || !phone || !session) {
        showMessage("All fields are required!", "error");
```



```
      // Validation checks
      if (!name || !email || !phone || !session) {
        showMessage("All fields are required!", "error");
        return;
      }

      if (!emailPattern.test(email)) {
        showMessage("Please enter a valid email address.", "error");
        return;
      }

      if (!phonePattern.test(phone)) {
        showMessage("Phone number must be 10 digits.", "error");
        return;
      }

      showMessage(`🎉 Registration successful! Welcome, ${name}.`, "success");
      form.reset();
    });

    function showMessage(text, type) {
      message.textContent = text;
      message.className = type;
    }
  </script>
</body>
</html>

```

OUTPUT:


Conference Event Registration

Full Name:

Enter your name

Email Address:

Enter your email

Phone Number:

e.g. 9876543210

Select Session:

-- Select a Session --

OBSERVATION:

Form Interface

- A **centered, professional-looking card** with fields:
  - Full Name
  - Email Address
  - Phone Number
  - Session Selection (dropdown)
- A "**Register Now**" button at the bottom.

Validation Feedback

- If the user:
  - Leaves any field empty → shows "All fields are required!"
  - Enters invalid email → shows "Please enter a valid email address."
  - Enters invalid phone → shows "Phone number must be 10 digits."
- If all inputs are valid → shows ✅ "Registration successful! Welcome, [Name]."

Accessibility Features

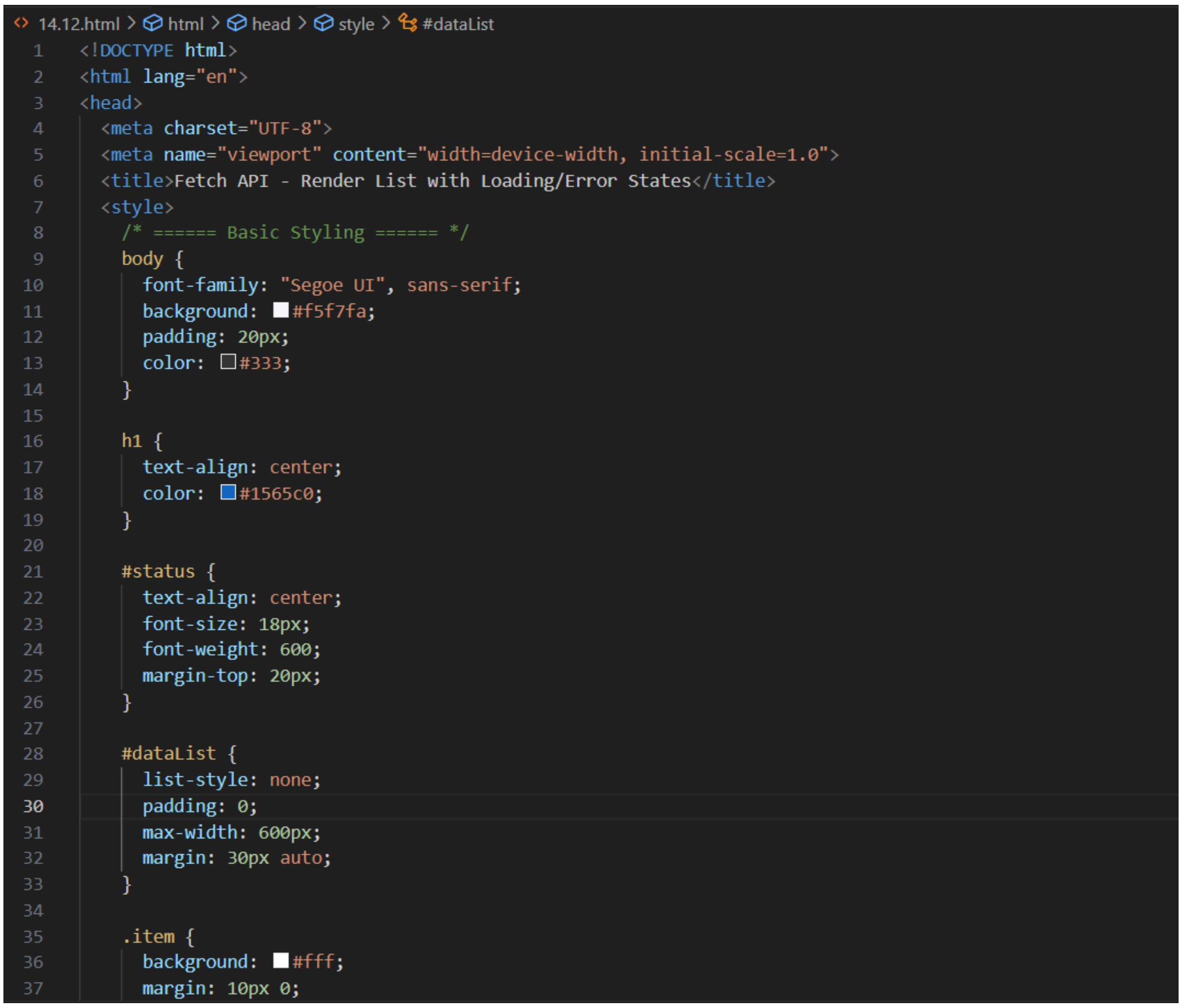
- Each input has a `<label>` connected by `for` attribute.
- `aria-required` ensures screen readers announce mandatory fields.
- `role="form"` and `aria-labelledby` improve navigation for assistive technologies.

TASK-4: (Data – Fetch API & Render List with Loading/Error States)

PROMPT:

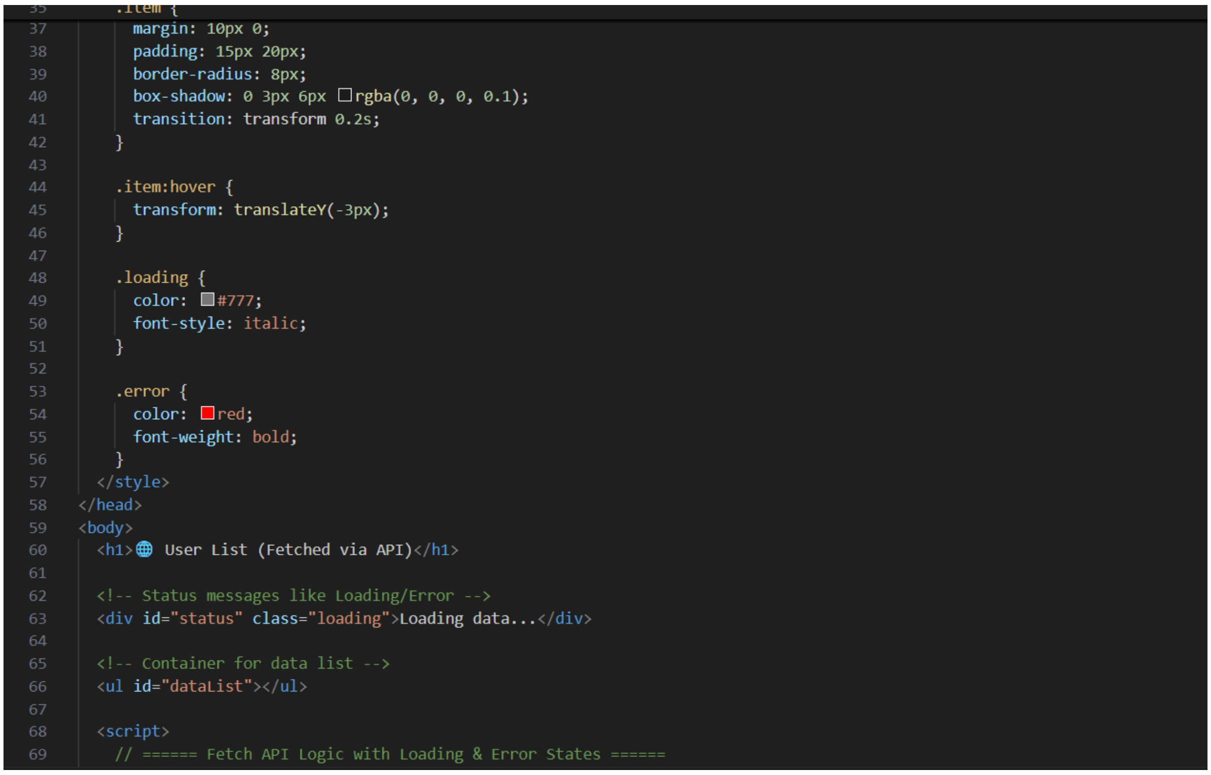
"Write HTML, CSS, and JavaScript code to fetch a JSON list of items (e.g., posts or users) from an API using the Fetch API. Show a loading message while fetching, display the data as a styled list once loaded, and show an error message if the fetch fails."

CODE:



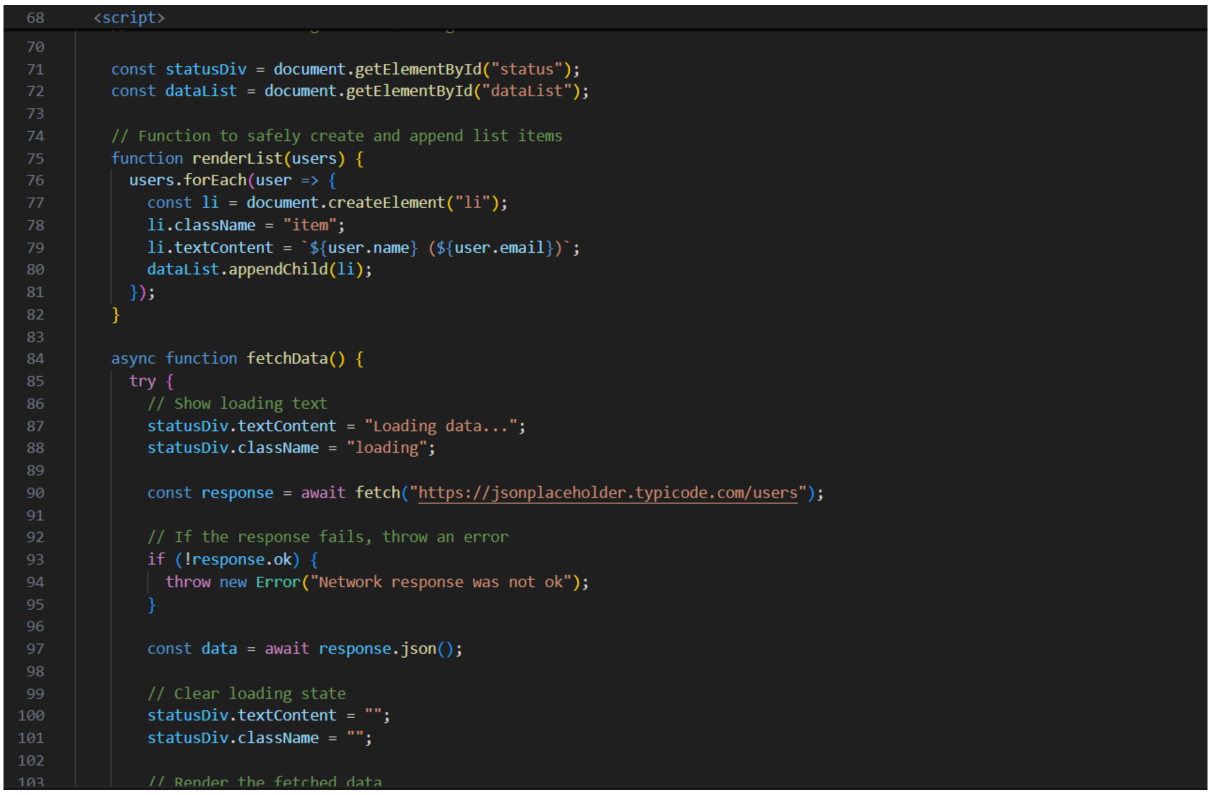
14.12.html > html > head > style > #dataList

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Fetch API - Render List with Loading/Error States</title>
  <style>
    /* ====== Basic Styling ====== */
    body {
      font-family: "Segoe UI", sans-serif;
      background: #f5f7fa;
      padding: 20px;
      color: #333;
    }

    h1 {
      text-align: center;
      color: #1565c0;
    }

    #status {
      text-align: center;
      font-size: 18px;
      font-weight: 600;
      margin-top: 20px;
    }

    #dataList {
      list-style: none;
      padding: 0;
      max-width: 600px;
      margin: 30px auto;
    }

    .item {
      background: #fff;
      margin: 10px 0;
```

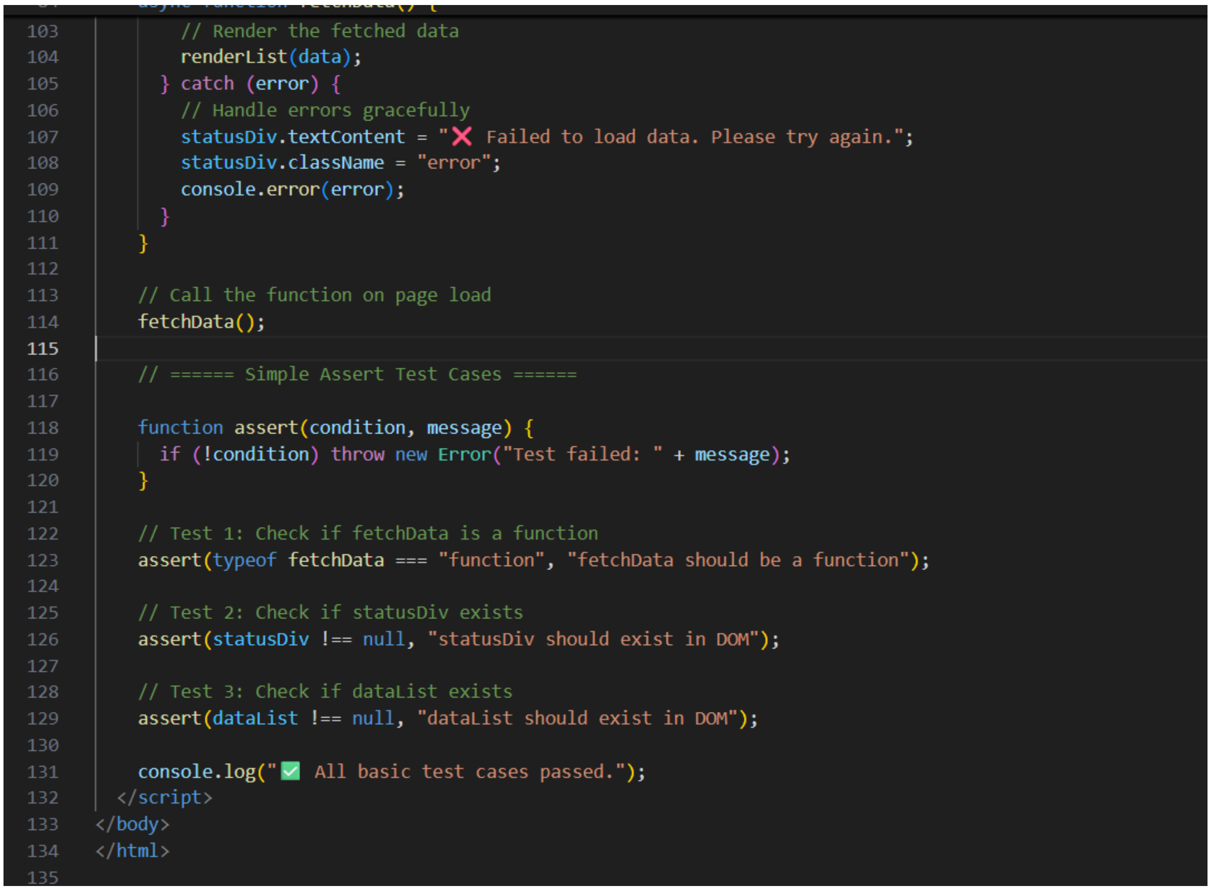


margin: 10px 0;

```html
      padding: 15px 20px;
      border-radius: 8px;
      box-shadow: 0 3px 6px rgba(0, 0, 0, 0.1);
      transition: transform 0.2s;
    }

    .item:hover {
      transform: translateY(-3px);
    }

    .loading {
      color: #777;
      font-style: italic;
    }

    .error {
      color: red;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <h1>🌐 User List (Fetched via API)</h1>

  <!-- Status messages like Loading/Error -->
  <div id="status" class="loading">Loading data...</div>

  <!-- Container for data list -->
  <ul id="dataList"></ul>

  <script>
    // ====== Fetch API Logic with Loading & Error States ======
```

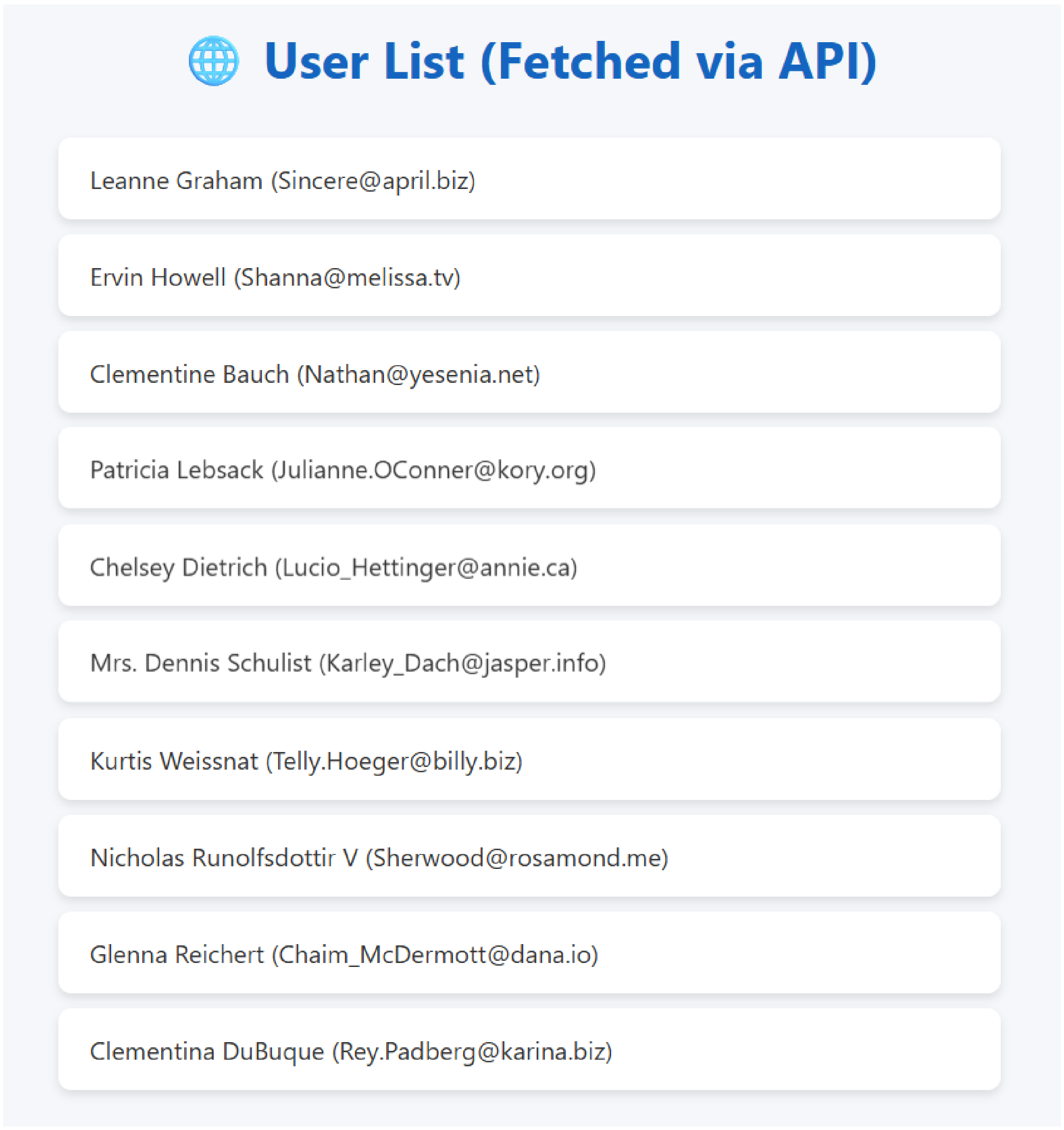


```
  <script>

    const statusDiv = document.getElementById("status");
    const dataList = document.getElementById("dataList");

    // Function to safely create and append list items
    function renderList(users) {
      users.forEach(user => {
        const li = document.createElement("li");
        li.className = "item";
        li.textContent = `${user.name} (${user.email})`;
        dataList.appendChild(li);
      });
    }

    async function fetchData() {
      try {
        // Show loading text
        statusDiv.textContent = "Loading data...";
        statusDiv.className = "loading";

        const response = await fetch("https://jsonplaceholder.typicode.com/users");

        // If the response fails, throw an error
        if (!response.ok) {
          throw new Error("Network response was not ok");
        }

        const data = await response.json();

        // Clear loading state
        statusDiv.textContent = "";
        statusDiv.className = "";

```



```
        // Render the fetched data
        renderList(data);
      } catch (error) {
        // Handle errors gracefully
        statusDiv.textContent = "❌ Failed to load data. Please try again.";
        statusDiv.className = "error";
        console.error(error);
      }
    }

    // Call the function on page load
    fetchData();

    // ====== Simple Assert Test Cases ======

    function assert(condition, message) {
      if (!condition) throw new Error("Test failed: " + message);
    }

    // Test 1: Check if fetchData is a function
    assert(typeof fetchData === "function", "fetchData should be a function");

    // Test 2: Check if statusDiv exists
    assert(statusDiv !== null, "statusDiv should exist in DOM");

    // Test 3: Check if dataList exists
    assert(dataList !== null, "dataList should exist in DOM");

    console.log("✅ All basic test cases passed.");
  </script>
</body>
</html>

```

OUTPUT:



🌐 User List (Fetched via API)

Leanne Graham (Sincere@april.biz)

Ervin Howell (Shanna@melissa.tv)

Clementine Bauch (Nathan@yesenia.net)

Patricia Lebsack (Julianne.OConner@kory.org)

Chelsey Dietrich (Lucio_Hettinger@annie.ca)

Mrs. Dennis Schulist (Karley_Dach@jasper.info)

Kurtis Weissnat (Telly.Hoeger@billy.biz)

Nicholas Runolfsdottir V (Sherwood@rosamond.me)

Glenna Reichert (Chaim_McDermott@dana.io)

Clementina DuBuque (Rey.Padberg@karina.biz)

OBSERVATION:

Purpose

This project demonstrates how to:

- Fetch and render data using the **Fetch API**.
- Provide a **user-friendly experience** with **loading** and **error** indicators.
- Ensure the DOM is updated dynamically after data retrieval.

Functionality

1. When the page loads, a "**Loading data...**" message appears.
2. The script fetches user data from the sample API:
   `https://jsonplaceholder.typicode.com/users`
3. On success:
   - The data is rendered as a styled list (name + email).
4. On failure:
   - Displays an **error message**: "❌ Failed to load data. Please try again."

Accessibility

- Uses semantic HTML (`<ul>` and `<li>` for lists).
- Visually distinguishes states (loading/error) for clarity.

THANK YOU