AI-Assisted Coding

Lab Assignment -14.1

Name: Neha Shazneen

Roll no:2403A510A1

Batch no:05

CSE 2nd Year

Task 1 – Portfolio Website Design

You are building a personal portfolio website to showcase your work.

Requirements:

• Create sections for About Me, Projects, and Contact.

• Use AI to:

o Suggest color palettes and typography. o Create a responsive layout with Grid/Flexbox.

o Add smooth scrolling navigation.

Prompt:

Build a personal portfolio website with these features: •

Sections: **About Me**, **Projects**, and **Contact**

- Use **AI to suggest**:
  - A modern **color palette** and **typography** ○ A **responsive layout** using **Grid/Flexbox** ○ **Smooth scrolling** navigation between sections

Add simple hover effects, a contact form, and make sure the site looks good on both desktop and mobile.

Code Generated:

14.1.task1.html > ...

```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>
  <style>
    /* Typography: Using Google Fonts */
    @import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&family=Open+Sans:wght@400;600&display=swap');

    :root {
      /* Color Palette: AI Suggested (modern + professional) */
      --primary-color: #2C3E50;
      --accent-color: #E67E22;
      --bg-color: #F4F4F4;
      --text-color: #333;
    }

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      scroll-behavior: smooth;
    }
```

ve\Desktop\AI_Assisted_Coding\task2.py

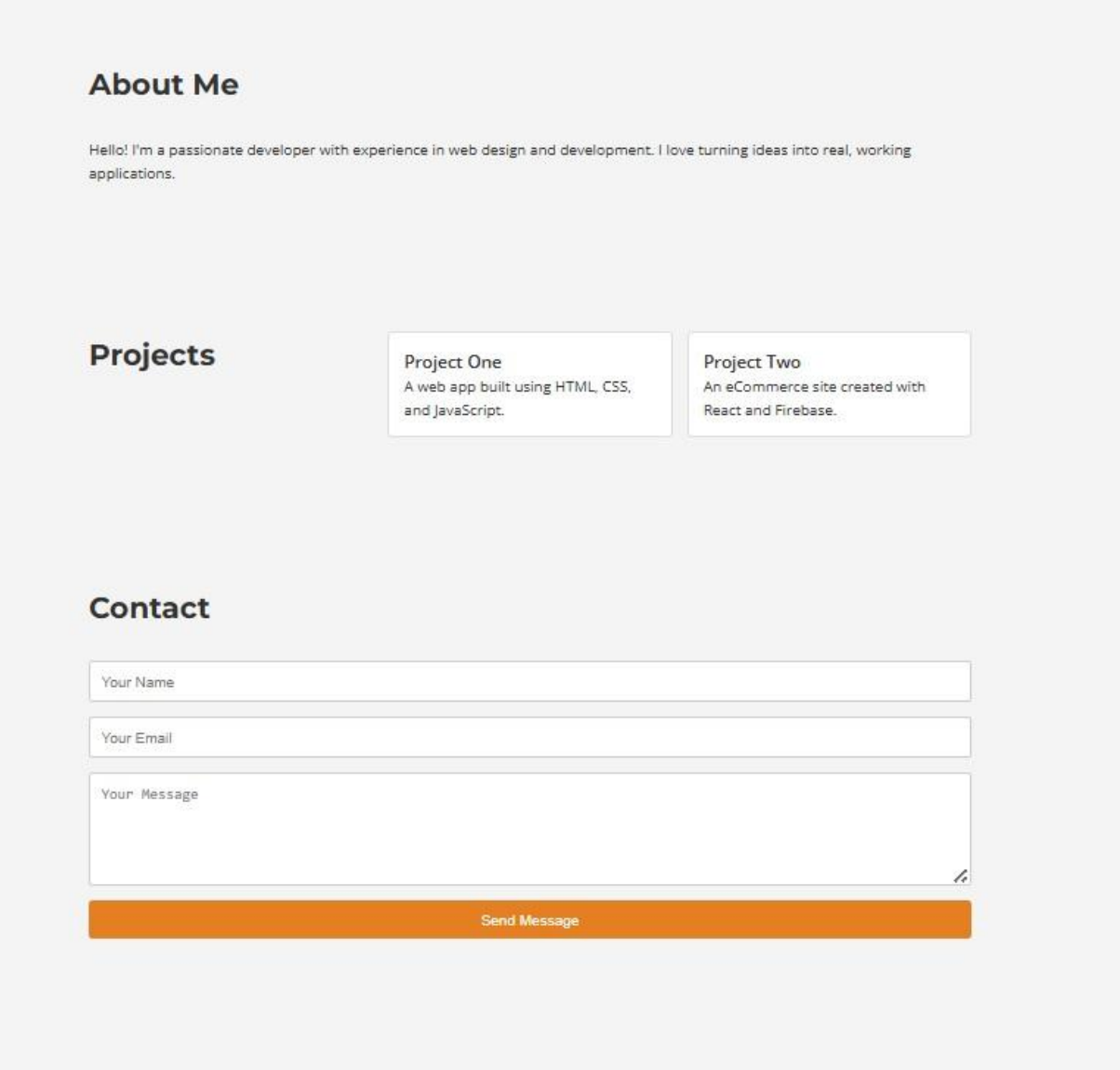
```
      font-family: 'Open Sans', sans-serif;
      background-color: var(--bg-color);
      color: var(--text-color);
      line-height: 1.6;
    }

    header {
      background: var(--primary-color);
      color: white;
      padding: 1rem 2rem;
      position: fixed;
      top: 0;
      width: 100%;
      z-index: 1000;
    }

    nav {
      display: flex;
      justify-content: space-between;
      align-items: center;
    }

    nav a {
      color: white;
      margin-left: 1.5rem;
      text-decoration: none;
      font-weight: 600;
    }

    nav a:hover {
      color: var(--accent-color);
    }

    section {
      padding: 100px 20px 60px;
      max-width: 1000px;
      margin: auto;
    }

    h2 {
      font-family: 'Montserrat', sans-serif;
      font-size: 2rem;
      margin-bottom: 1rem;
    }
```

```
}

.about,
.projects,
.contact {
  display: grid;
  gap: 1rem;
}

.projects .project {
  background: white;
  padding: 1rem;
  border: 1px solid #ddd;
  border-radius: 5px;
  transition: transform 0.3s ease;
}

.projects .project:hover {
  transform: scale(1.02);
}

form {
  display: flex;
  flex-direction: column;
  gap: 1rem;
}

input, textarea {
  padding: 0.75rem;
  border: 1px solid #ccc;
  border-radius: 4px;
  font-size: 1rem;
}
```

```
    input[type="submit"] {
      background-color: var(--accent-color);
      color: white;
      border: none;
      cursor: pointer;
    }

    input[type="submit"]:hover {
      background-color: #cf6e17;
    }

    /* Responsive Layout */
    @media (min-width: 768px) {
      .projects {
        grid-template-columns: repeat(2, 1fr);
      }
    }

    @media (min-width: 1024px) {
      .projects {
        grid-template-columns: repeat(3, 1fr);
      }
    }

    /* Space at top to account for fixed header */
    .spacer {
      height: 80px;
    }
  </style>
</head>
<body>

```

```
<header>
  <nav>
    <div><strong>My Portfolio</strong></div>
    <div>
      <a href="#about">About Me</a>
      <a href="#projects">Projects</a>
      <a href="#contact">Contact</a>
    </div>
  </nav>
</header>

<div class="spacer"></div>

<section id="about" class="about">
  <h2>About Me</h2>
  <p>Hello! I'm a passionate developer with experience in web design and development. I love turning ideas into real, working applications.</p>
</section>

<section id="projects" class="projects">
  <h2>Projects</h2>
  <div class="project">
    <h3>Project One</h3>
    <p>A web app built using HTML, CSS, and JavaScript.</p>
  </div>
  <div class="project">
    <h3>Project Two</h3>
    <p>An eCommerce site created with React and Firebase.</p>
  </div>
  <!-- Add more projects here -->
</section>

<section id="contact" class="contact">
  <h2>Contact</h2>
  <form>
    <input type="text" placeholder="Your Name" required />
    <input type="email" placeholder="Your Email" required />
    <textarea rows="5" placeholder="Your Message" required></textarea>
    <input type="submit" value="Send Message" />
  </form>
</section>

</body>
</html>
```

Output:



Observation:

1. Section Creation

- **About Me**, **Projects**, and **Contact** sections are clearly defined using <section> elements with IDs for navigation.

2. AI-Enhanced Design Suggestions

- **Color Palette**: A clean, professional palette with a dark blue (#2C3E50), accent orange (#E67E22), and a light gray background (#F4F4F4)—suggested for modern portfolios.
- **Typography**: Google Fonts Montserrat (for headings) and Open Sans (for body text) are used for clean, readable design.

3. Responsive Layout

- **CSS Grid** is used for the .projects section.
- Media queries adjust the grid layout for tablets and desktops.
- The layout adapts to different screen sizes, maintaining readability and structure.

4. Smooth Scrolling Navigation

- scroll-behavior: smooth is applied globally using the * selector.
- The nav links use anchor tags (href="#section-id") to allow smooth transitions when clicked.

5. Flexbox for Navigation

- Flexbox is used in the nav element to align the brand title and navigation links horizontally and responsively.

Task 2 – Online Store Product Page

Design a product display page for an online store. Requirements:

- Display product image, title, price, and "Add to Cart" button.
- Use AI to:

o Style with BEM methodology. o Make layout responsive.
o Add hover effects and "Add to Cart" alert

Prompt

Design a responsive product display page for an online store.
Requirements:

- Show product image, title, price, and an "Add to Cart" button.
- Use AI to:
  - Apply BEM methodology for CSS class naming. ○ Make layout responsive with Flexbox/Grid.
  - Add hover effects and show an alert when "Add to Cart" is clicked.

Code Generated:

14.1.task2.html > ...

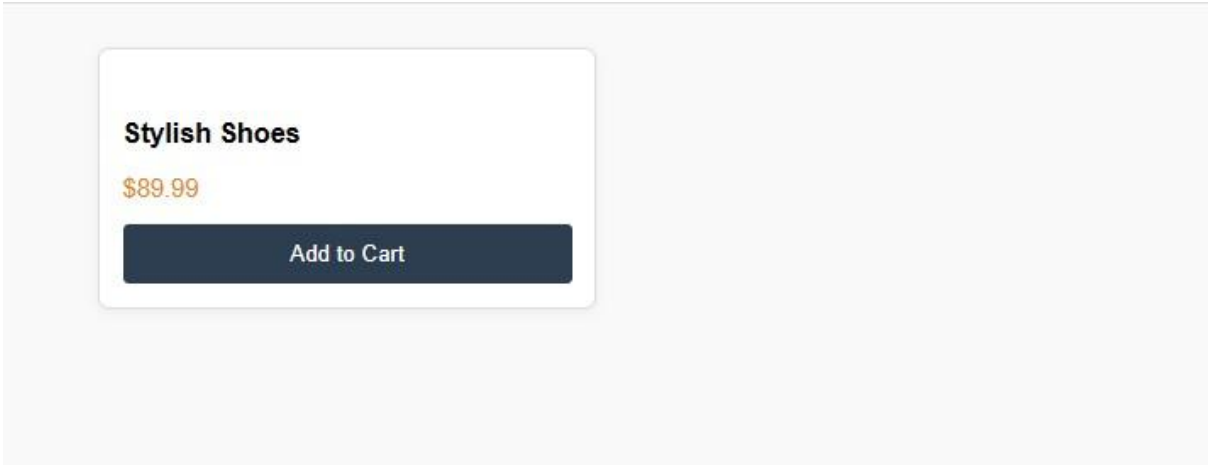
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Product Page</title>
  <style>
    /* BEM-style CSS and AI-Suggested Styling */
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f9f9f9;
    }

    .product-page {
      display: flex;
      flex-direction: column;
      align-items: center;
      padding: 2rem;
    }

    .product-card {
      background-color: #fff;
      border: 1px solid #ddd;
      border-radius: 8px;
      max-width: 350px;
      width: 100%;
      box-shadow: 0 2px 8px rgba(0, 0, 0, 0.05);
      overflow: hidden;
      transition: transform 0.3s ease;
    }

    .product-card:hover {
      transform: scale(1.02);
    }
```

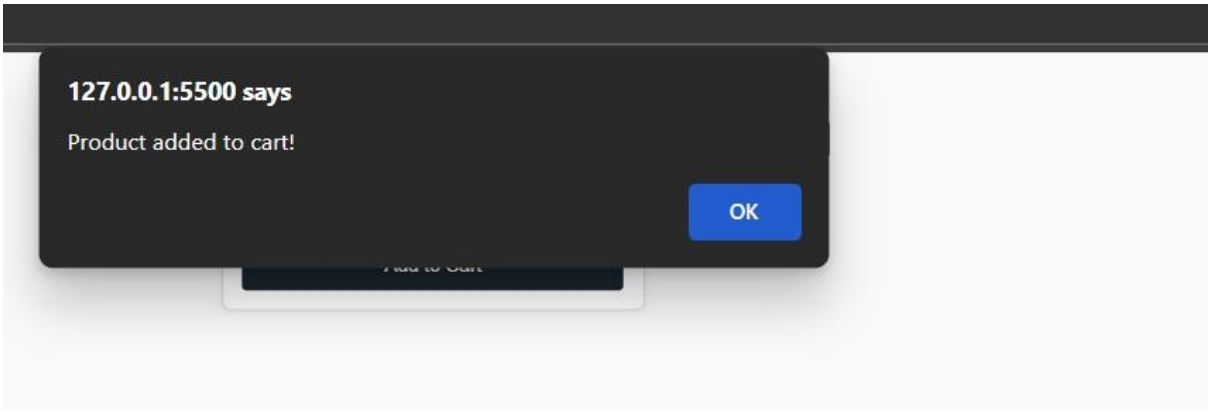
```css
.product-card__image {
  width: 100%;
  height: auto;
}

.product-card__content {
  padding: 1rem;
}

.product-card__title {
  font-size: 1.2rem;
  font-weight: bold;
  margin-bottom: 0.5rem;
}

.product-card__price {
  color: #e67e22;
  font-size: 1.1rem;
  margin-bottom: 1rem;
}

.product-card__button {
  background-color: #2c3e50;
  color: #fff;
  padding: 0.75rem 1rem;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 1rem;
  width: 100%;
  transition: background-color 0.3s;
}
```

```
    .product-card__button:hover {
      background-color: #1a252f;
    }

    /* Responsive */
    @media (min-width: 768px) {
      .product-page {
        flex-direction: row;
        justify-content: center;
      }
    }
  </style>
</head>
<body>

  <main class="product-page">
    <div class="product-card">
      <img src="https://via.placeholder.com/350x250" alt="Product" class="product-card__image" />
      <div class="product-card__content">
        <h2 class="product-card__title">Stylish Shoes</h2>
        <p class="product-card__price">$89.99</p>
        <button class="product-card__button" onclick="addToCart()">Add to Cart</button>
      </div>
    </div>
  </main>

  <script>
    function addToCart() {
      alert("Product added to cart!");
    }
  </script>
```

```
</body>
</html>
```

Output:





Observation:

Required Elements Present

- Shows product **image**, **title**, **price**, and **Add to Cart** button.

2. BEM Methodology Used

- Class names follow BEM (Block__Element format), e.g., product-card__title, product-card__button.

3. Responsive Layout

- Mobile-first design with Flexbox.
- Switches to horizontal layout on wider screens (@media breakpoint at 768px).

4. Interactivity

- Hover effect on the card (transform: scale(1.02)).
- Button hover changes background.
- Clicking **"Add to Cart"** triggers a simple JavaScript alert.

Task 3 – Event Registration Form

Build an event registration form for a conference.

Requirements:

- Collect name, email, phone number, and session selection.
- Use AI to:

o Add form validation with JavaScript.

o Make the form accessible with labels and ARIA.

o Style with a professional look. **Prompt:**

Create an **event registration form** for a conference.

Requirements:

- Collect: **Name**, **Email**, **Phone Number**, and **Session Selection**
- Use **AI to**:

o Add **JavaScript form validation** ○ Ensure **accessibility** with proper labels and ARIA attributes ○ Apply **professional CSS styling**

Code Generated:

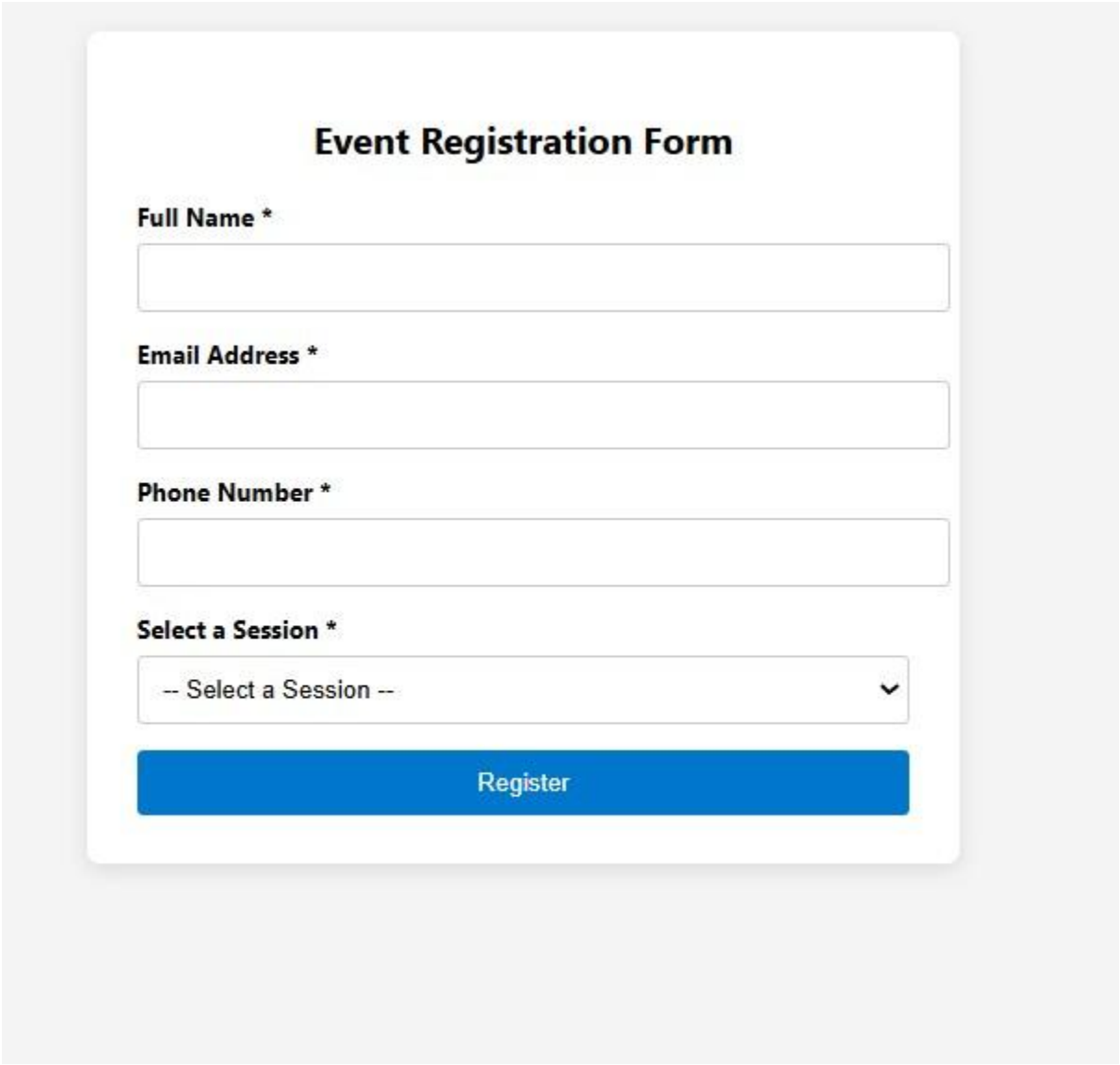
14.1.task3.html > ...

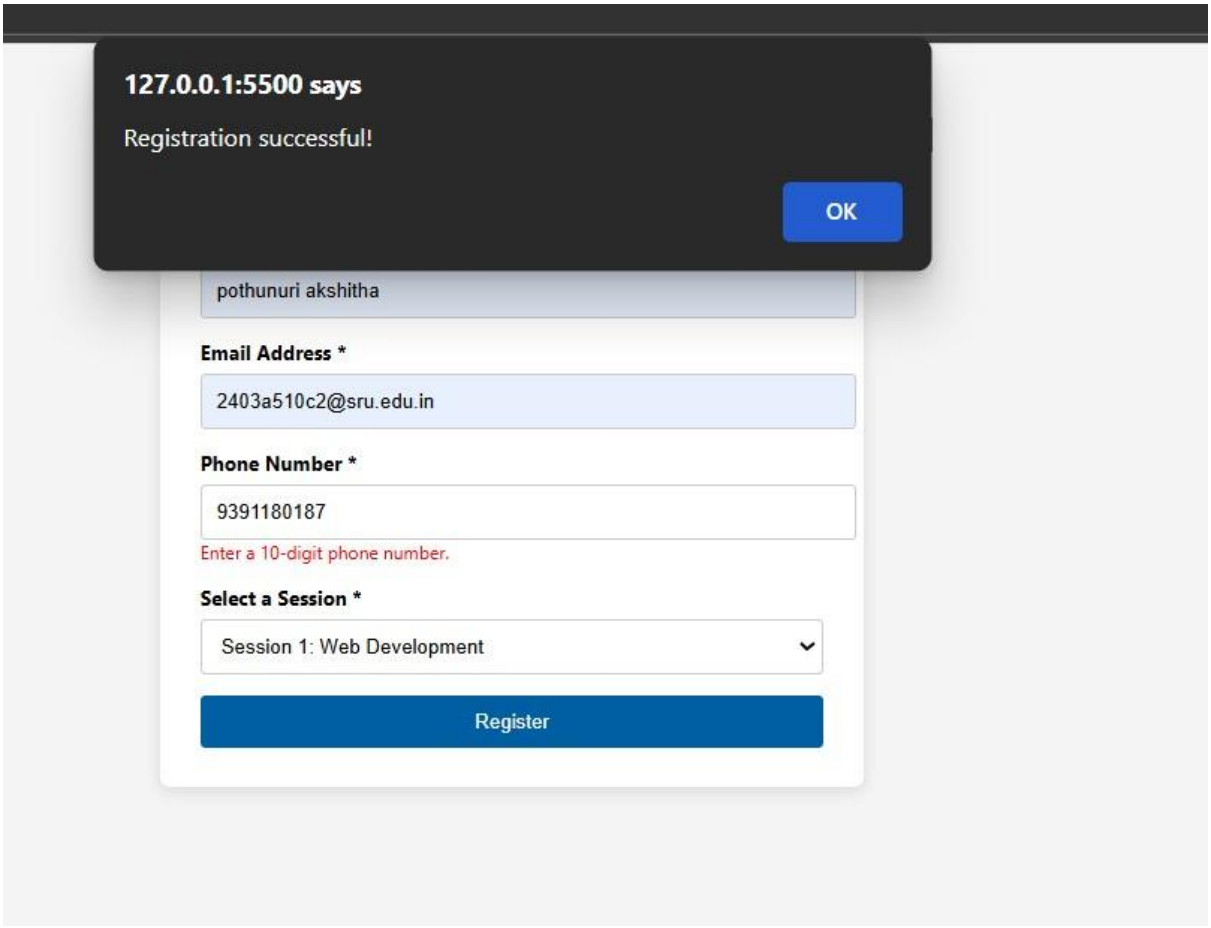
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Conference Registration</title>
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background-color: #f4f4f4;
      padding: 2rem;
    }

    .form-container {
      background: #fff;
      max-width: 500px;
      margin: auto;
      padding: 2rem;
      border-radius: 8px;
      box-shadow: 0 4px 12px rgba(0,0,0,0.1);
    }

    .form-container h2 {
      text-align: center;
      margin-bottom: 1.5rem;
    }

    .form-group {
      margin-bottom: 1rem;
    }

    label {
      display: block;
      margin-bottom: 0.5rem;
      font-weight: bold;
    }

    input, select {
      width: 100%;
```

```
      padding: 0.75rem;
      border: 1px solid #ccc;
      border-radius: 4px;
      font-size: 1rem;
    }

    input:focus, select:focus {
      border-color: #0077cc;
      outline: none;
    }

    .error-message {
      color: red;
      font-size: 0.9rem;
      display: none;
    }

    button {
      width: 100%;
      padding: 0.75rem;
      background-color: #0077cc;
      color: white;
      font-size: 1rem;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }

    button:hover {
      background-color: #005fa3;
    }
  </style>
</head>
<body>

  <div class="form-container" role="form" aria-labelledby="form-title">
    <h2 id="form-title">Event Registration Form</h2>
    <form id="registrationForm" novalidate>
      <div class="form-group">
        <label for="name">Full Name *</label>
        <input type="text" id="name" name="name" aria-required="true" required />
        <div class="error-message" id="nameError">Please enter your name.</div>
      </div>

      <div class="form-group">
        <label for="email">Email Address *</label>
        <input type="email" id="email" name="email" aria-required="true" required />
        <div class="error-message" id="emailError">Please enter a valid email.</div>
      </div>

      <div class="form-group">
        <label for="phone">Phone Number *</label>
        <input type="tel" id="phone" name="phone" pattern="[0-9]{10}" aria-required="true" required />
        <div class="error-message" id="phoneError">Enter a 10-digit phone number.</div>
      </div>

      <div class="form-group">
        <label for="session">Select a Session *</label>
        <select id="session" name="session" aria-required="true" required>
          <option value="">-- Select a Session --</option>
          <option value="session1">Session 1: Web Development</option>
          <option value="session2">Session 2: AI & ML</option>
          <option value="session3">Session 3: Cybersecurity</option>
        </select>
        <div class="error-message" id="sessionError">Please select a session.</div>
      </div>

      <button type="submit">Register</button>
    </form>
  </div>
```

```
110
111  <script>
112    const form = document.getElementById('registrationForm');
113
       Complexity is 6 It's time to do something...
114    form.addEventListener('submit', function (e) {
115      e.preventDefault();
116      let valid = true;
117
118      // Name validation
119      const name = document.getElementById('name');
120      const nameError = document.getElementById('nameError');
121      if (name.value.trim() === '') {
122        nameError.style.display = 'block';
123        valid = false;
124      } else {
125        nameError.style.display = 'none';
126      }
127
128      // Email validation
129      const email = document.getElementById('email');
130      const emailError = document.getElementById('emailError');
131      const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
132      if (!emailRegex.test(email.value)) {
133        emailError.style.display = 'block';
134        valid = false;
135      } else {
136        emailError.style.display = 'none';
137      }
138
139      // Phone validation
140      const phone = document.getElementById('phone');
141      const phoneError = document.getElementById('phoneError');
142      const phoneRegex = /^\d{10}$/;
143      if (!phoneRegex.test(phone.value)) {
144        phoneError.style.display = 'block';
145        valid = false;
146      } else {
147        phoneError.style.display = 'none';
148      }
149
150      // Session validation
151      const session = document.getElementById('session');
152      const sessionError = document.getElementById('sessionError');
153      if (session.value === '') {
154        sessionError.style.display = 'block';
155        valid = false;
156      } else {
157        sessionError.style.display = 'none';
158      }
159
160      // Final submission
161      if (valid) {
162        alert("Registration successful!");
163        form.reset();
164      }
165    });
166  </script>
167  </body>
168  </html>
169
```

Output:


Event Registration Form
Full Name *
Email Address *
Phone Number *
Select a Session *
-- Select a Session --
Register



Observation:

1. Form Requirements Met

- Captures: **Name**, **Email**, **Phone**, **Session Selection** using input and select fields.
- Each input has a <label> for accessibility and ARIA roles where applicable.

2. Form Validation with JavaScript

- Validates all fields:
  - Name: not empty ○
    Email: valid format ○
    Phone: 10-digit number
  - Session: option selected
- Shows error messages for invalid input.
- Displays alert("Registration successful!") on valid submit.

3. Accessibility Features

- Labels are correctly linked to inputs via for and id.
- ARIA attributes like aria-required and aria-labelledby used.
- role="form" used for screen readers.

4. Professional Styling

- Clean, modern UI with spacing, borders, shadows, and focus states.
- Uses responsive, accessible input styles and button hover effect.

Task Description #4 (Data – Fetch API & Render List with Loading/Error States)

- Task: Fetch JSON from an API and render items to the DOM with loading and error UI.
- Instructions: o Ask AI to write fetch() logic, create DOM nodes safely, and add skeleton/loading text.

Prompt

Write code to **fetch JSON data** from an API and **render a list of items** to the DOM.
Use AI to:

- Write clean fetch() logic with proper **error handling**
- Create **DOM elements safely** • Show **loading text or skeleton UI** while data is loading
- Show an **error message** if fetch fails

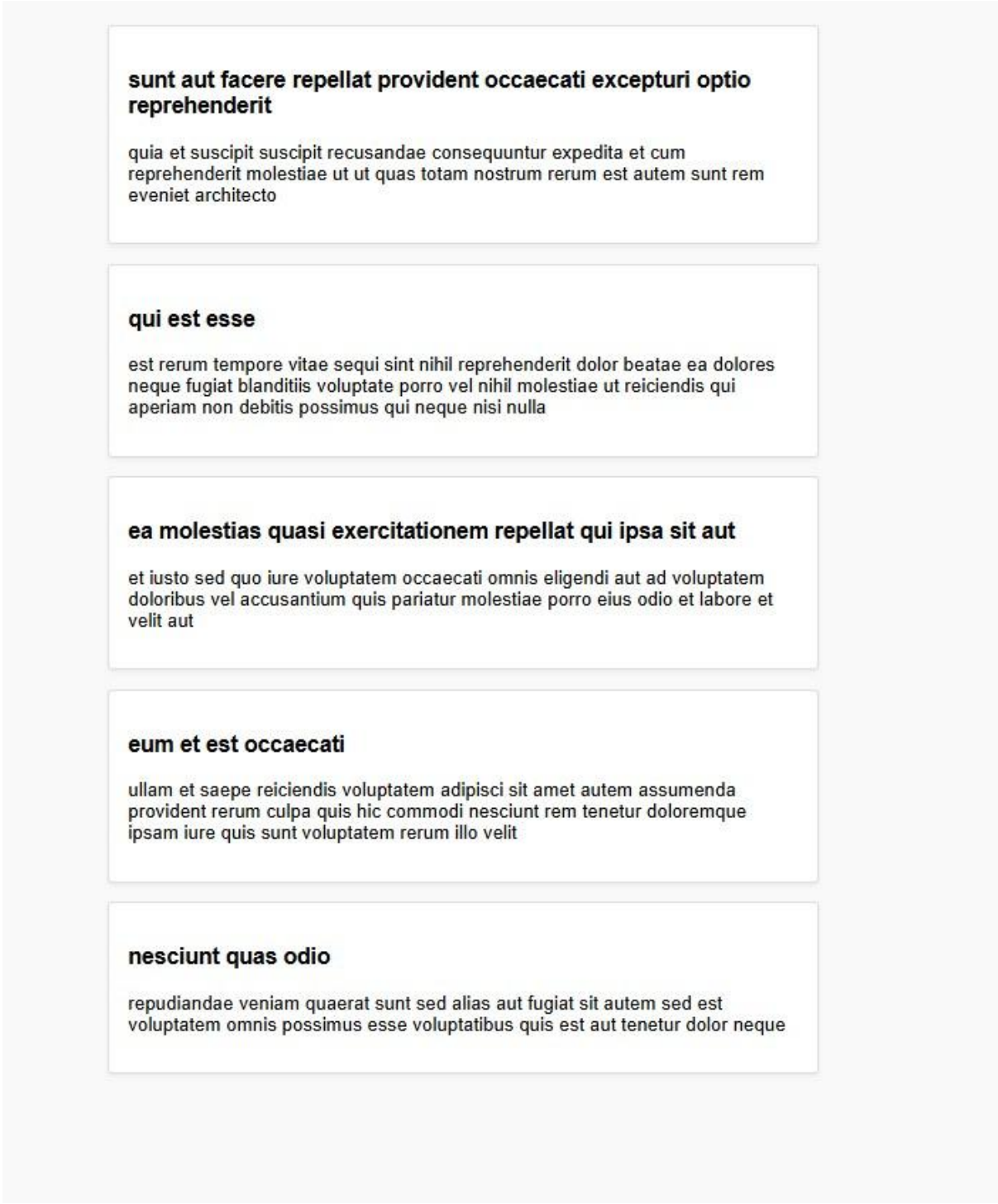
Code Generated:

14.1.task4.html > ...

```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 List</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 2rem;
      background-color: #f9f9f9;
    }

    .container {
      max-width: 600px;
      margin: auto;
    }

    .item {
      background-color: #fff;
      padding: 1rem;
      border: 1px solid #ddd;
      border-radius: 4px;
      margin-bottom: 1rem;
      box-shadow: 0 2px 4px rgba(0,0,0,0.05);
    }

    .loading, .error {
      text-align: center;
      font-size: 1.2rem;
      color: #555;
      margin-top: 2rem;
    }
  </style>
</head>
<body>

  <div class="container" id="data-container">
```

```
    <div class="loading" id="loading">Loading items...</div>
  </div>

  <script>
    const container = document.getElementById('data-container');
    const loading = document.getElementById('loading');

    // Sample API for demo (can be replaced with any API)
    const API_URL = 'https://jsonplaceholder.typicode.com/posts?_limit=5';

    function createItemElement(item) {
      const div = document.createElement('div');
      div.className = 'item';

      const title = document.createElement('h3');
      title.textContent = item.title;

      const body = document.createElement('p');
      body.textContent = item.body;

      div.appendChild(title);
      div.appendChild(body);

      return div;
    }

    Complexity is 6 It's time to do something...
    async function fetchData() {
      try {
        // Show loading
        loading.style.display = 'block';

        const response = await fetch(API_URL);

        if (!response.ok) {
          throw new Error(`HTTP error! Status: ${response.status}`);
        }

        const data = await response.json();

        // Remove loading
        loading.style.display = 'none';

        // Render items
        data.forEach(item => {
          const element = createItemElement(item);
          container.appendChild(element);
        });

      } catch (error) {
        loading.style.display = 'none';
        const errorMessage = document.createElement('div');
        errorMessage.className = 'error';
        errorMessage.textContent = 'Failed to load data. Please try again later.';
        container.appendChild(errorMessage);
        console.error('Fetch error:', error);
      }
    }

    fetchData();
  </script>

</body>
</html>
```

Output:



Observation:

1. Fetch JSON Data

- Uses fetch() to get sample JSON data from jsonplaceholder.typicode.com (a placeholder public API).
- You can replace the URL with any other API that returns JSON.

2. Loading UI

- Shows "Loading items..." text while the data is being fetched.
- Hidden after data is successfully loaded or an error occurs.

3. Safe DOM Creation

- All DOM nodes (title, paragraph) are created using document.createElement, ensuring security (avoids innerHTML).
- Elements are appended cleanly into a container.

4. Error Handling

- Uses try...catch block to catch fetch or parsing errors.
- Displays a user-friendly error message in the DOM if the fetch fails.

5. UI Styling

- Clean, card-style layout for items.
- Text-centered loading and error messages with minimal styling.