SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name	Venkataramana Veeramsetty		
Instructor(s) Name	Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)		
Course Code	24CS002PC215	**Course Title**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week7 - Monday	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	
AssignmentNumber:14.1(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	**Lab 14: Web Design Application – AI-Assisted HTML/CSS/JS Generation** **Lab Objectives** • Design **functional, visually appealing** web applications using HTML, CSS, and JavaScript with AI assistance. • Apply **responsive, accessible, and interactive design principles**.	Week 7 - Monday

- Create **practical UI components** for real-world web applications.
- Use AI to optimize **layout, UX, and performance**.

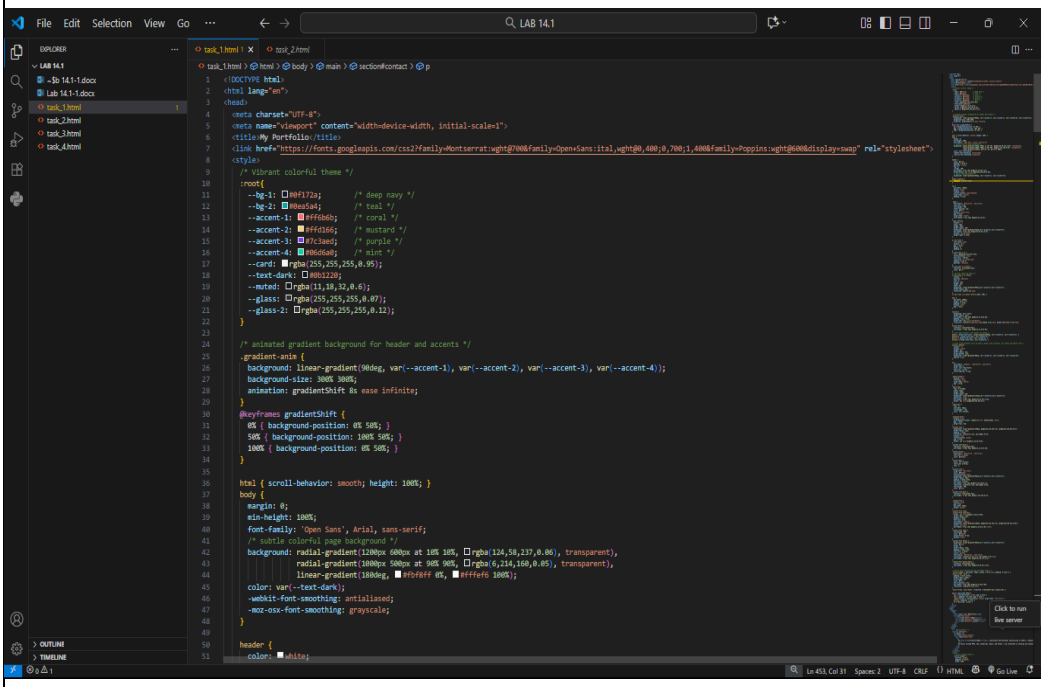
Task 1 – Portfolio Website Design

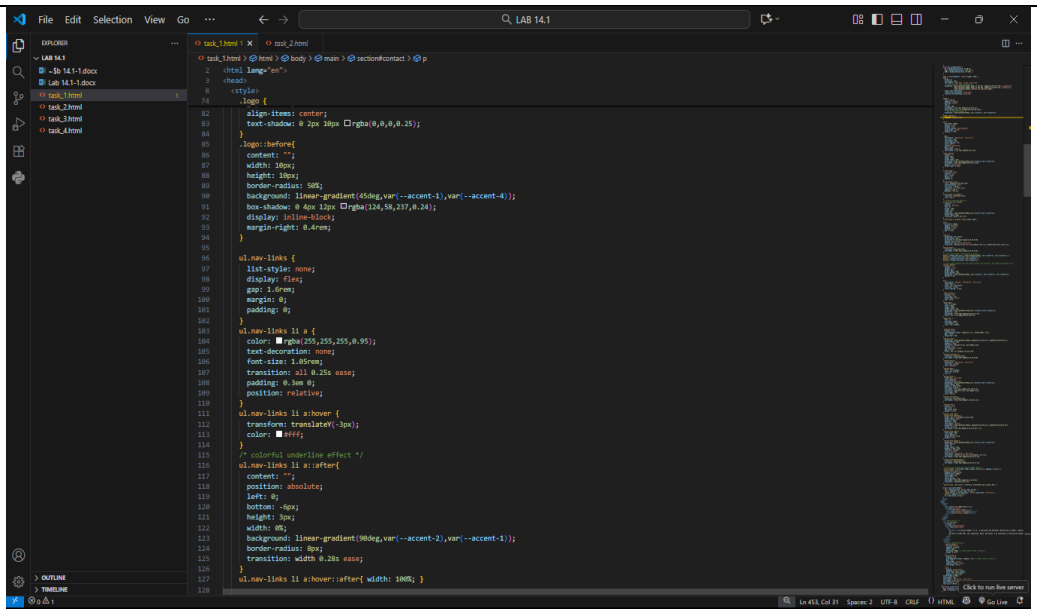
You are building a personal portfolio website to showcase your work.

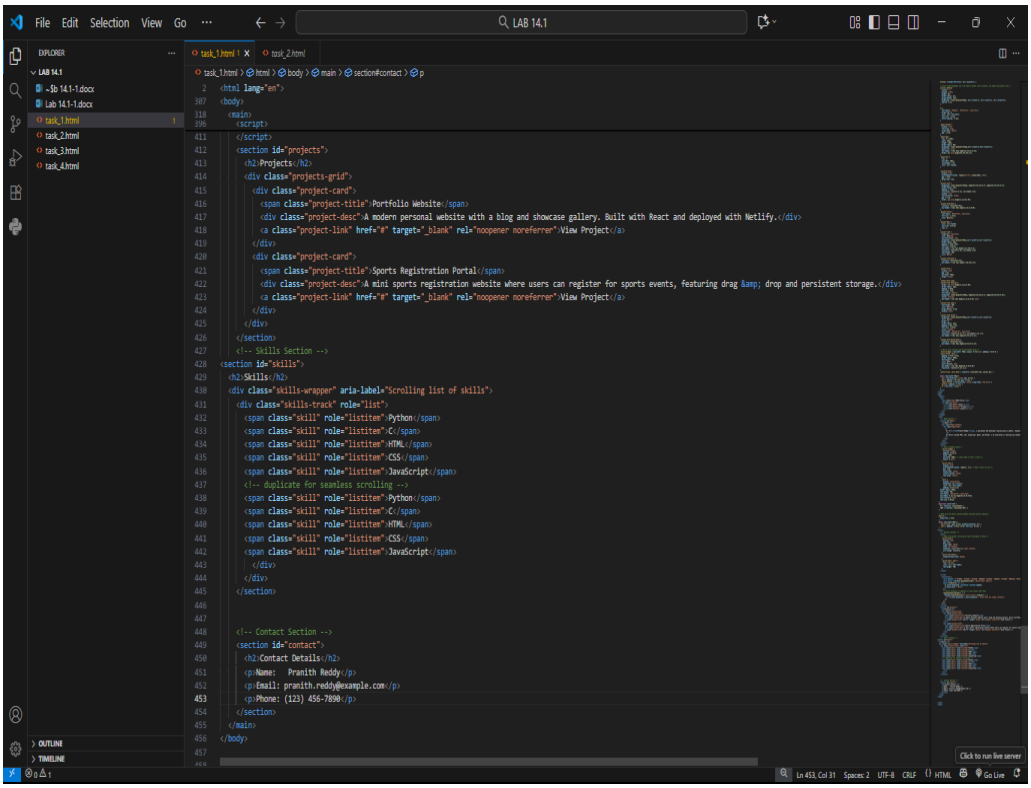
Requirements:

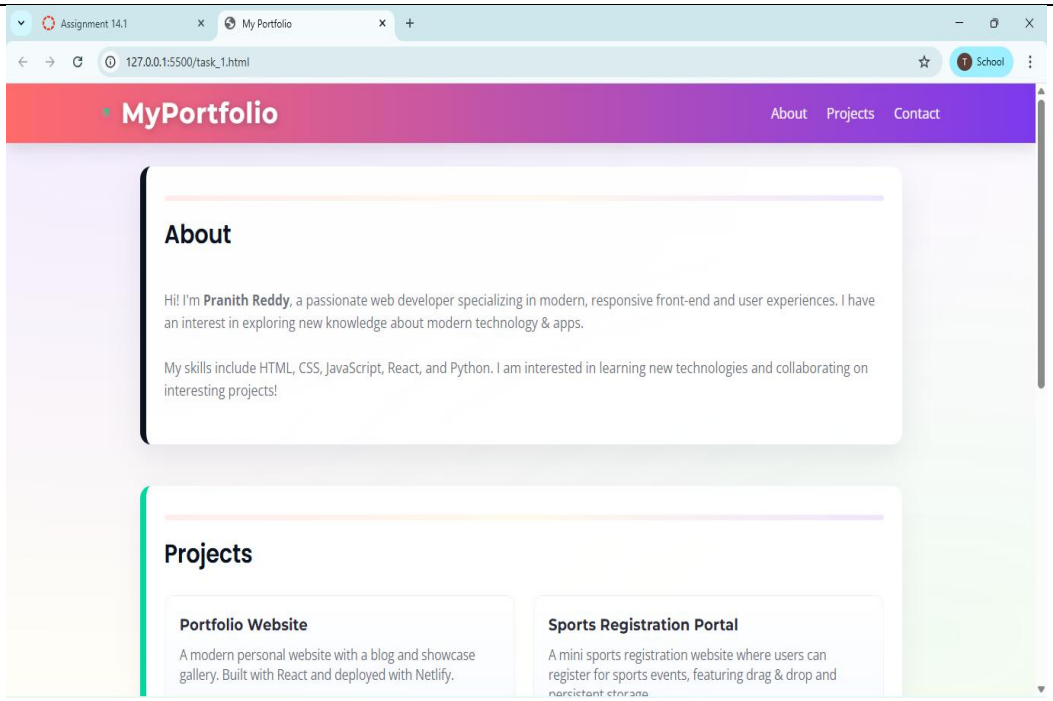
- Create sections for **About Me**, **Projects**, and **Contact**.
- Use AI to:
  - Suggest **color palettes** and typography.
  - Create a **responsive layout** with Grid/Flexbox.
  - Add smooth scrolling navigation.

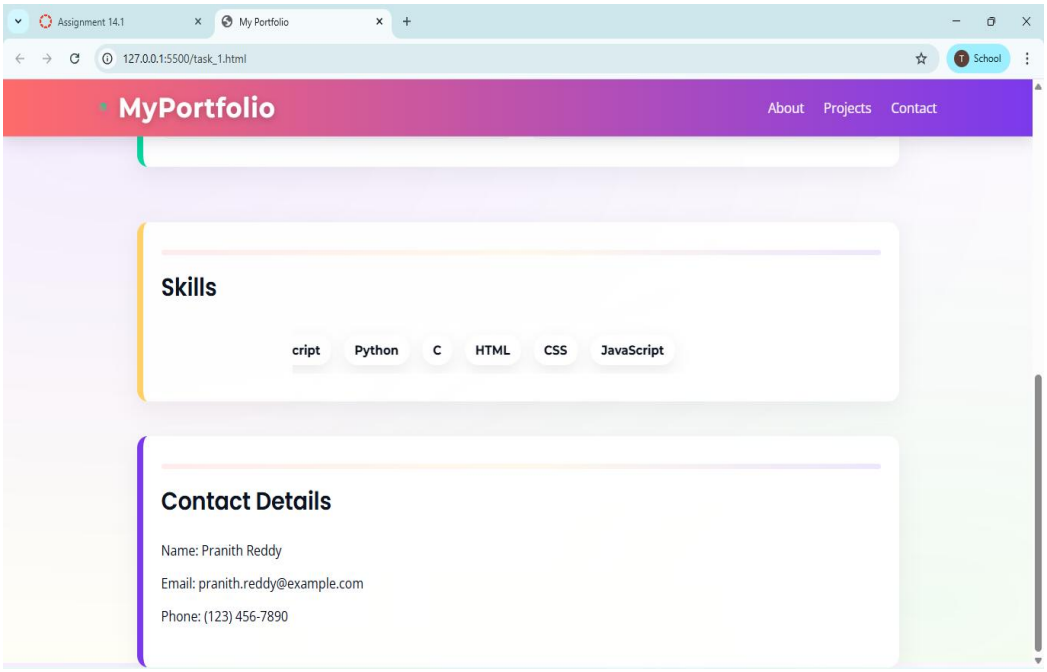
Prompt: Generate a responsive portfolio layout with About, Projects, and Contact sections using Flexbox/Grid. Include smooth scrolling and apply the suggested color palette and typography.




File Edit Selection View Go
LAB 14.1
EXPLORER
LAB 14.1
~$b 14.1-1.docx
Lab 14.1-1.docx
task_1.html
task_2.html
task_3.html
task_4.html
task_1.html
task_2.html
OUTLINE
TIMELINE
Click to run live server


File Edit Selection View Go
LAB 14.1
EXPLORER
LAB 14.1
~$b 14.1-1.docx
Lab 14.1-1.docx
task_1.html
task_2.html
task_3.html
task_4.html
task_1.html
task_2.html
OUTLINE
TIMELINE
Click to run live server





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.

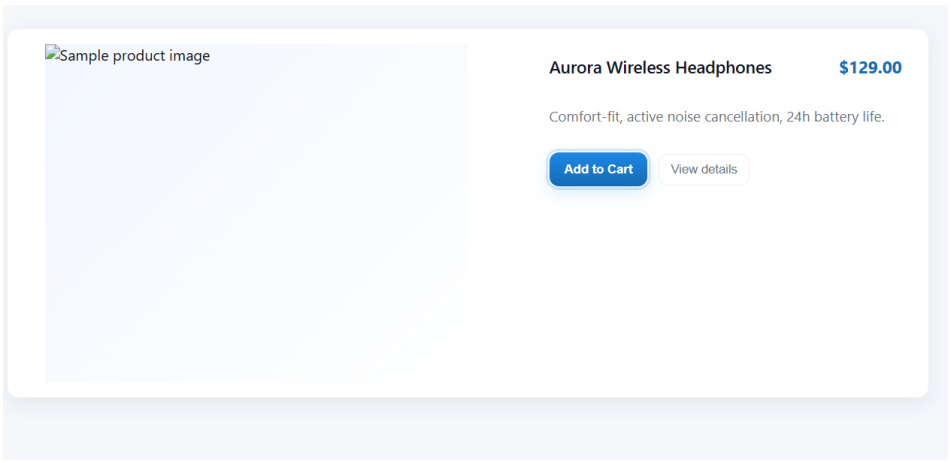
	• Use AI to:
	  ◦ Style with **BEM methodology**.
	  ◦ Make layout **responsive**.
	  ◦ Add **hover effects** and **"Add to Cart" alert**.
	
	Prompt: Create a responsive product card using BEM CSS. Include image, title, price, and an 'Add to Cart' button with hover effects and alert	

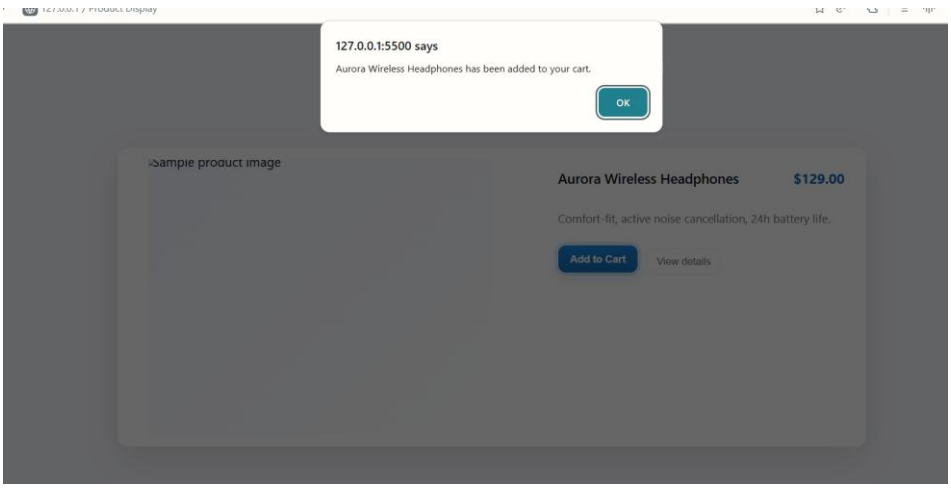
task_2.html > html > head > style > :root

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Product Display – Lab 14.1</title>
  <style>
    /* BEM: block = product */
    :root{
      --bg:#f6f7fb;
      --card:#ffffff;
      --accent:#1e88e5;
      --accent-strong:#166bb3;
      --muted:#6b7280;
      --radius:12px;
      --gap:18px;
    }

    html,body{height:100%;margin:0;font-family:Inter, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Ari

    .product {
      display:flex;
      align-items:center;
      justify-content:center;
      padding:40px 18px;
      box-sizing:border-box;
      min-height:100vh;
    }

    /* Card */
    .product__card {
      background:var(--card);
      border-radius:var(--radius);
      box-shadow:0 6px 20px rgba(16,24,40,0.08);
      max-width:980px;
      width:100%;
      overflow:hidden;
      transition:transform .25s ease, box-shadow .25s ease;
      display:flex;
      flex-direction:column;
      gap:var(--gap);
    }
    .product__card:hover { transform:translateY(-6px); box-shadow:0 14px 40px rgba(16,24,40,0.12); }

    /* Media & info layout */
    .product__media {
      position:relative;
      overflow:hidden;
      background:linear-gradient(135deg,#f3f7ff 0%, #ffffff 100%);
      display:block;
      min-height:220px;
      display:flex;
      align-items:center;
      justify-content:center;
    }
    .product__media img {
      width:100%;
      height:100%;
      object-fit:cover;
      display:block;
      transition:transform .45s cubic-bezier(.2,.9,.2,1);
      transform-origin:center;
    }
    /* hover scale effect */
    .product__card:hover .product__media img { transform:scale(1.05); }

    .product__info {
      padding:22px;
      display:flex;
      flex-direction:column;
      gap:14px;
    }

    .product__row {
      display:flex;
      align-items:center;
      justify-content:space-between;
      gap:12px;
    }
```

```
    .product__title {
      margin:0;
      font-size:1.125rem;
      line-height:1.2;
      font-weight:600;
      color:#0f1724;
    }

    .product__price {
      font-weight:700;
      color:var(--accent-strong);
      font-size:1.125rem;
    }

    .product__meta {
      color:var(--muted);
      font-size:.95rem;
    }

    .product__actions {
      display:flex;
      gap:12px;
      align-items:center;
    }

    .product__button {
      appearance:none;
      border:0;
      background:linear-gradient(180deg,var(--accent),var(--accent-strong));
      color:white;
      padding:10px 16px;
      font-weight:600;
      border-radius:10px;
      cursor:pointer;
      transition:transform .15s ease, box-shadow .15s ease, filter .15s ease;
      box-shadow:0 6px 18px rgba(30,136,229,0.18);
    }
    .product__button:hover { transform:translateY(-3px); filter:brightness(.98); }
    .product__button:active { transform:translateY(0); }
    .product__button:focus { outline:3px solid rgba(30,136,229,0.18); outline-offset:2px; }

    .product__secondary {
      background:transparent;
      color:var(--muted);
      padding:8px 12px;
      border-radius:10px;
      border:1px solid rgba(15,23,36,0.06);
      cursor:pointer;
    }

    /* Responsive: desktop layout with image left, info right */
    @media (min-width:720px){
      .product__card {
        flex-direction:row;
        align-items:stretch;
      }
      .product__media {
        flex:0 0 46%;
        min-height:360px;
      }
      .product__info {
        flex:1;
        padding:28px;
      }
    }

    /* Smaller tweaks */
    @media (max-width:420px){
      .product__title { font-size:1rem; }
      .product__price { font-size:1rem; }
    }
  </style>
</head>
<body>
  <main class="product" role="main">
    <article class="product__card" aria-label="Product card">
      <figure class="product__media" aria-hidden="false">
        <img src="https://via.placeholder.com/900x600.png?text=Product+Image" alt="Sample product image" />
```

```
    </figure>

    <section class="product__info">
      <div class="product__row">
        <h1 class="product__title">Aurora Wireless Headphones</h1>
        <div class="product__price" aria-hidden="false">$129.00</div>
      </div>

      <p class="product__meta">Comfort-fit, active noise cancellation, 24h battery life.</p>

      <div class="product__actions" role="group" aria-label="Purchase actions">
        <button type="button" class="product__button" id="addToCart" data-product-name="Aurora Wireless Headphones" aria-label="Add Aurora Wireless Headphones to cart">
          Add to Cart
        </button>
        <button type="button" class="product__secondary" aria-label="View details">View details</button>
      </div>
    </section>
  </article>
</main>

<script>
  // Simple Add to Cart alert (accessible)
  (function(){
    const btn = document.getElementById('addToCart');
    btn.addEventListener('click', function(e){
      const name = this.getAttribute('data-product-name') || document.querySelector('.product__title').textContent.trim();
      // Use setTimeout to ensure focus behavior doesn't get interrupted by alert on some browsers
      setTimeout(() => { alert(name + ' has been added to your cart.'); }, 10);
    });

    // Keyboard "Enter" and "Space" handled automatically for button element
  })();
</script>
</body>
</html>
```





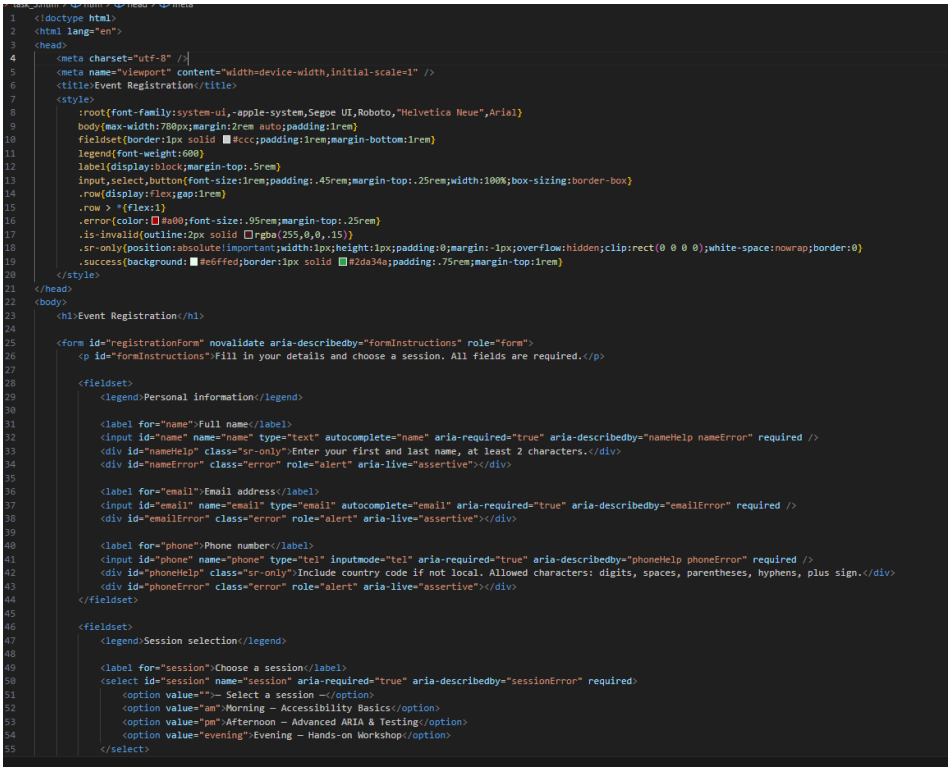
Task 3 – Event Registration Form

Build an event registration form for a conference.

Requirements:

- Collect **name, email, phone number, and session selection**.
- Use AI to:
    - Add **form validation** with JavaScript.
    - Make the form **accessible** with labels and ARIA.
    - Style with a **professional look**.

Prompt: Build an accessible event registration form with ARIA, labels, and JavaScript validation for name, email, phone, and session

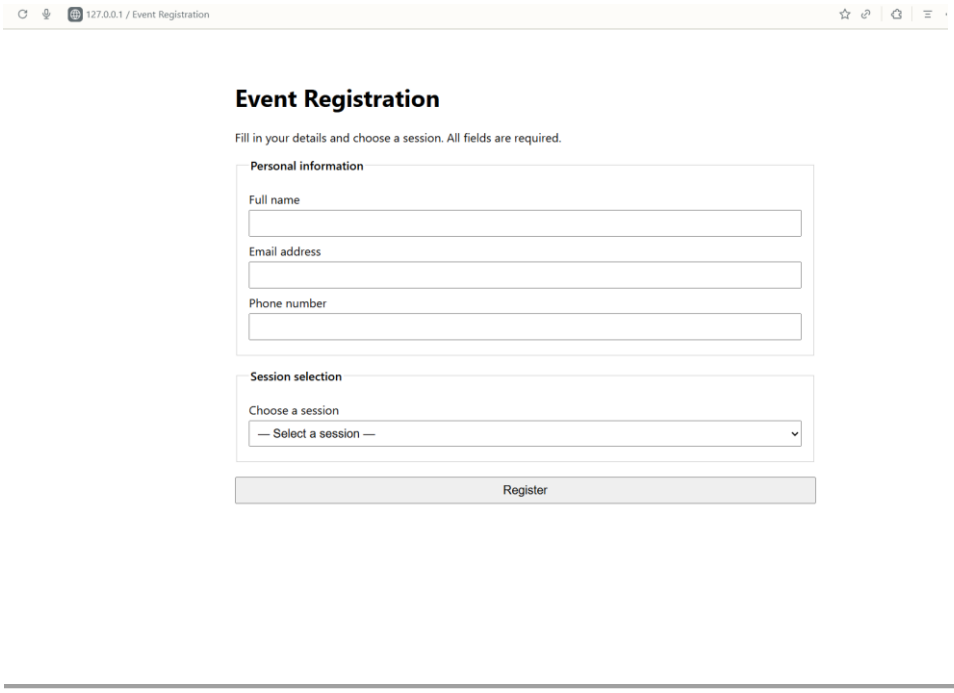


```html
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1" />
    <title>Event Registration</title>
    <style>
        :root{font-family:system-ui,-apple-system,Segoe UI,Roboto,"Helvetica Neue",Arial}
        body{max-width:780px;margin:2rem auto;padding:1rem}
        fieldset{border:1px solid #ccc;padding:1rem;margin-bottom:1rem}
        legend{font-weight:600}
        label{display:block;margin-top:.5rem}
        input,select,button{font-size:1rem;padding:.45rem;margin-top:.25rem;width:100%;box-sizing:border-box}
        .row{display:flex;gap:1rem}
        .row > *{flex:1}
        .error{color:#a00;font-size:.95rem;margin-top:.25rem}
        .is-invalid{outline:2px solid rgba(255,0,0,.15)}
        .sr-only{position:absolute!important;width:1px;height:1px;padding:0;margin:-1px;overflow:hidden;clip:rect(0 0 0 0);white-space:nowrap;border:0}
        .success{background:#e6ffed;border:1px solid #2da34a;padding:.75rem;margin-top:1rem}
    </style>
</head>
<body>
    <h1>Event Registration</h1>

    <form id="registrationForm" novalidate aria-describedby="formInstructions" role="form">
        <p id="formInstructions">Fill in your details and choose a session. All fields are required.</p>

        <fieldset>
            <legend>Personal information</legend>

            <label for="name">Full name</label>
            <input id="name" name="name" type="text" autocomplete="name" aria-required="true" aria-describedby="nameHelp nameError" required />
            <div id="nameHelp" class="sr-only">Enter your first and last name, at least 2 characters.</div>
            <div id="nameError" class="error" role="alert" aria-live="assertive"></div>

            <label for="email">Email address</label>
            <input id="email" name="email" type="email" autocomplete="email" aria-required="true" aria-describedby="emailError" required />
            <div id="emailError" class="error" role="alert" aria-live="assertive"></div>

            <label for="phone">Phone number</label>
            <input id="phone" name="phone" type="tel" inputmode="tel" aria-required="true" aria-describedby="phoneHelp phoneError" required />
            <div id="phoneHelp" class="sr-only">Include country code if not local. Allowed characters: digits, spaces, parentheses, hyphens, plus sign.</div>
            <div id="phoneError" class="error" role="alert" aria-live="assertive"></div>
        </fieldset>

        <fieldset>
            <legend>Session selection</legend>

            <label for="session">Choose a session</label>
            <select id="session" name="session" aria-required="true" aria-describedby="sessionError" required>
                <option value="">— Select a session —</option>
                <option value="am">Morning — Accessibility Basics</option>
                <option value="pm">Afternoon — Advanced ARIA & Testing</option>
                <option value="evening">Evening — Hands-on Workshop</option>
            </select>
```

```
            </select>
            <div id="sessionError" class="error" role="alert" aria-live="assertive"></div>
        </fieldset>

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

        <div id="formMessage" class="success" role="status" aria-live="polite" hidden></div>
    </form>

    <script>
        // GitHub Copilot
        (function () {
            const form = document.getElementById('registrationForm');
            const fields = {
                name: document.getElementById('name'),
                email: document.getElementById('email'),
                phone: document.getElementById('phone'),
                session: document.getElementById('session')
            };
            const errors = {
                name: document.getElementById('nameError'),
                email: document.getElementById('emailError'),
                phone: document.getElementById('phoneError'),
                session: document.getElementById('sessionError')
            };
            const formMessage = document.getElementById('formMessage');

            function clearErrors() {
                Object.values(errors).forEach(e => {
                    e.textContent = '';
                });
                Object.values(fields).forEach(f => {
                    f.removeAttribute('aria-invalid');
                    f.classList.remove('is-invalid');
                });
                formMessage.hidden = true;
                formMessage.textContent = '';
            }

            function focusFirstInvalid(invalidField) {
                if (invalidField && typeof invalidField.focus === 'function') {
                    invalidField.focus();
                    invalidField.scrollIntoView({behavior: 'smooth', block: 'center'});
                }
            }

            function validateName(value) {
                if (!value.trim()) return 'Name is required.';
                if (value.trim().length < 2) return 'Please enter at least 2 characters.';
                // allow letters, hyphens, apostrophes, spaces
```

```
        if (value.trim().length < 2) return 'Please enter at least 2 characters.';
        // allow letters, hyphens, apostrophes, spaces
        if (!/^[\p{L}\p{M}'.\-\s]+$/u.test(value.trim())) return 'Name contains invalid characters.';
        return '';
    }

    function validateEmail(value) {
        if (!value.trim()) return 'Email is required.';
        // basic RFC-like check
        const re = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (!re.test(value.trim())) return 'Please enter a valid email address.';
        return '';
    }

    function validatePhone(value) {
        if (!value.trim()) return 'Phone number is required.';
        // allow "+", digits, spaces, parentheses, hyphens; require 7-20 digits total
        const digits = value.replace(/\D/g, '');
        if (digits.length < 7 || digits.length > 20) return 'Enter a valid phone number with 7-20 digits.';
        if (!/^[+\d\s().-]+$/.test(value)) return 'Phone number contains invalid characters.';
        return '';
    }

    function validateSession(value) {
        if (!value) return 'Please select a session.';
        return '';
    }

    form.addEventListener('submit', function (ev) {
        ev.preventDefault();
        clearErrors();

        const vName = fields.name.value;
        const vEmail = fields.email.value;
        const vPhone = fields.phone.value;
        const vSession = fields.session.value;

        const nameErr = validateName(vName);
        const emailErr = validateEmail(vEmail);
        const phoneErr = validatePhone(vPhone);
        const sessionErr = validateSession(vSession);

        const feedback = { name: nameErr, email: emailErr, phone: phoneErr, session: sessionErr };

        let firstInvalid = null;
        Object.keys(feedback).forEach(key => {
            const message = feedback[key];
            if (message) {
                errors[key].textContent = message;
                fields[key].setAttribute('aria-invalid', 'true');
```

```
                    fields[key].setAttribute('aria-invalid', 'true');
                    fields[key].classList.add('is-invalid');
                    if (!firstInvalid) firstInvalid = fields[key];
                }
            });

            if (firstInvalid) {
                focusFirstInvalid(firstInvalid);
                return;
            }

            // If all valid: simulate submission success
            formMessage.hidden = false;
            formMessage.textContent = 'Registration received. A confirmation email will be sent shortly.';
            // Clear form (optional)
            form.reset();
            // move focus to confirmation for screen reader users
            formMessage.focus?.();
        });

        // Live inline validation (on blur)
        Object.keys(fields).forEach(key => {
            fields[key].addEventListener('blur', () => {
                let msg = '';
                switch (key) {
                    case 'name': msg = validateName(fields.name.value); break;
                    case 'email': msg = validateEmail(fields.email.value); break;
                    case 'phone': msg = validatePhone(fields.phone.value); break;
                    case 'session': msg = validateSession(fields.session.value); break;
                }
                errors[key].textContent = msg;
                if (msg) {
                    fields[key].setAttribute('aria-invalid', 'true');
                    fields[key].classList.add('is-invalid');
                } else {
                    fields[key].removeAttribute('aria-invalid');
                    fields[key].classList.remove('is-invalid');
                }
            });
        });

    })();
  </script>
</body>
</html>
```



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:**
  - Ask AI to write fetch() logic, create DOM nodes safely, and add skeleton/loading text.

Prompt: Write fetch logic to get JSON data, show loading text, handle errors, and render list items to the DOM

```html
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,initial-scale=1" />
    <title>Fetch JSON and Render List</title>
    <style>
        body { font-family: system-ui, sans-serif; margin: 20px; }
        #loading { color: #0a58ca; }
        #error { color: #b02a37; margin-top: 12px; }
        ul { padding-left: 20px; }
        li { margin-bottom: 8px; }
        button { margin-top: 8px; }
    </style>
</head>
<body>
    <h1>Users</h1>

    <div id="loading" aria-live="polite" hidden>Loading…</div>
    <div id="error" role="alert" hidden></div>

    <ul id="list" aria-live="polite"></ul>

    <script>
        // Change to the API you want to fetch
        const API_URL = 'https://jsonplaceholder.typicode.com/users';

        const loadingEl = document.getElementById('loading');
        const errorEl = document.getElementById('error');
        const listEl = document.getElementById('list');

        async function fetchAndRender(url = API_URL) {
            showLoading(true);
            showError(null);
            clearList();

            try {
                const res = await fetch(url, { cache: 'no-store' });
                if (!res.ok) {
                    throw new Error('Server responded with ' + res.status + ' ' + res.statusText);
                }

                const data = await res.json();

                if (!Array.isArray(data)) {
                    throw new Error('Expected an array in response');
                }

                renderList(data);
            } catch (err) {
                showError(err.message || 'An unknown error occurred');
            } finally {
                showLoading(false);
            }
        }
```

```
        function showLoading(visible) {
            loadingEl.hidden = !visible;
        }

        function showError(message) {
            if (!message) {
                errorEl.hidden = true;
                errorEl.textContent = '';
                return;
            }
            errorEl.hidden = false;
            errorEl.textContent = message + ' ';
            // add a retry button when there's an error
            const retry = document.createElement('button');
            retry.textContent = 'Retry';
            retry.addEventListener('click', () => fetchAndRender());
            // clear any previous children and append
            errorEl.appendChild(retry);
        }

        function clearList() {
            listEl.innerHTML = '';
        }

        function renderList(items) {
            if (items.length === 0) {
                listEl.innerHTML = '<li>No items found.</li>';
                return;
            }

            const fragment = document.createDocumentFragment();

            items.forEach(item => {
                // adjust display fields depending on the JSON schema
                // for JSONPlaceholder users we show name and email
                const li = document.createElement('li');
                const title = item.name || item.title || item.id || 'Unnamed item';
                const subtitle = item.email ? ' — ' + item.email : (item.body ? ' — ' + String(item.body).slice(0, 60) + '…' : '')
                li.textContent = title + subtitle;
                fragment.appendChild(li);
            });

            listEl.appendChild(fragment);
        }

        // start on load
        window.addEventListener('DOMContentLoaded', () => {
            fetchAndRender();
        });
    </script>
</body>
```

Users

- 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

	☑ Deliverables (For All Tasks) 1. AI-generated prompts for code and test case generation. 2. At least 3 assert test cases for each task. 3. AI-generated initial code and execution screenshots. 4. Analysis of whether code passes all tests. 5. Improved final version with inline comments and explanation. 6. Compiled report (Word/PDF) with prompts, test cases, assertions, code, and output.	