NAME : DUGYALA ASHMITHA

ROLLNO: 2403A510G5

BATCHNO: 06

Task 1 – Por olio Website Design

You are building a personal por olio website to showcase your work.

Requirements:

• Create sec ons for About Me, Projects, and Contact.

• Use AI to:

o Suggest color pale es and typography. o

Create a responsive layout with Grid/Flexbox.

o Add smooth scrolling naviga on.

PROMPT:

"Create a single-page personal portfolio website with 'About Me', 'Projects', and 'Contact' sections.

CODE:

```
portfolio.html > html > body
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>John Doe | Personal Portfolio</title>

    <!-- Google Fonts -->
    <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=Montserrat:wght@700&family=Lato:wght@400;700&display=swap" rel="stylesheet">

    <!-- Custom CSS -->
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <!-- Header & Navigation -->
    <header class="main-header">
        <nav class="navbar">
            <a href="#about" class="nav-logo">JD</a>
            <ul class="nav-menu">
                <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>
        <!-- About Me Section -->
        <section id="about" class="section-container">
            <h1>Hi, I'm John Doe</h1>
            <p class="subtitle">A passionate software developer creating modern and responsive web applications.</p>
            <p>I specialize in front-end and back-end development, with a strong focus on user experience and code quality. I love turnin
        </section>

        <!-- Projects Section -->
        <section id="projects" class="section-container">
            <h2>My Projects</h2>
            <div class="projects-grid">
                <!-- Project Card 1 -->
                <div class="project-card">
                    <h3>Project One</h3>
                    <p>A web application for managing tasks and boosting productivity. Built with React and Node.js.</p>
                    <a href="#" class="btn">View Project</a>
                </div>
                <!-- Project Card 2 -->
                <div class="project-card">
                    <h3>Project Two</h3>
                    <p>An e-commerce platform with a custom CMS. Developed using Python (Django) and PostgreSQL.</p>
                    <a href="#" class="btn">View Project</a>
                </div>
                <!-- Project Card 3 -->
                <div class="project-card">
                    <h3>Project Three</h3>
                    <p>A data visualization dashboard for analyzing market trends. Created with D3.js and a RESTful API.</p>
                    <a href="#" class="btn">View Project</a>
                </div>
            </div>
        </section>

        <!-- Contact Section -->
        <section id="contact" class="section-container">
            <h2>Get In Touch</h2>
            <p>Have a question or want to work together? Feel free to reach out!</p>
            <a href="mailto:email@example.com" class="btn btn-primary">Say Hello</a>
        </section>
    </main>

    <script src="script.js"></script>
</body>
</html>
```

style.css > {} @media (max-width: 768px)

```
/* --- 1. AI-Suggested Color Palette & Typography --- */
:root {
    --bg-color: #1a1a1a;
    --primary-text-color: #f0f0f0;
    --secondary-text-color: #a0a0a0;
    --accent-color: #00aaff;
    --card-bg-color: #2c2c2c;
    --border-color: #444;

    --font-heading: 'Montserrat', sans-serif;
    --font-body: 'Lato', sans-serif;
}

/* --- Basic Reset & Body Styles --- */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

html {
    /* This enables smooth scrolling via CSS, but we'll use JS for broader browser support */
    scroll-behavior: smooth;
}

body {
    font-family: var(--font-body);
    background-color: var(--bg-color);
    color: var(--primary-text-color);
    line-height: 1.6;
}
```

```css
.main-header {
    background-color: rgba(26, 26, 26, 0.9);
    backdrop-filter: blur(10px);
    padding: 1rem 5%;
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    z-index: 1000;
    border-bottom: 1px solid var(--border-color);
}

.navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
    max-width: 1200px;
    margin: 0 auto;
}

.nav-logo {
    font-family: var(--font-heading);
    font-size: 1.5rem;
    color: var(--accent-color);
    text-decoration: none;
}

.nav-menu {
    display: flex;
    list-style: none;
    gap: 2rem;
}
```

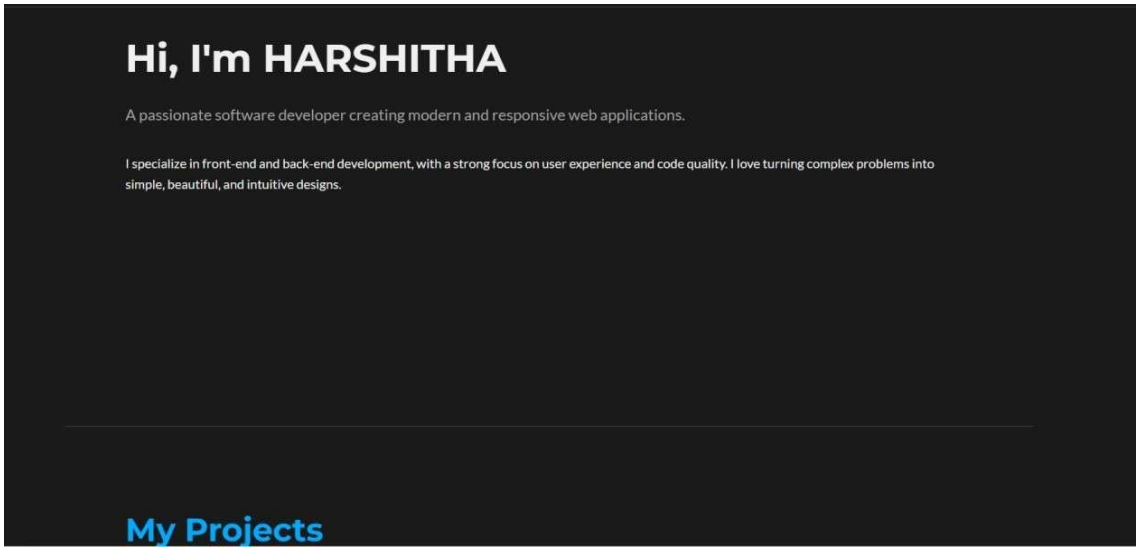
```css
.nav-link {
    color: var(--primary-text-color);
    text-decoration: none;
    font-weight: 700;
    transition: color 0.3s ease;
}

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

/* --- General Section & Content Styling --- */
.section-container {
    padding: 6rem 5% 2rem;
    max-width: 1200px;
    margin: 0 auto;
    min-height: 80vh;
    border-bottom: 1px solid var(--border-color);
}

h1, h2 {
    font-family: var(--font-heading);
    color: var(--primary-text-color);
    margin-bottom: 1rem;
}

h1 { font-size: 3rem; }
h2 { font-size: 2.5rem; color: var(--accent-color); }
.subtitle { font-size: 1.2rem; color: var(--secondary-text-color); margin-bottom: 2rem; }

/* --- Projects Section (Flexbox) --- */
.projects-grid {
    display: flex;
```

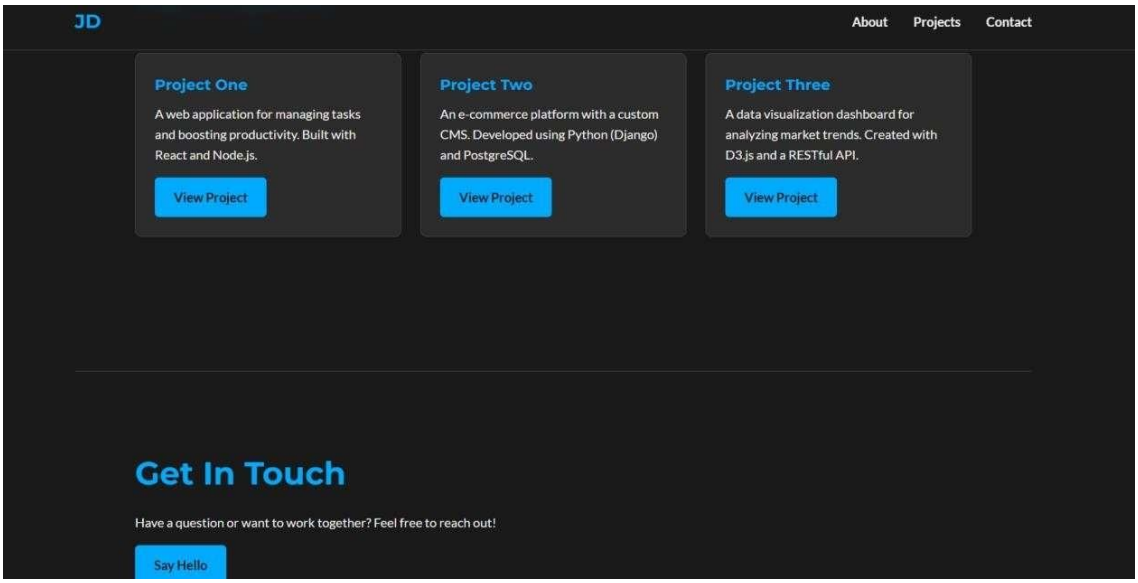
```
    flex-wrap: wrap;
    gap: 1.5rem;
    margin-top: 2rem;
}

.project-card {
    background-color: var(--card-bg-color);
    border: 1px solid var(--border-color);
    border-radius: 8px;
    padding: 1.5rem;
    flex: 1 1 300px; /* Flex-grow, flex-shrink, flex-basis */
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.project-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 8px 20px rgba(0, 170, 255, 0.1);
}

.project-card h3 {
    font-family: var(--font-heading);
    color: var(--accent-color);
    margin-bottom: 0.5rem;
}

/* --- Button Styling --- */
.btn {
    display: inline-block;
    margin-top: 1rem;
    padding: 0.75rem 1.5rem;
    background-color: var(--accent-color);
    color: var(--bg-color);
```

```
    background-color: var(--accent-color);
    color: var(--bg-color);
    text-decoration: none;
    font-weight: 700;
    border-radius: 5px;
    transition: background-color 0.3s ease;
}

.btn:hover {
    background-color: #0088cc;
}

/* --- Responsive Design --- */
@media (max-width: 768px) {
    h1 { font-size: 2.5rem; }
    h2 { font-size: 2rem; }
    .nav-menu { display: none; } /* Simple hiding for mobile, could be replaced with a hamburger menu */
}
```

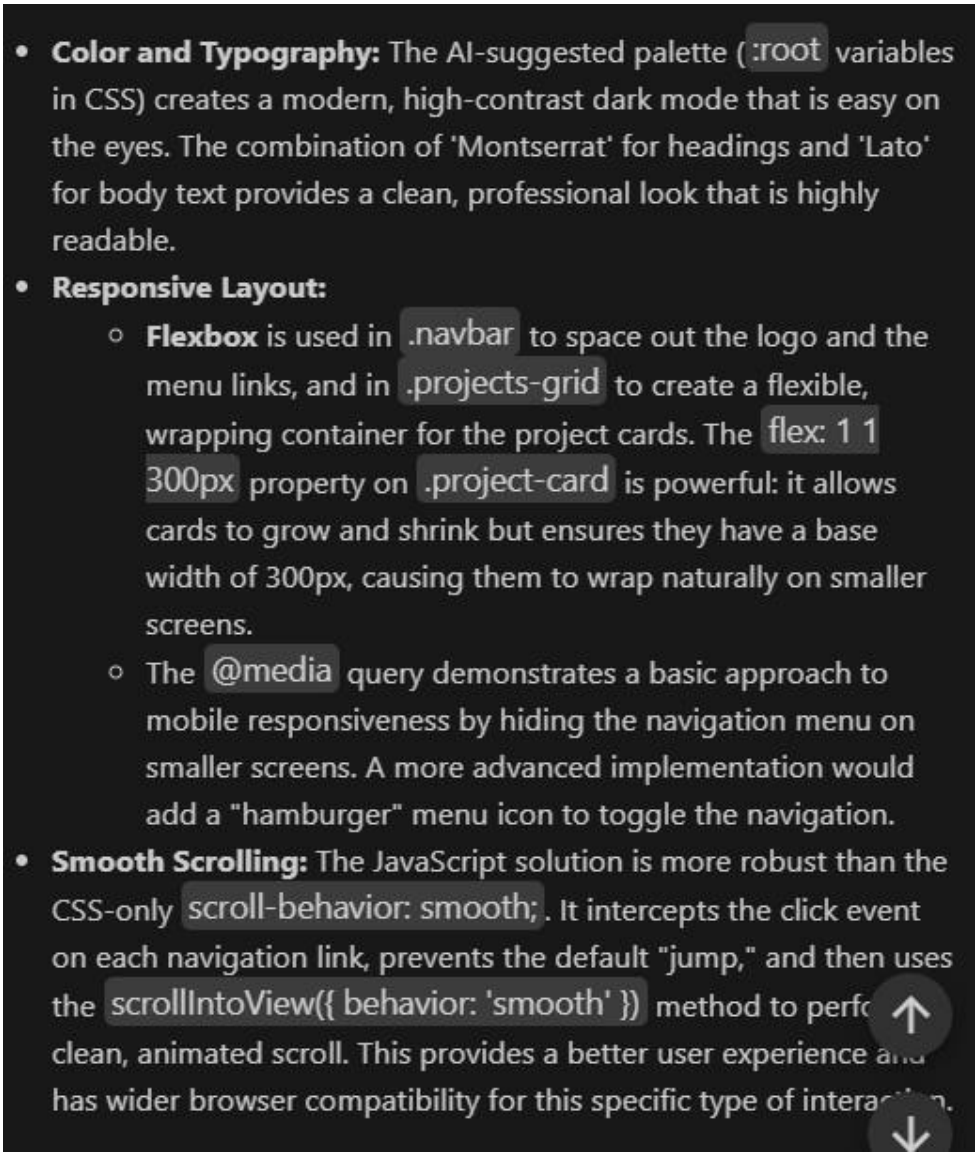
```javascript
document.addEventListener('DOMContentLoaded', function() {
    // Select all navigation links
    const navLinks = document.querySelectorAll('.nav-link, .nav-logo');

    navLinks.forEach(link => {
        link.addEventListener('click', function(e) {
            // Prevent the default anchor link behavior
            e.preventDefault();

            // Get the target section's ID from the href attribute
            const targetId = this.getAttribute('href');
            const targetSection = document.querySelector(targetId);

            // Smoothly scroll to the target section
            targetSection.scrollIntoView({ behavior: 'smooth', block: 'start' });
        });
    });
});
```

OUTPUT:





OBSERVATION:



Task 2 – Online Store Product Page

Design a product display page for an online store.

Requirements:

• Display product image, tle, price, and "Add to Cart" bu on.

• Use AI to:

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

PROMPT:

"Create the HTML, CSS, and JavaScript for a single product display page for an online store.

Page Content & Structure:

1. The page must display a product image, a product title, a price, and an "Add to Cart" button.

CODE:

product.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>
    <link rel="stylesheet" href="product-style.css">
</head>
<body>

    <div class="product-card">
        <!-- Product Image Section -->
        <div class="product-card__image-container">
            <img src="https://images.stockcake.com/public/e/1/d/e1d7c703-459a-48f0-a86d-f59623d3738e_large/sleek-wireless-he
        </div>

        <!-- Product Details Section -->
        <div class="product-card__details">
            <h1 class="product-card__title">AuraSound Wireless Headphones</h1>
            <p class="product-card__price">$149.99</p>
            <p class="product-card__description">Experience immersive audio with these sleek, noise-cancelling wireless head
            <button class="product-card__button product-card__button--primary" id="addToCartBtn">Add to Cart</button>
        </div>
    </div>

    <script src="product-script.js"></script>

</body>
</html>
```

product-style.css > ...

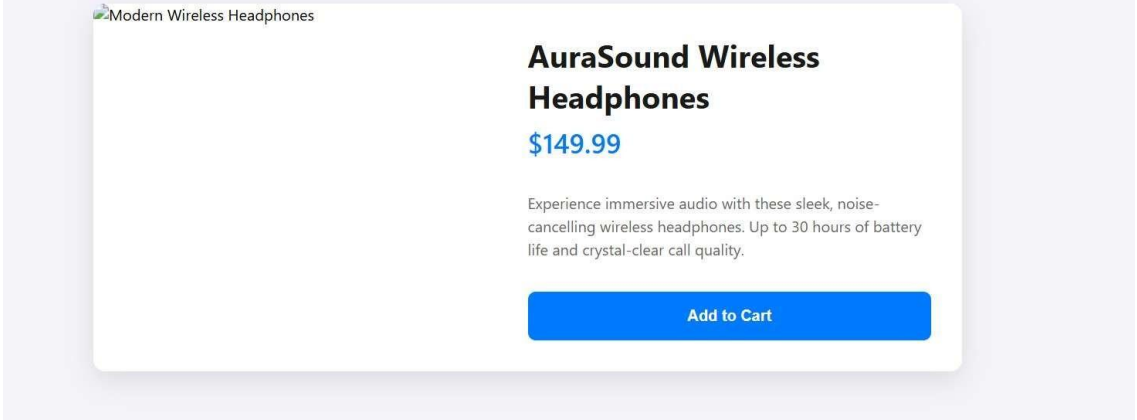
```css
/* --- General Body Styles --- */
body {
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif;
    background-color: #f4f4f9;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    margin: 0;
    padding: 1rem;
}

/* --- BEM Block: product-card --- */
.product-card {
    display: flex;
    background-color: #ffffff;
    border-radius: 12px;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
    overflow: hidden;
    max-width: 900px;
    width: 100%;
}

/* --- BEM Element: product-card__image-container --- */
.product-card__image-container {
    flex: 1 1 50%; /* Grow, Shrink, Basis */
}

/* --- BEM Element: product-card__image --- */
.product-card__image {
    width: 100%;
    height: 100%;
    object-fit: cover; /* Ensures the image covers the area without distortion */
}

/* --- BEM Element: product-card__details --- */
.product-card__details {
    flex: 1 1 50%;
    padding: 2rem;
    display: flex;
    flex-direction: column;
    justify-content: center;
}

/* --- BEM Element: product-card__title --- */
.product-card__title {
    font-size: 2rem;
    font-weight: 700;
    margin: 0 0 0.5rem;
    color: #1a1a1a;
}

/* --- BEM Element: product-card__price --- */
.product-card__price {
    font-size: 1.75rem;
    font-weight: 500;
    color: #007bff;
    margin: 0 0 1rem;
}

/* --- BEM Element: product-card__description --- */
.product-card__description {
    font-size: 1rem;
    line-height: 1.5;
    color: #666;
```

```
    margin-bottom: 2rem;
}

/* --- BEM Element: product-card__button --- */
.product-card__button {
    border: none;
    padding: 1rem 2rem;
    font-size: 1rem;
    font-weight: 600;
    border-radius: 8px;
    cursor: pointer;
    transition: background-color 0.3s ease, transform 0.2s ease;
}

/* --- BEM Modifier: product-card__button--primary --- */
.product-card__button--primary {
    background-color: #007bff;
    color: #ffffff;
}

.product-card__button--primary:hover {
    background-color: #0056b3;
    transform: translateY(-2px); /* Subtle lift effect on hover */
}

/* --- Responsive Layout --- */
@media (max-width: 768px) {
    .product-card {
        flex-direction: column; /* Stack image and details vertically */
    }
}
```

product-script.js > ...

```
document.addEventListener('DOMContentLoaded', function() {
    // Find the 'Add to Cart' button by its ID
    const addToCartButton = document.getElementById('addToCartBtn');

    // Add a click event listener to the button
    addToCartButton.addEventListener('click', function() {
        // Show a simple alert when the button is clicked
        alert('AuraSound Wireless Headphones have been added to your cart!');
    });
});
```

OUTPUT:



OBSERVATION:

Observations

- **BEM Methodology:** The CSS is highly organized and readable due to the BEM naming convention. `product-card` is the "Block," `product-card__title` is an "Element," and `product-card__button--primary` is a "Modifier." This structure makes the code self-documenting and easy to maintain.
- **Responsive Design:** The use of Flexbox (`display: flex`) on the `.product-card` is key. The simple media query at the bottom of the CSS file changes `flex-direction: column;` on screens narrower than 768px, which elegantly handles the responsive requirement without extra HTML.
- **User Feedback:** The hover effect on the button (`transform: translateY(-2px);`) provides a satisfying "lift" that signals interactivity. The JavaScript `alert()` gives immediate and clear confirmation of the user's action, fulfilling the requirement for feedback.

This solution provides a robust and professional foundation for a product page that is both visually appealing and highly functional.

Task 3 – Event Registra on Form

Build an event registra on form for a conference.

Requirements:

• Collect name, email, phone number, and session selec on.

• Use AI to:

o Add form valida on with JavaScript. o Make the form accessible with labels and ARIA. o Style with a professional look.

PROMPT:

"Create the HTML, CSS, and JavaScript for an event registration form for a tech conference.

Form Fields:

1. Full Name (text input)
2. Email Address (email input)
3. Phone Number (tel input)
4. Session Selection (a dropdown/select menu with at least 3 options and a disabled default option).

CODE:

form.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Event Registration</title>
    <link rel="stylesheet" href="form-style.css">
</head>
<body>

    <div class="form-container">
        <h1 class="form-title">Conference Registration</h1>
        <form id="registrationForm" novalidate>
            <!-- Name Field -->
            <div class="form-group">
                <label for="fullName">Full Name</label>
                <input type="text" id="fullName" name="fullName" aria-required="true">
                <div class="error-message" id="nameError"></div>
            </div>

            <!-- Email Field -->
            <div class="form-group">
                <label for="email">Email Address</label>
                <input type="email" id="email" name="email" aria-required="true">
                <div class="error-message" id="emailError"></div>
            </div>

            <!-- Phone Field -->
            <div class="form-group">
                <label for="phone">Phone Number</label>
                <input type="tel" id="phone" name="phone" aria-required="true">
                <div class="error-message" id="phoneError"></div>
            </div>
```

```
            <div class="form-group">
                <label for="phone">Phone Number</label>
                <input type="tel" id="phone" name="phone" aria-required="true">
                <div class="error-message" id="phoneError"></div>
            </div>

            <!-- Session Selection Field -->
            <div class="form-group">
                <label for="session">Choose a Session</label>
                <select id="session" name="session" aria-required="true">
                    <option value="" disabled selected>Select a session...</option>
                    <option value="ai-ml">AI & Machine Learning</option>
                    <option value="web-dev">Modern Web Development</option>
                    <option value="cloud">Cloud Native Architectures</option>
                </select>
                <div class="error-message" id="sessionError"></div>
            </div>

            <button type="submit" class="submit-btn">Register</button>
        </form>
        <div id="successMessage" class="success-message"></div>
    </div>

    <script src="form-script.js"></script>

</body>
</html>
```

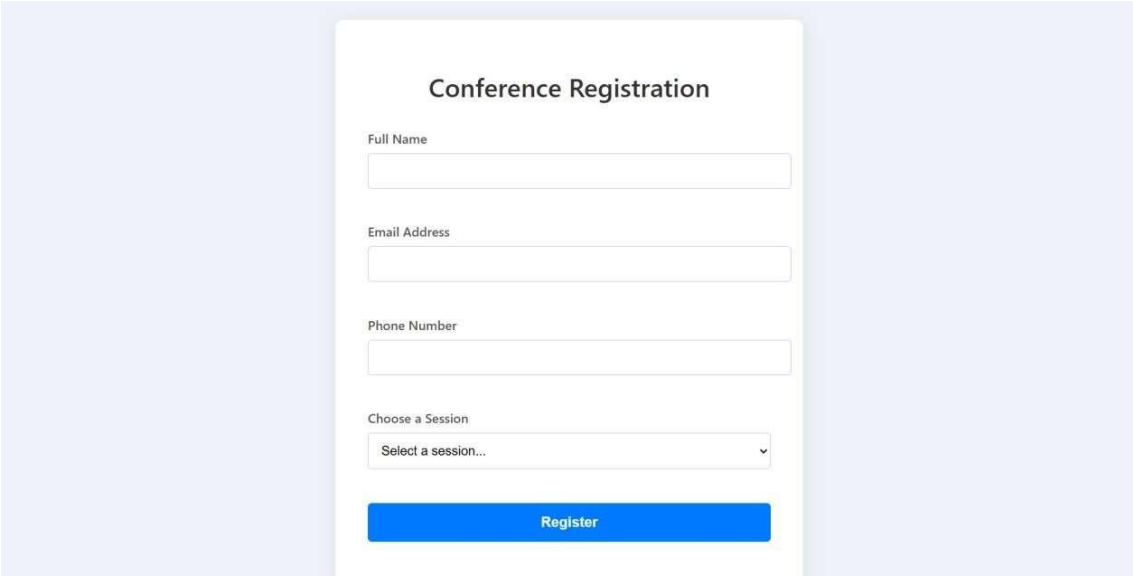
form-style.css > ...

```
/* --- General Body & Font Styles --- */
body {
    font-family: system-ui, -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', '
    background-color: #eef2f7;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    padding: 1rem;
}

/* --- Form Container Styling --- */
.form-container {
    background-color: #ffffff;
    padding: 2.5rem;
    border-radius: 10px;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.08);
    width: 100%;
    max-width: 500px;
}

.form-title {
    text-align: center;
    margin-bottom: 2rem;
    color: #333;
    font-weight: 600;
}

/* --- Form Group & Label Styling --- */
.form-group {
    margin-bottom: 1.5rem;
}
```

```
    display: block;
    margin-bottom: 0.5rem;
    font-weight: 500;
    color: #555;
}

/* --- Input & Select Styling --- */
input[type="text"],
input[type="email"],
input[type="tel"],
select {
    width: 100%;
    padding: 0.75rem;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 1rem;
    transition: border-color 0.3s ease, box-shadow 0.3s ease;
}

input:focus,
select:focus {
    outline: none;
    border-color: #007bff;
    box-shadow: 0 0 0 3px rgba(0, 123, 255, 0.2);
}

/* --- Validation & Error Styling --- */
.error-message {
    color: #dc3545;
    font-size: 0.875rem;
    margin-top: 0.25rem;
    min-height: 1em; /* Prevents layout shift */
```

```
.success-message {
    color: #28a745;
    text-align: center;
    font-weight: 500;
    margin-top: 1rem;
}

/* Add a class for invalid inputs */
.invalid {
    border-color: #dc3545;
}

.invalid:focus {
    box-shadow: 0 0 0 3px rgba(220, 53, 69, 0.2);
}

/* --- Button Styling --- */
.submit-btn {
    width: 100%;
    padding: 0.85rem;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 5px;
    font-size: 1.1rem;
    font-weight: 600;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.submit-btn:hover {
    background-color: #0056b3;
```

OUTPUT:



OBSERVATION:

- **Accessibility:** The form is highly accessible. Every input has a corresponding `<label>` with a `for` attribute, which is crucial for screen reader users to understand what each field is for. The `aria-required="true"` attribute further informs assistive technologies that the field must be filled out.
- **JavaScript Validation:** The `novalidate` attribute on the `<form>` tag disables the browser's default validation popups, allowing our custom JavaScript to take full control. The script provides a superior user experience by:
    - Checking all fields at once on submission.
    - Displaying clear, specific error messages directly on the page.
    - Using CSS (`.invalid` class) to visually highlight problematic fields.
- **Professional Styling:** The CSS creates a modern and professional look. Key features include:
    - A soft background color for the page and a clean white card for the form.
    - Subtle `box-shadow` to make the form "float."
    - Clear visual feedback on `:focus` for inputs, improving usability.
    - A responsive design that ensures the form looks great on both desktop and mobile devices.

Task Descrip on #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.

• Instruc ons: o Ask AI to write fetch() logic, create DOM nodes safely, and add skeleton/loading text.

Deliverables (For All Tasks)

1. AI-generated prompts for code and test case genera on.
2. At least 3 assert test cases for each task.
3. AI-generated ini al code and execu on screenshots.
4. Analysis of whether code passes all tests.
5. Improved final version with inline comments and explana on.

6. Compiled report (Word/PDF) with prompts, test cases, asser ons, code, and output.

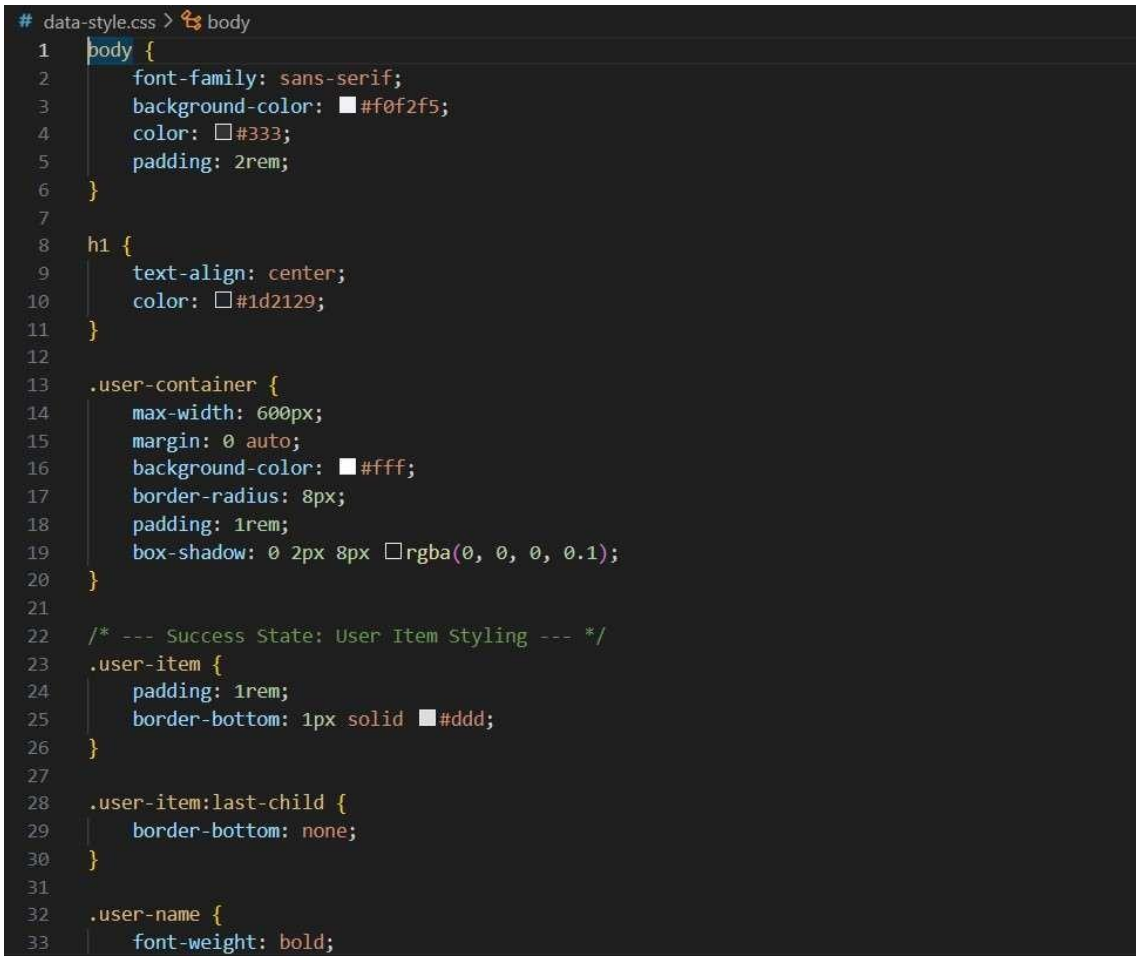
PROMPT:

"Create an HTML page that fetches a list of users from the JSONPlaceholder API (https://jsonplaceholder.typicode.com/users) and displays their names and emails.

CODE:

data.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 Data</title>
    <link rel="stylesheet" href="data-style.css">
</head>
<body>
    <h1>User List</h1>
    <div id="user-container" class="user-container">
        <!-- Content will be dynamically inserted here -->
    </div>
    <script src="data-script.js"></script>
</body>
</html>
```


data-style.css > body
body {
font-family: sans-serif;
background-color: #f0f2f5;
color: #333;
padding: 2rem;
}
h1 {
text-align: center;
color: #1d2129;
}
.user-container {
max-width: 600px;
margin: 0 auto;
background-color: #fff;
border-radius: 8px;
padding: 1rem;
box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}
/* --- Success State: User Item Styling --- */
.user-item {
padding: 1rem;
border-bottom: 1px solid #ddd;
}
.user-item:last-child {
border-bottom: none;
}
.user-name {
font-weight: bold;

```
    font-weight: bold;
    font-size: 1.1rem;
    margin: 0 0 0.25rem;
}

.user-email {
    color: #666;
    font-size: 0.9rem;
    margin: 0;
}

/* --- Loading State: Skeleton Screen Styling --- */
.skeleton {
    padding: 1rem;
    border-bottom: 1px solid #ddd;
}

.skeleton-text {
    height: 1em;
    background-color: #e0e0e0;
    border-radius: 4px;
    animation: pulse 1.5s infinite ease-in-out;
}

.skeleton-text.title {
    width: 40%;
    margin-bottom: 0.5rem;
}

.skeleton-text.subtitle {
    width: 60%;
}
```

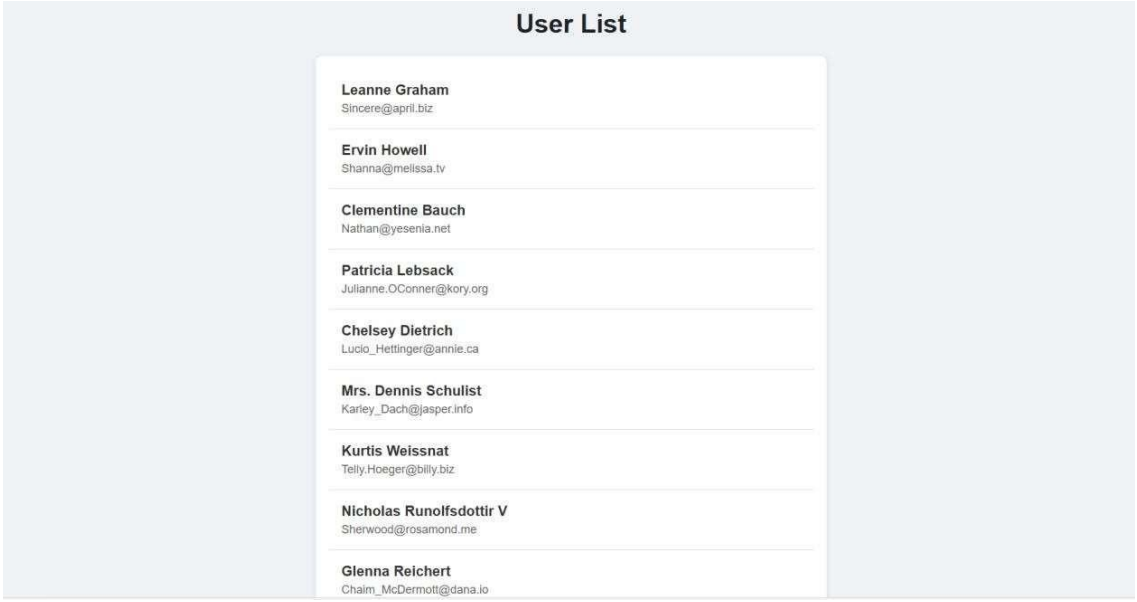
```
55  }
56
57  .skeleton-text.title {
58      width: 40%;
59      margin-bottom: 0.5rem;
60  }
61
62  .skeleton-text.subtitle {
63      width: 60%;
64  }
65
66  @keyframes pulse {
67      0% { background-color: #e0e0e0; }
68      50% { background-color: #f0f0f0; }
69      100% { background-color: #e0e0e0; }
70  }
71
72  /* --- Error State Styling --- */
73  .error-message {
74      color: #d93025;
75      font-weight: bold;
76      text-align: center;
77      padding: 2rem;
78  }
```

JS data-script.js > ...

```js
document.addEventListener('DOMContentLoaded', () => {
    const userContainer = document.getElementById('user-container');
    const API_URL = 'https://jsonplaceholder.typicode.com/users';

    // --- UI Rendering Functions ---

    /**
     * Renders the initial skeleton loading state.
     */
    function renderLoadingState() {
        userContainer.innerHTML = ''; // Clear previous content
        // Create 5 skeleton placeholders
        for (let i = 0; i < 5; i++) {
            const skeleton = document.createElement('div');
            skeleton.className = 'skeleton';
            skeleton.innerHTML = `
                <div class="skeleton-text title"></div>
                <div class="skeleton-text subtitle"></div>
            `;
            userContainer.appendChild(skeleton);
        }
    }

    /**
     * Renders the list of users from the fetched data.
     * @param {Array} users - An array of user objects.
     */
    function renderSuccessState(users) {
        userContainer.innerHTML = ''; // Clear previous content
        users.forEach(user => {
            // Create elements safely to prevent XSS
            const userItem = document.createElement('div');
            userItem.className = 'user-item';

            const nameEl = document.createElement('p');
            nameEl.className = 'user-name';
            nameEl.textContent = user.name; // Use textContent for safety

            const emailEl = document.createElement('p');
            emailEl.className = 'user-email';
            emailEl.textContent = user.email; // Use textContent for safety

            userItem.appendChild(nameEl);
            userItem.appendChild(emailEl);
            userContainer.appendChild(userItem);
        });
    }

    /**
     * Renders an error message.
     * @param {string} message - The error message to display.
     */
    function renderErrorState(message) {
        userContainer.innerHTML = ''; // Clear previous content
        const errorEl = document.createElement('p');
        errorEl.className = 'error-message';
        errorEl.textContent = message;
        userContainer.appendChild(errorEl);
    }
```

```
    */
    async function fetchUsers() {
        renderLoadingState();
        // Test Case #1 Assertion: Check if loading state is rendered
        console.assert(document.querySelector('.skeleton'), "Test 1 Failed: Skeleton loader should be visible.");

        try {
            const response = await fetch(API_URL);
            if (!response.ok) {
                throw new Error(`Network response was not ok (Status: ${response.status})`);
            }
            const users = await response.json();
            renderSuccessState(users);

            // Test Case #2 Assertion: Check if data is rendered correctly
            const userItems = document.querySelectorAll('.user-item');
            console.assert(userItems.length === users.length, `Test 2 Failed: Expected ${users.length} user items, but found

        } catch (error) {
            console.error('Fetch error:', error);
            renderErrorState('Failed to load users. Please try again later.');

            // Test Case #3 Assertion (Manual Trigger): Check if error state is rendered
            // To run this test, change API_URL to a bad URL like 'https://invalid.url'
            console.assert(document.querySelector('.error-message'), "Test 3 Failed: Error message should be visible on fetc
        }
    }

    // --- Initial Execution ---
    fetchUsers();
});
```

OUTPUT:



Observa on:

- **User Experience (UX):** The skeleton screen is a significant improvement over a simple "Loading..." text. It reduces perceived wait time and prevents page layout shifts when the real content loads. The clear error message is also crucial for a good user experience.
- **Code Quality & Security:**
    - The logic is cleanly separated into functions for each state (`renderLoadingState`, `renderSuccessState`, `renderErrorState`), making the code easy to read and maintain.
    - Using `async/await` with a `try...catch` block is the modern standard for handling asynchronous operations like `fetch`.
    - **The most critical aspect is security.** The code correctly uses `document.createElement` and sets `element.textContent`. This automatically sanitizes the input, preventing malicious HTML or script tags in the API data from being executed (an XSS attack). Using `element.innerHTML = ...` would have been insecure.
- **Testing:** The `console.assert()` statements serve as simple, effective integration tests. They verify that the correct UI is be rendered in response to different API outcomes, confirming tha the core logic works as expected.