Name:BANDI SRISHANTH

Batch:06

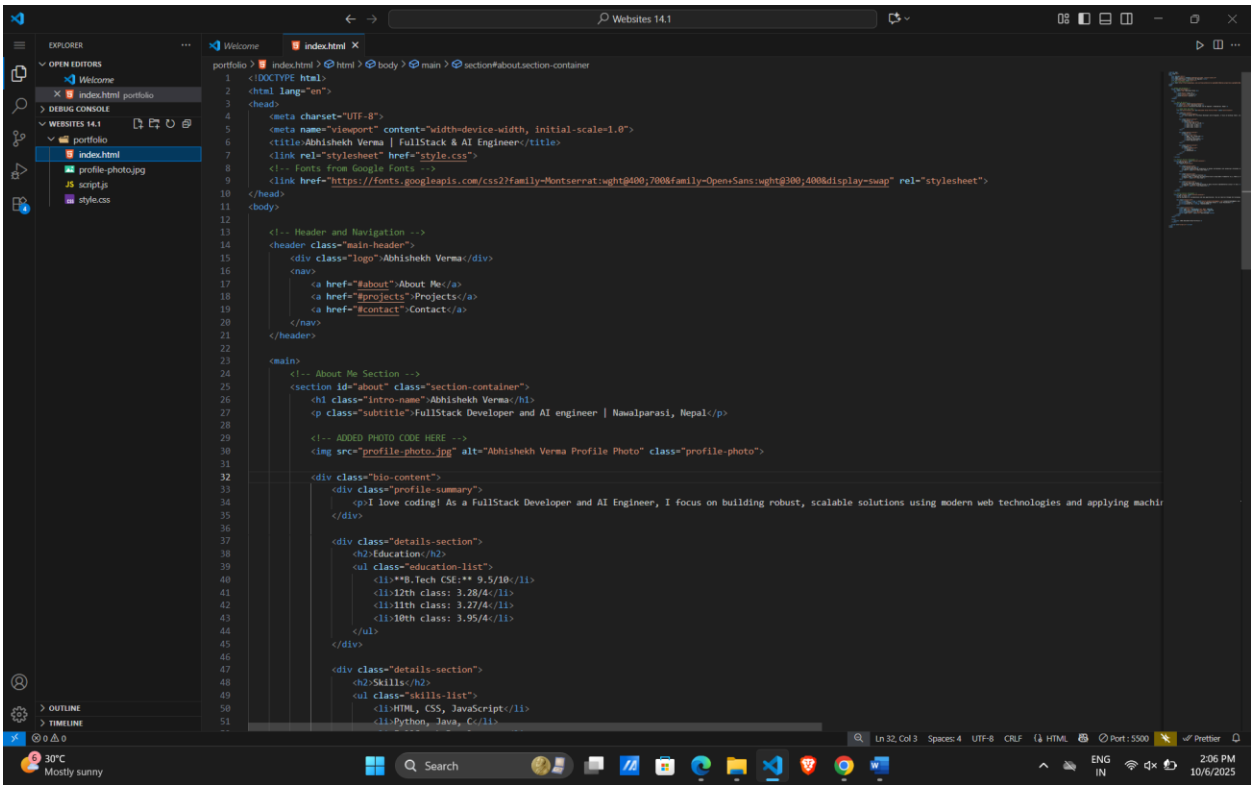
Hall Tno. :2403A510G3

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 Portfoio website using html css js.it should contain section :about me,projects,contact and it should be responsive layout with grid/flexbox and smooth scrolling navigation

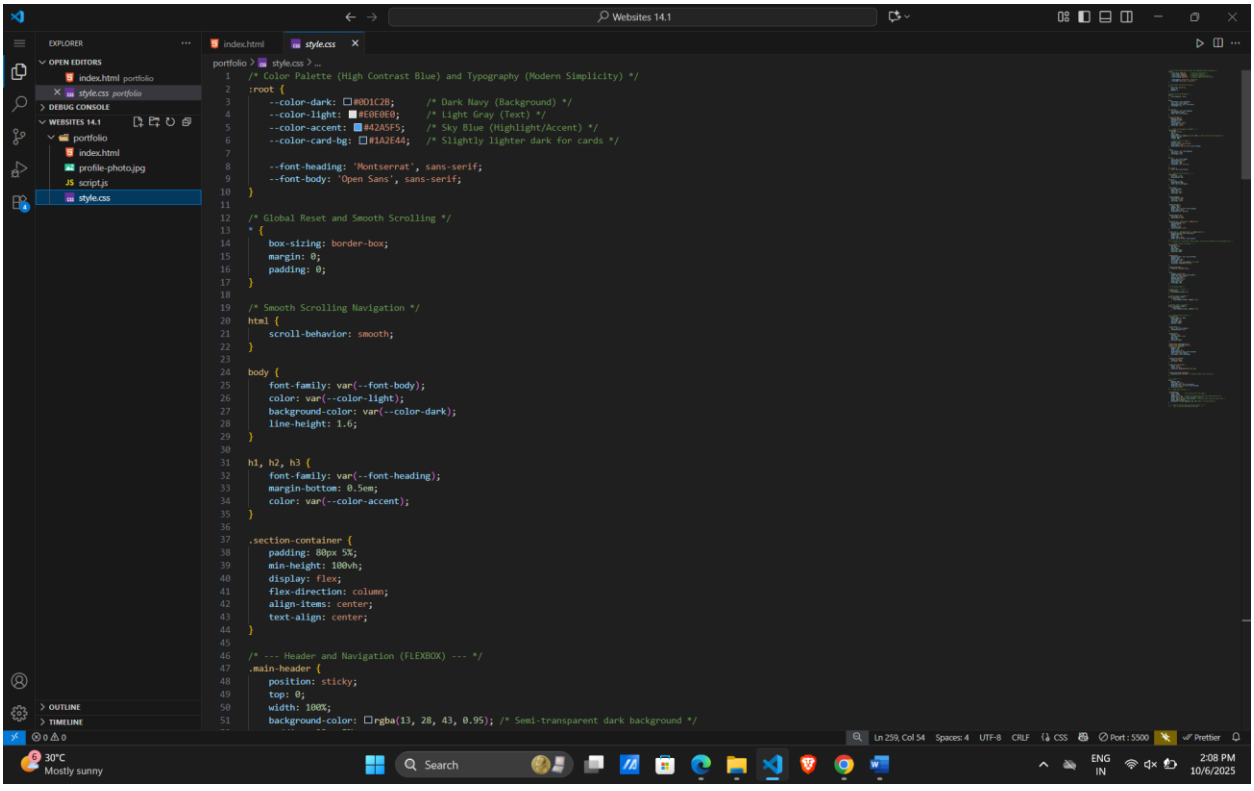
Codes:




```
                    <h2>Skills</h2>
                    <ul class="skills-list">
                        <li>HTML, CSS, JavaScript</li>
                        <li>Python, Java, C</li>
                        <li>FullStack Development</li>
                        <li>AI/ML Engineering</li>
                    </ul>
                </div>

                <div class="details-section">
                    <h2>Hobbies</h2>
                    <ul class="hobbies-list">
                        <li>Coding</li>
                        <li>Innovation</li>
                    </ul>
                </div>
            </div>

        </section>

        <!-- Projects Section (Unchanged) -->
        <section id="projects" class="section-container">
            <h2>Featured Work</h2>
            <div class="project-grid">

                <div class="project-card">
                    <h3>AI-Powered Web App</h3>
                    <p>A project demonstrating integration of Python (AI Backend) with JavaScript (Frontend) for real-time data processing.</p>
                    <a href="#" class="btn">View Case Study</a>
                </div>

                <div class="project-card">
                    <h3>FullStack E-Commerce Platform</h3>
                    <p>Details about a complete web application built using modern frameworks (e.g., Node.js, React) and database management.</p>
                    <a href="#" class="btn">View Code</a>
                </div>

                <div class="project-card">
                    <h3>C/Java Algorithm Design</h3>
                    <p>Showcase of complex problem-solving or data structure implementation using C or Java.</p>
                    <a href="#" class="btn">View Repository</a>
                </div>

            </div>
        </section>
```

```
93
94          <!-- Contact Section (Unchanged) -->
95          <section id="contact" class="section-container">
96              <h2>Get in Touch</h2>
97              <p>I am available for collaborations and new opportunities. You can reach me through the following:</p>
98
99              <div class="contact-info">
100                 <p><strong>Email:</strong> <a href="mailto:kundanverma321@gmail.com">kundanverma321@gmail.com</a></p>
101                 <p><strong>LinkedIn:</strong> <a href="#">Abhishekh Verma</a> (Link Placeholder)</p>
102                 <p><strong>Contact No.:</strong> +918765716460</p>
103             </div>
104
105             <form class="contact-form">
106                 <input type="text" placeholder="Your Name" required>
107                 <input type="email" placeholder="Your Email" required>
108                 <textarea placeholder="Your Message"></textarea>
109                 <button type="submit" class="btn">Send Message</button>
110             </form>
111         </section>
112     </main>
113
114     <footer>
115         <p>&copy; 2025 Abhishekh Verma Portfolio</p>
116     </footer>
117
118     <script src="script.js"></script>
119 </body>
120 </html>
```

CSS



```
    z-index: 1000;

    /* Flexbox setup for Header */
    display: flex;
    justify-content: space-between;
    align-items: center;
    border-bottom: 1px solid var(--color-card-bg);
}

.logo {
    font-family: var(--font-heading);
    font-size: 1.5rem;
    font-weight: bold;
}

nav a {
    color: var(--color-light);
    text-decoration: none;
    margin-left: 25px;
    font-weight: 400;
    transition: color 0.3s ease;
}

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

/* --- About Me Content Styling --- */
.intro-name {
    font-size: 2.5rem;
    margin-bottom: 5px;
}
.subtitle {
    font-size: 1.1rem;
    margin-bottom: 30px;
    color: var(--color-light);
}

.bio-content {
    max-width: 800px;
    width: 100%;
    margin-top: 20px;
    text-align: left;
}

.profile-summary {
    margin-bottom: 40px;
    font-size: 1.1rem;
    text-align: center;
}
```

```css
.details-section {
    margin: 30px 0;
    padding: 20px;
    border: 1px solid var(--color-card-bg);
    border-radius: 5px;
    background-color: #12253a;
}

.details-section h2 {
    text-align: center;
    margin-bottom: 15px;
}

.education-list, .skills-list, .hobbies-list {
    list-style-type: none;
    padding-left: 0;
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
}

.skills-list li, .education-list li, .hobbies-list li {
    background-color: var(--color-dark);
    padding: 8px 15px;
    margin: 5px;
    border-radius: 20px;
    border: 1px solid var(--color-accent);
}
/* If stacked list is preferred, remove display: flex and use traditional list-style-type: disc; */

/* --- Projects Section (CSS GRID) --- */
.project-grid {
    display: grid;
    gap: 30px;
    width: 100%;
    max-width: 1200px;
    margin-top: 40px;
}

.project-card {
    background-color: var(--color-card-bg);
    padding: 25px;
    border-radius: 8px;
    text-align: left;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
    transition: transform 0.3s ease;
}

.project-card:hover {
```

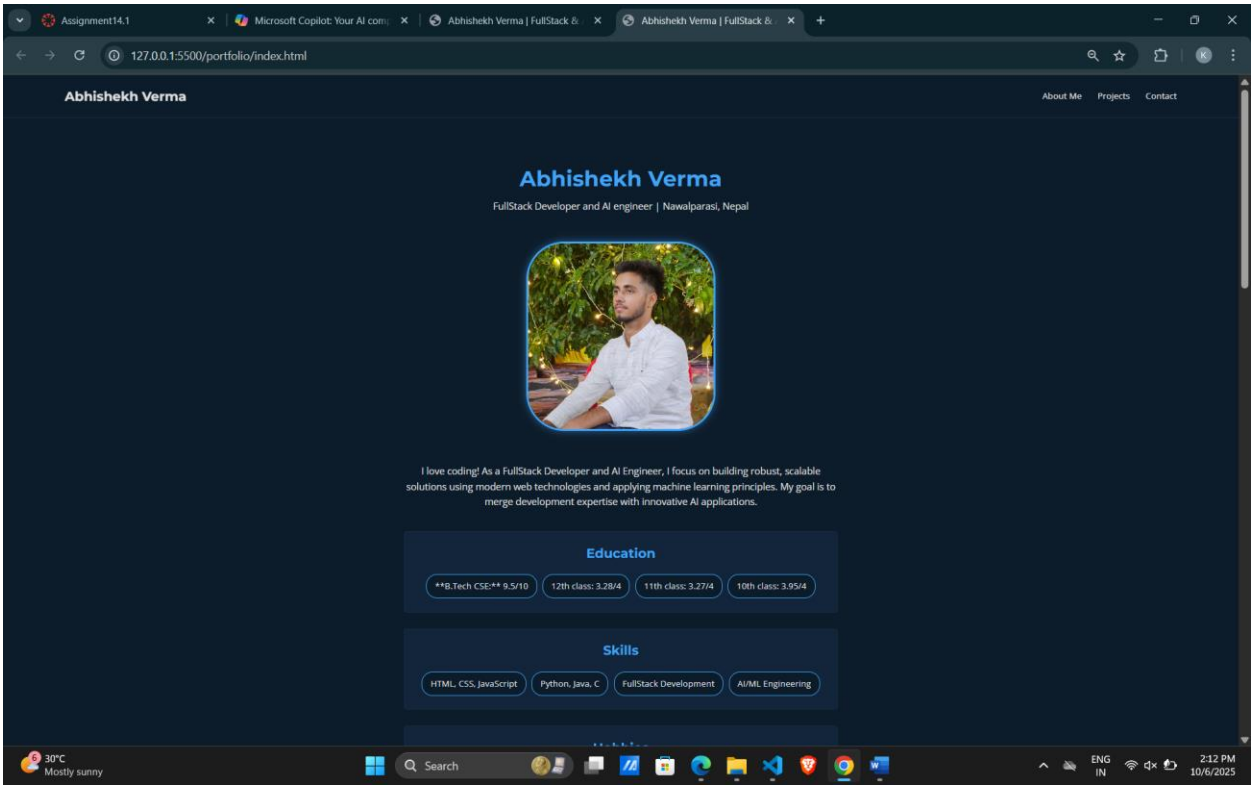
```css
/* Grid Responsiveness */

/* Mobile First (1 column) */
.project-grid {
    grid-template-columns: 1fr;
}

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

/* Desktop Layout (3 columns) */
@media (min-width: 1200px) {
    .project-grid {
        grid-template-columns: repeat(3, 1fr);
    }
}


/* --- Contact Section Styling --- */
.contact-info {
    margin: 20px auto 40px;
    line-height: 2;
    text-align: left;
    max-width: 500px;
    padding: 0 10px;
}

.contact-info a {
    color: var(--color-accent);
    text-decoration: none;
}

.contact-form {
    display: flex;
    flex-direction: column;
    gap: 15px;
    width: 100%;
    max-width: 500px;
}

.contact-form input[type="text"],
.contact-form input[type="email"],
.contact-form textarea {
    padding: 15px;
    border: none;
    border-radius: 5px;
    background-color: var(--color-card-bg);
```

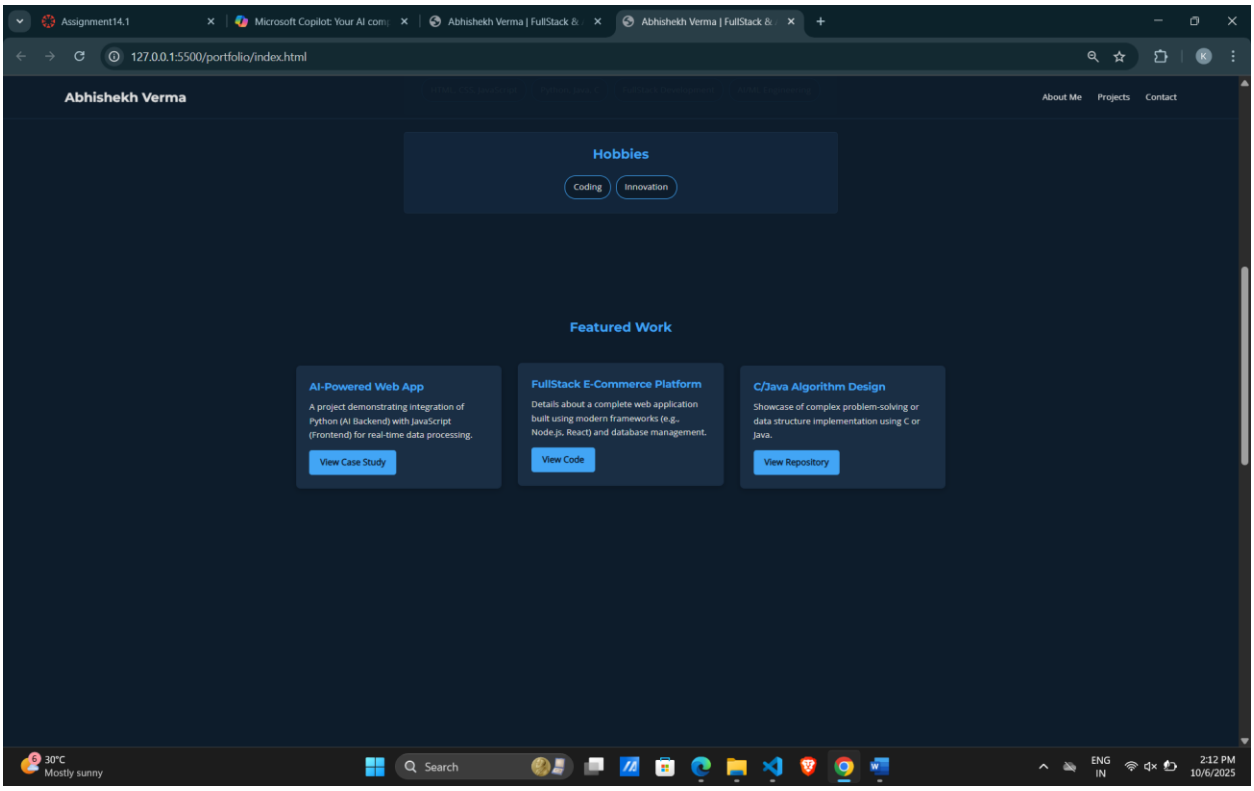
```
    padding: 15px;
    border: none;
    border-radius: 5px;
    background-color: var(--color-card-bg);
    color: var(--color-light);
    font-family: var(--font-body);
}

.contact-form textarea {
    resize: vertical;
    min-height: 150px;
}

.contact-form button.btn {
    border: none;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.contact-form button.btn:hover {
    background-color: #64B5F6; /* Slightly lighter accent on hover */
}


/* --- Footer --- */
footer {
    padding: 20px;
    text-align: center;
    background-color: var(--color-dark);
    border-top: 1px solid var(--color-card-bg);
    font-size: 0.9rem;
}
/* --- Added Photo Styling --- */

.profile-photo {
    width: 150px;       /* Set fixed size for the image */
    height: 150px;
    object-fit: cover;  /* Ensures the image covers the area without distortion */
    border-radius: 50%; /* Makes the image circular */
    border: 4px solid var(--color-accent); /* Optional border using the accent color */
    margin: 20px 0 40px 0;
    box-shadow: 0 0 15px rgba(66, 165, 245, 0.5); /* Soft glow effect */
}

/* --- About Me Content Styling (Existing styles) --- */
/* ... (rest of the CSS code remains the same) ... */
```

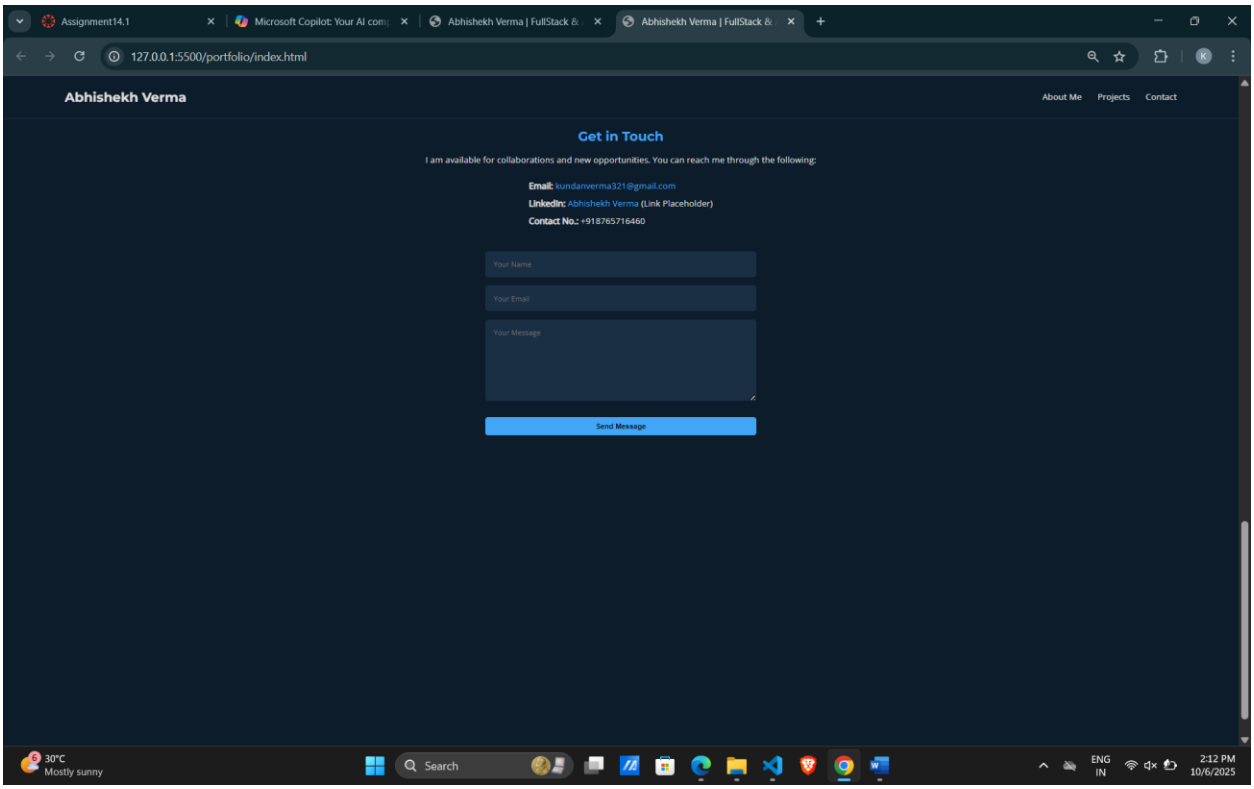
JS

```
/**
 * script.js
 * Portfolio interactivity and setup
 */

document.addEventListener('DOMContentLoaded', () => {
    console.log("Abhishekh Verma Portfolio website loaded successfully.");

    // Note: Smooth scrolling is handled by CSS (scroll-behavior: smooth).

    // Example: Basic form submission listener (prevents default page reload)
    const contactForm = document.querySelector('.contact-form');
    if (contactForm) {
        contactForm.addEventListener('submit', function(event) {
            event.preventDefault();

            // In a real application, this data would be sent to a server/service.
            alert('Message simulated as sent! Thank you for contacting Abhishekh Verma.');

            // Clear the form fields after submission (optional)
            contactForm.reset();
        });
    }
});
```

Output:


Assignment14.1
Microsoft Copilot: Your AI com
Abhishekh Verma | FullStack &
Abhishekh Verma | FullStack &
127.0.0.1:5500/portfolio/index.html
Abhishekh Verma
About Me
Projects
Contact
Abhishekh Verma
FullStack Developer and AI engineer | Nawalparasi, Nepal
I love coding! As a FullStack Developer and AI Engineer, I focus on building robust, scalable solutions using modern web technologies and applying machine learning principles. My goal is to merge development expertise with innovative AI applications.
Education
B.Tech CSE: 9.5/10
12th class: 3.28/4
11th class: 3.27/4
10th class: 3.95/4
Skills
HTML, CSS, JavaScript
Python, Java, C
FullStack Development
AI/ML Engineering
30°C
Mostly sunny
Search
ENG
IN
2:12 PM
10/6/2025


Assignment14.1
Microsoft Copilot: Your AI com
Abhishekh Verma | FullStack &
Abhishekh Verma | FullStack &
127.0.0.1:5500/portfolio/index.html
Abhishekh Verma
About Me
Projects
Contact
Hobbies
Coding
Innovation
Featured Work
AI-Powered Web App
A project demonstrating integration of Python (AI Backend) with JavaScript (Frontend) for real-time data processing.
View Case Study
FullStack E-Commerce Platform
Details about a complete web application built using modern frameworks (e.g., Node.js, React) and database management.
View Code
C/Java Algorithm Design
Showcase of complex problem-solving or data structure implementation using C or Java.
View Repository
30°C
Mostly sunny
Search
ENG
IN
2:12 PM
10/6/2025



Observation:

The code provides a **modern, responsive dark-mode portfolio** for Abhishekh Verma.

It is well-organized using:

1. **HTML** for structure and personal details (skills, education).
2. **CSS** for aesthetics (dark blue theme, clean fonts) and layout, using **Flexbox** for navigation and **CSS Grid** for a responsive project gallery.
3. **Smooth Scrolling** is enabled via CSS for a polished user experience.
4. **JavaScript** is minimal, only handling the contact form submission gracefully.

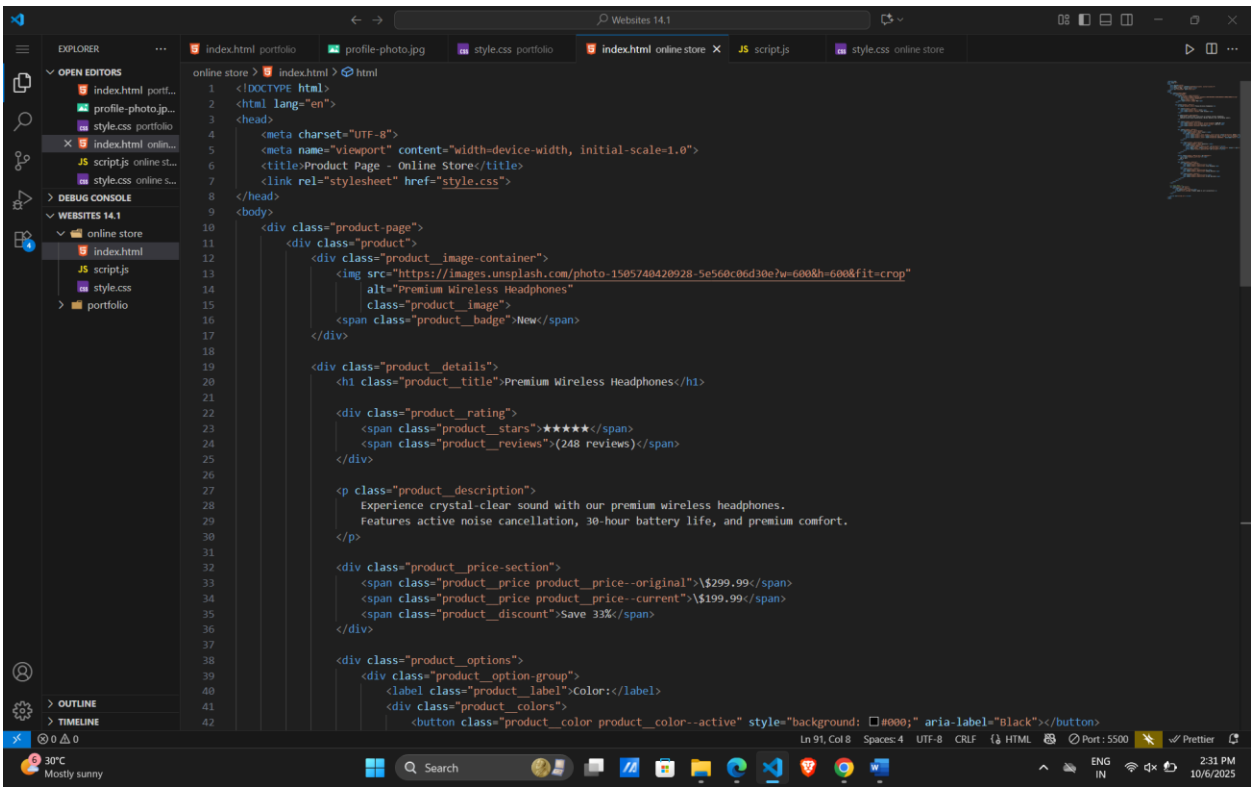
Task 2 – Online Store Product Page

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:

Generate the code for a responsive product page with an image, title, price, and an "Add to Cart" button.
Style everything using the BEM methodology and add interactive hover effects.
Make the "Add to Cart" button trigger a JavaScript alert when clicked.

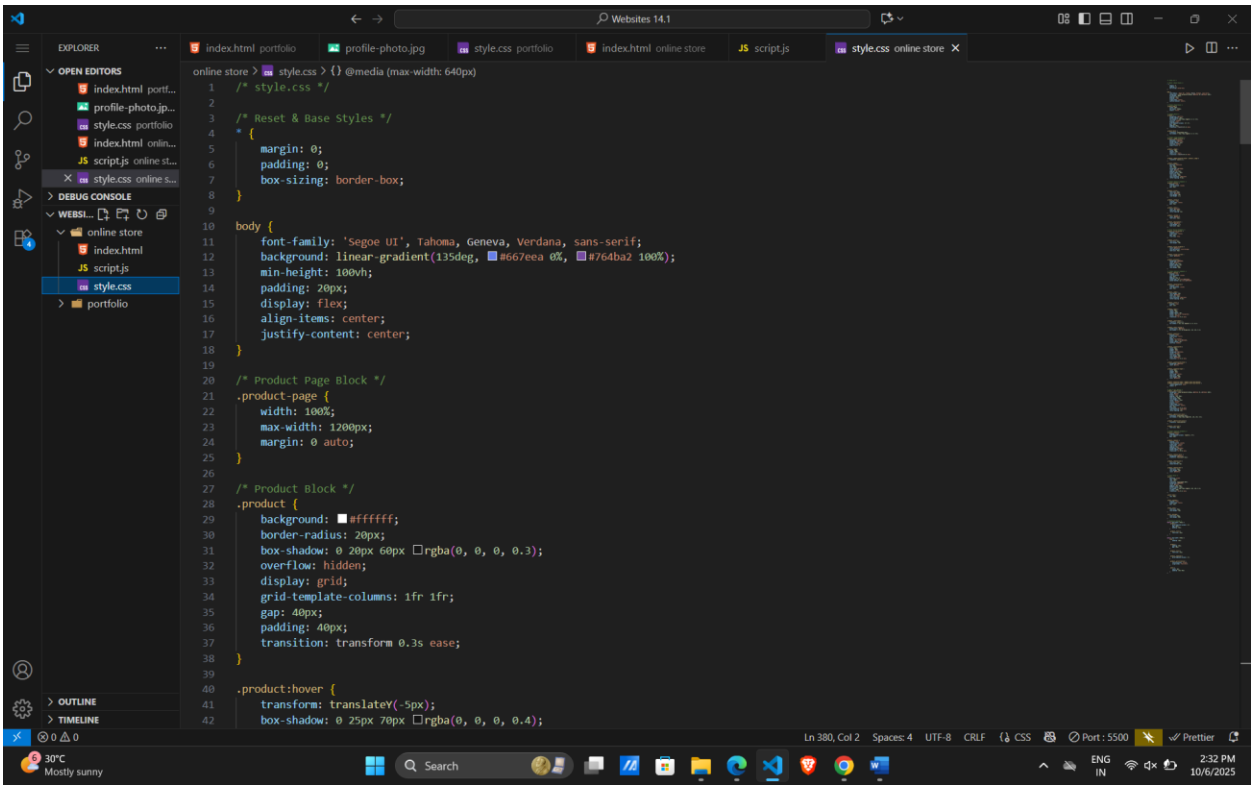
Code:



```
                    <button class="product__color product__color--active" style="background: #000;" aria-label="Black"></button>
                    <button class="product__color" style="background: #fff; border: 1px solid #ddd;" aria-label="White"></button>
                    <button class="product__color" style="background: #4a90e2;" aria-label="Blue"></button>
                </div>
            </div>

            <div class="product__option-group">
                <label class="product__label">Quantity:</label>
                <div class="product__quantity">
                    <button class="product__quantity-btn" id="decrease">-</button>
                    <input type="number" class="product__quantity-input" id="quantity" value="1" min="1" max="10">
                    <button class="product__quantity-btn" id="increase">+</button>
                </div>
            </div>
        </div>

        <button class="product__add-to-cart" id="addToCart">
            <span class="product__cart-icon">🛒</span>
            Add to Cart
        </button>

        <div class="product__features">
            <div class="product__feature">
                <span class="product__feature-icon">🚚</span>
                <span class="product__feature-text">Free Shipping</span>
            </div>
            <div class="product__feature">
                <span class="product__feature-icon">🔁</span>
                <span class="product__feature-text">30-Day Returns</span>
            </div>
            <div class="product__feature">
                <span class="product__feature-icon">✓</span>
                <span class="product__feature-text">2-Year Warranty</span>
            </div>
        </div>
    </div>
</div>
```

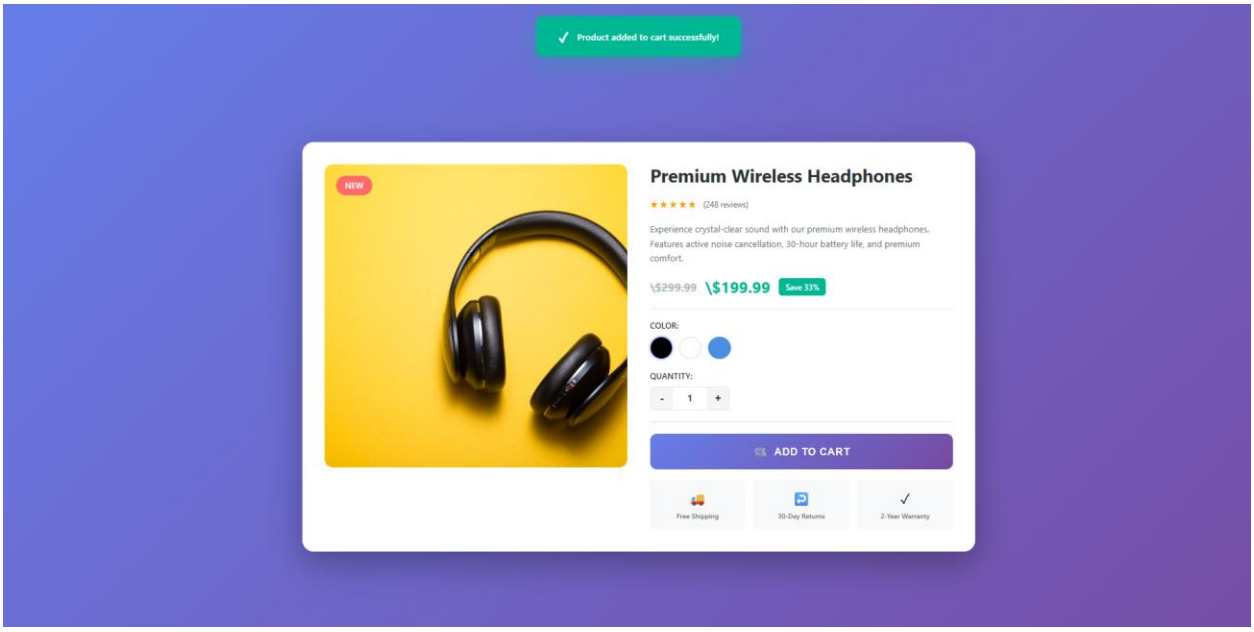
```
            <div class="product__features">
                <div class="product__feature">
                    <span class="product__feature-icon">🚚</span>
                    <span class="product__feature-text">Free Shipping</span>
                </div>
                <div class="product__feature">
                    <span class="product__feature-icon">🔁</span>
                    <span class="product__feature-text">30-Day Returns</span>
                </div>
                <div class="product__feature">
                    <span class="product__feature-icon">✓</span>
                    <span class="product__feature-text">2-Year Warranty</span>
                </div>
            </div>
        </div>
    </div>

    <!-- Alert Modal -->
    <div class="alert" id="alert">
        <div class="alert__content">
            <span class="alert__icon">✓</span>
            <p class="alert__message">Product added to cart successfully!</p>
        </div>
    </div>

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

Css




```
}

.alert__icon {
    font-size: 24px;
    font-weight: 700;
}

.alert__message {
    font-size: 16px;
    font-weight: 600;
}

/* Responsive Design */
@media (max-width: 968px) {
    .product {
        grid-template-columns: 1fr;
        gap: 30px;
        padding: 30px;
    }

    .product__title {
        font-size: 28px;
    }
}

@media (max-width: 640px) {
    body {
        padding: 10px;
    }

    .product {
        padding: 20px;
        gap: 20px;
    }

    .product__title {
        font-size: 24px;
    }

    .product__features {
        grid-template-columns: 1fr;
    }

    .product__price-section {
        flex-direction: column;
        align-items: flex-start;
    }

    .alert {
        width: 90%;
        padding: 15px 20px;
```

Output:



Observation:

 The HTML structure is clean and uses the BEM methodology for class names, which makes the relationship between the markup and styling very clear.

 CSS is used effectively to create a modern, visually appealing card design, complete with subtle hover effects that improve the user experience.

 The design is fully responsive, using a media query to adjust the layout and ensure the product looks good on smaller mobile devices.

 JavaScript is minimal and efficient, using an event listener to provide immediate and direct feedback to the user upon a successful action.

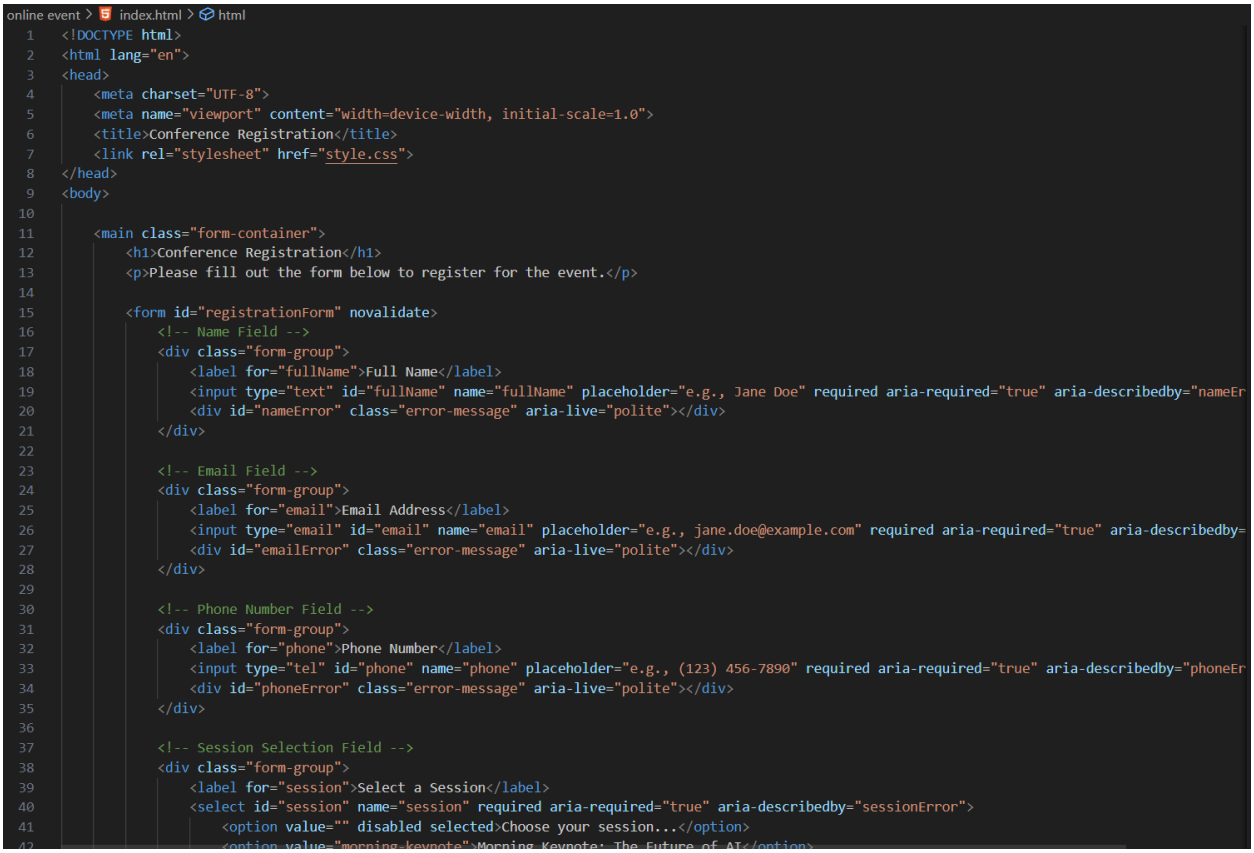
 Overall, the code demonstrates a strong separation of concerns, with HTML for content, CSS for presentation, and JavaScript for behavior.

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:make an event registration form for a conference that collect name, email, phone number, and session selection style it in profrssional way via css add js also

Code:



Css

```css
/* General Body Styles */
body {
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial, sans-serif;
    background-color: #f4f7f6;
    color: #333;
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    margin: 0;
    padding: 20px;
    box-sizing: border-box;
}

/* Form Container */
.form-container {
    background-color: #ffffff;
    padding: 40px;
    border-radius: 8px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    width: 100%;
    max-width: 500px;
}

h1 {
    margin-top: 0;
    margin-bottom: 10px;
    font-size: 24px;
    color: #2c3e50;
}

p {
    margin-bottom: 30px;
    color: #555;
}

/* Form Grouping */
.form-group {
    margin-bottom: 20px;
}

/* Labels */
label {
    display: block;
    font-weight: 600;
    margin-bottom: 8px;
    color: #34495e;
}

/* Input and Select Fields */
input[type="text"],
input[type="email"],
input[type="tel"],
select {
    width: 100%;
    padding: 12px;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box; /* Ensures padding doesn't affect width */
    font-size: 16px;
    transition: border-color 0.3s, box-shadow 0.3s;
}
```

```css
/* Focus State for Inputs */
input:focus,
select:focus {
    border-color: #3498db;
    box-shadow: 0 0 0 3px rgba(52, 152, 219, 0.2);
    outline: none;
}

/* Submit Button */
button {
    width: 100%;
    padding: 15px;
    background-color: #3498db;
    color: #ffffff;
    border: none;
    border-radius: 4px;
    font-size: 18px;
    font-weight: 600;
    cursor: pointer;
    transition: background-color 0.3s;
}

button:hover {
    background-color: #2980b9;
}

/* Error Message Styling */
.error-message {
    color: #e74c3c;
    font-size: 14px;
    margin-top: 5px;
    min-height: 1em; /* Prevents layout shift when messages appear */
}

/* Input Error State */
input.input-error,
select.input-error {
    border-color: #e74c3c;
}

input.input-error:focus,
select.input-error:focus {
    box-shadow: 0 0 0 3px rgba(231, 76, 60, 0.2);
}
```

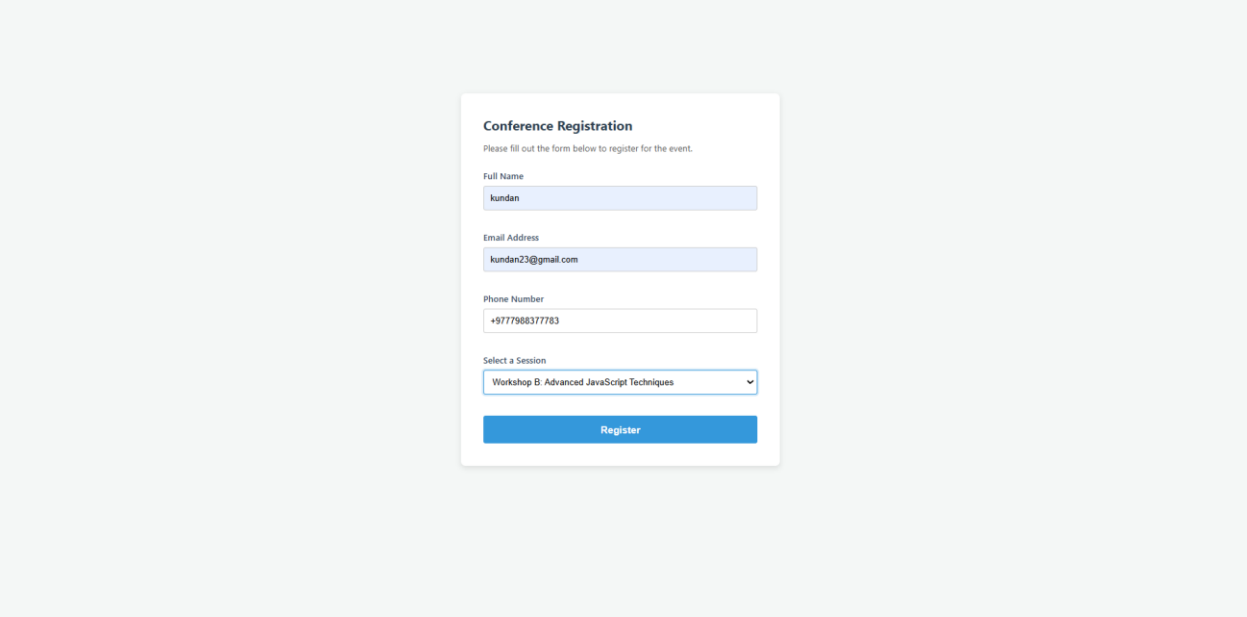
Javascript:

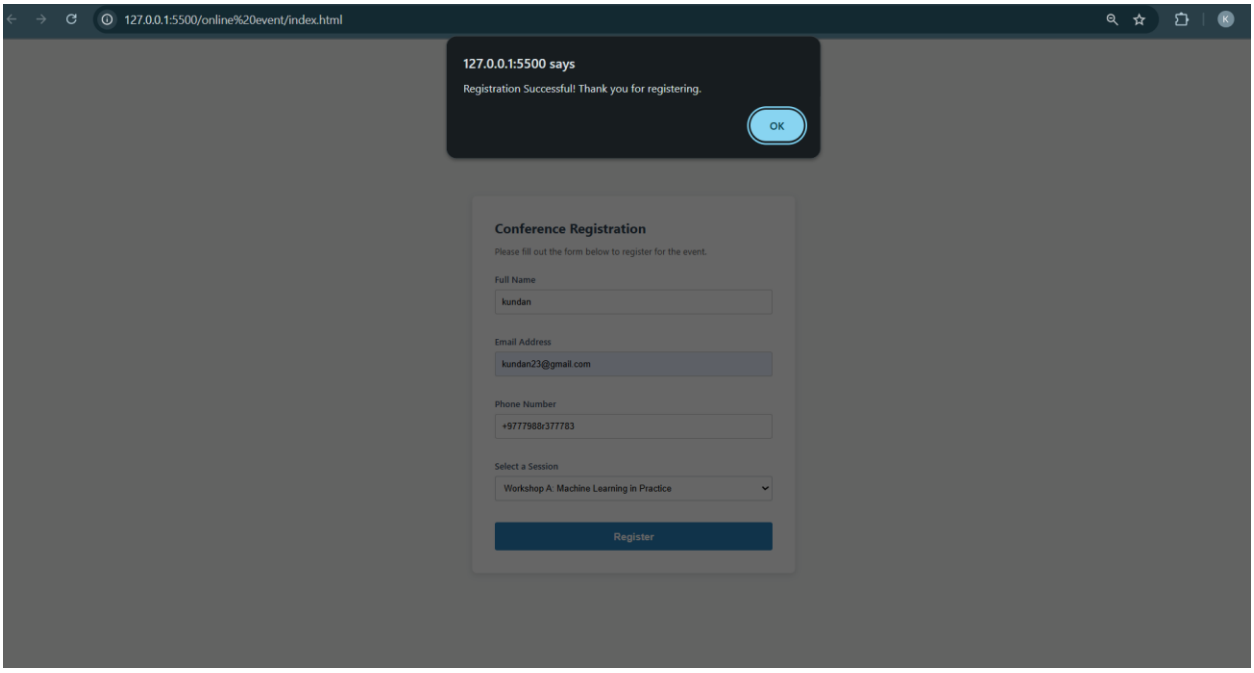
online event > JS script.js > ...

```javascript
document.addEventListener('DOMContentLoaded', () => {
    const form = document.getElementById('registrationForm');

    // Input Fields
    const fullName = document.getElementById('fullName');
    const email = document.getElementById('email');
    const phone = document.getElementById('phone');
    const session = document.getElementById('session');

    // Error Message Divs
    const nameError = document.getElementById('nameError');
    const emailError = document.getElementById('emailError');
    const phoneError = document.getElementById('phoneError');
    const sessionError = document.getElementById('sessionError');

    form.addEventListener('submit', (event) => {
        // Prevent default form submission to handle validation
        event.preventDefault();

        // Perform validation and check if the form is valid
        const isFormValid = validateForm();

        if (isFormValid) {
            // In a real application, you would submit the data to a server here.
            alert('Registration Successful! Thank you for registering.');
            form.reset(); // Clear the form fields

            // Clear any lingering error states
            clearAllErrors();
        }
    });

    function validateForm() {
        // Run validation for each field and store the result
        const isNameValid = validateName();
        const isEmailValid = validateEmail();
        const isPhoneValid = validatePhone();
        const isSessionValid = validateSession();

        // Return true only if all fields are valid
        return isNameValid && isEmailValid && isPhoneValid && isSessionValid;
    }

    function setError(inputElement, errorElement, message) {
        inputElement.classList.add('input-error');
        inputElement.setAttribute('aria-invalid', 'true');
        errorElement.textContent = message;
    }

    function clearError(inputElement, errorElement) {
        inputElement.classList.remove('input-error');
        inputElement.setAttribute('aria-invalid', 'false');
        errorElement.textContent = '';
    }

    function clearAllErrors() {
        clearError(fullName, nameError);
        clearError(email, emailError);
        clearError(phone, phoneError);
        clearError(session, sessionError);
    }
```

```
function validateName() {
    if (fullName.value.trim() === '') {
        setError(fullName, nameError, 'Full Name is required.');
        return false;
    }
    clearError(fullName, nameError);
    return true;
}

function validateEmail() {
    const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (email.value.trim() === '') {
        setError(email, emailError, 'Email Address is required.');
        return false;
    }
    if (!emailPattern.test(email.value)) {
        setError(email, emailError, 'Please enter a valid email address.');
        return false;
    }
    clearError(email, emailError);
    return true;
}

function validatePhone() {
    if (phone.value.trim() === '') {
        setError(phone, phoneError, 'Phone Number is required.');
        return false;
    }
    // Basic phone validation: at least 7 digits
    if (phone.value.replace(/\D/g, '').length < 7) {
        setError(phone, phoneError, 'Please enter a valid phone number.');
        return false;
    }
    clearError(phone, phoneError);
    return true;
}

function validateSession() {
    if (session.value === '') {
        setError(session, sessionError, 'Please select a session.');
        return false;
    }
    clearError(session, sessionError);
    return true;
}
});
```

Output:





Observation:

- **HTML Structure:** Builds a semantic and accessible registration form with fields for name, email, phone, and a session dropdown menu.
- **CSS Styling:** Applies a modern, professional design with a clean layout, responsive adjustments, and clear focus states for better user experience.
- **JavaScript Validation:** Implements client-side validation to ensure all required fields are filled and the email format is correct before allowing submission.
- **User Feedback:** Provides real-time error messages next to invalid fields and uses CSS classes to visually highlight them, guiding the user to correct their input.
- **Submission Handling:** Prevents the default form submission, validates all fields at once, and displays a success alert message after all data is correctly entered.

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: build a website using html css js that Fetch JSON from an API and render items to the DOM
with loading and error UI

Code:

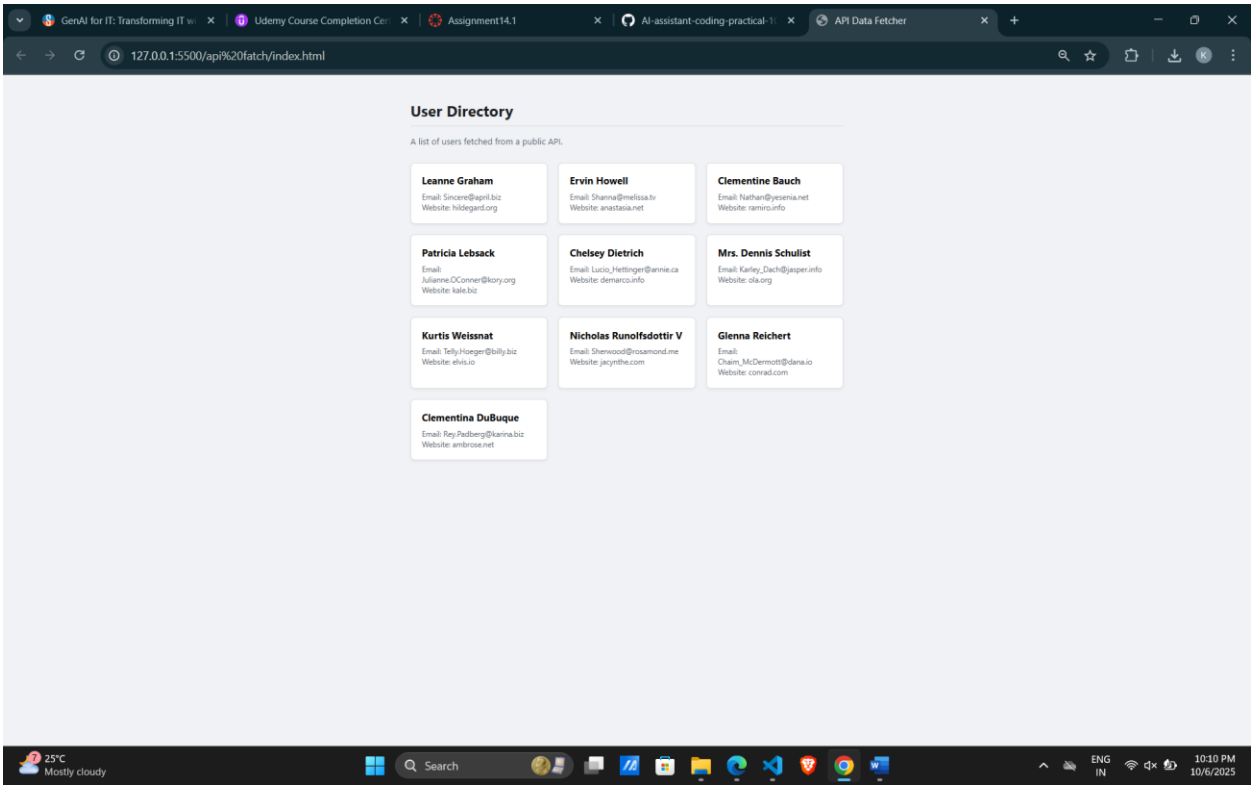
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>API Data Fetcher</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <main class="container">
        <h1>User Directory</h1>
        <p>A list of users fetched from a public API.</p>

        <!-- Loading State: Skeleton UI -->
        <div id="loading-state" class="loading-container">
            <div class="skeleton-card">
                <div class="skeleton-line skeleton-title"></div>
                <div class="skeleton-line skeleton-text"></div>
                <div class="skeleton-line skeleton-text"></div>
            </div>
            <div class="skeleton-card">
                <div class="skeleton-line skeleton-title"></div>
                <div class="skeleton-line skeleton-text"></div>
                <div class="skeleton-line skeleton-text"></div>
            </div>
            <div class="skeleton-card">
                <div class="skeleton-line skeleton-title"></div>
                <div class="skeleton-line skeleton-text"></div>
                <div class="skeleton-line skeleton-text"></div>
            </div>
        </div>

        <!-- Error State -->
        <div id="error-state" class="error-container" style="display: none;" role="alert">
            <!-- Error message will be injected here by JavaScript -->
        </div>

        <!-- Success State: User List -->
        <div id="user-list" class="user-list-container">
            <!-- User cards will be rendered here by JavaScript -->
        </div>
    </main>

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

Javascript:

```javascript
document.addEventListener('DOMContentLoaded', () => {
    // DOM Elements
    const loadingState = document.getElementById('loading-state');
    const errorState = document.getElementById('error-state');
    const userList = document.getElementById('user-list');

    // API Endpoint
    const API_URL = 'https://jsonplaceholder.typicode.com/users';

    // Main function to fetch and render users
    async function fetchAndRenderUsers() {
        // --- 1. Set Initial State (Show Loading) ---
        showLoadingState();

        try {
            // --- 2. Fetch Data ---
            const response = await fetch(API_URL);

            // Handle HTTP errors (e.g., 404, 500)
            if (!response.ok) {
                throw new Error(`HTTP error! Status: ${response.status}`);
            }

            const users = await response.json();

            // --- 3. Render Data (Success) ---
            renderUserList(users);

        } catch (error) {
            // --- 4. Handle Errors ---
            console.error('Failed to fetch users:', error);
            showErrorState(`Failed to load user data. Please try again later.`);
        }
    }

    function showLoadingState() {
        loadingState.style.display = 'grid'; // Use grid to match its layout
        errorState.style.display = 'none';
        userList.style.display = 'none';
    }

    function showErrorState(message) {
        loadingState.style.display = 'none';
        errorState.textContent = message;
        errorState.style.display = 'block';
        userList.style.display = 'none';
    }

    function showSuccessState() {
        loadingState.style.display = 'none';
        errorState.style.display = 'none';
```

```
    }

    function showSuccessState() {
        loadingState.style.display = 'none';
        errorState.style.display = 'none';
        userList.style.display = 'grid'; // Use grid to match its layout
    }

    // Safely creates and appends user cards to the DOM
    function renderUserList(users) {
        // Clear any previous content
        userList.innerHTML = '';

        // Loop through each user and create a card
        users.forEach(user => {
            // Create elements
            const card = document.createElement('div');
            card.className = 'user-card';

            const nameElement = document.createElement('h3');
            const emailElement = document.createElement('p');
            const websiteElement = document.createElement('p');

            // --- SAFELY Set Content using .textContent ---
            // This prevents XSS attacks by not parsing the string as HTML
            nameElement.textContent = user.name;
            emailElement.textContent = `Email: ${user.email}`;
            websiteElement.textContent = `Website: ${user.website}`;

            // Append elements to the card
            card.appendChild(nameElement);
            card.appendChild(emailElement);
            card.appendChild(websiteElement);

            // Append the card to the list container
            userList.appendChild(card);
        });

        // Finally, show the populated list
        showSuccessState();
    }

    // Initial call to start the process
    fetchAndRenderUsers();
});
```

Output:



Observation:

 State-Driven UI: The application's core design is based on managing three states. The HTML provides dedicated containers for each state, and the JavaScript's primary job is to show the appropriate one based on the status of the API request.

 Asynchronous Data Fetching: It uses async/await with the fetch() API, which makes the asynchronous code for fetching data clean and easy to read.

 Enhanced Loading Experience: Instead of a simple "Loading..." message, the code implements a "skeleton screen." This provides a better user experience by showing a placeholder that mimics the final layout, reducing perceived load time.

 Robust Error Handling: The script correctly handles two types of potential failures:

1. **Network Errors:** A try...catch block catches issues like a lost internet connection.
2. **HTTP Errors:** It explicitly checks if response.ok is true, allowing it to handle server-side errors (like a 404 "Not Found" or 500 "Server Error").

 Secure DOM Manipulation: This is a critical security feature. The code avoids using .innerHTML to inject the fetched data. Instead, it uses document.createElement() and sets the content with .textContent. This prevents Cross-Site Scripting (XSS) attacks by ensuring that any malicious code within the API data is rendered as plain text, not executed as HTML.