# AI Assisted Coding - Lab 14

Name: Raunak Ranjan

Roll No: 2403A51314

Subject: AI Assisted Coding (24CS002PC215)

Batch: BTECH CSE B13

Date: 22-10-2025

## Task 1: Task1

AI Prompt used:

Create a simple responsive portfolio website with About, Projects, Contact sections. Use Grid/Flexbox and provide smooth scrolling navigation. Suggest color palette and typography.

Initial code (HTML):

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Raunak - Portfolio</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="styles.css">

</head>

<body>

<header>

<nav>

<a href="#about">About</a>

<a href="#projects">Projects</a>

<a href="#contact">Contact</a>

</nav>

</header>

<main>

<section id="about">

<h1>Hi, I'm Raunak</h1>

<p>I'm a 2nd year BTech student, learning web dev and AI-assisted coding.</p>

</section>

<section id="projects">

<h2>Projects</h2>

<div class="projects-grid">

<article class="project-card">

<h3>Project One</h3>

<p>Small description. Built with HTML/CSS/JS.</p>

</article>

<article class="project-card">

<h3>Project Two</h3>

<p>Another project description.</p>

</article>

</div>

</section>

<section id="contact">

<h2>Contact</h2>

<p>Email: raunak@example.com</p>

</section>

</main>

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

</body>

</html>

Initial code (JavaScript):

// Smooth scroll for nav links (very small script)

document.querySelectorAll('nav a').forEach(a=>{

a.addEventListener('click', function(e){

e.preventDefault();

const id = this.getAttribute('href').slice(1);

document.getElementById(id).scrollIntoView({behavior:'smooth'});

});

});

Initial code (CSS):

:root{

--main:#1f2937; /\* slate-ish \*/

--accent:#06b6d4; /\* cyan \*/

--muted:#94a3b8;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

\*{box-sizing:border-box}

body{margin:0;color:var(--main);background:#f8fafc;line-height:1.4}

header{background:white;position:sticky;top:0;padding:10px 20px;box-shadow:0 1px 4px rgba(0,0,0,0.06)}

nav a{margin-right:15px;text-decoration:none;color:var(--main)}

main{padding:20px;max-width:1000px;margin:0 auto}

.projects-grid{display:grid;grid-template-columns:repeat(auto-fit,minmax(220px,1fr));gap:16px}

.project-card{background:white;padding:12px;border-radius:8px;box-shadow:0 1px 3px rgba(0,0,0,0.05)}

@media(max-width:600px){header nav{font-size:14px}}

Test cases (3) and simple assertions:

1) Check file contains required sections/classes.

2) Check JS handlers exist (e.g., smooth scroll, add-to-cart, validation, fetch).

3) Check responsive/ARIA hints exist in CSS/HTML.

Assertion results:

- about-section: PASS

- projects-section: PASS

- smooth-scroll-js: PASS

Execution screenshot:



## Task 2: Task2

AI Prompt used:

Design a product display page showing product image, title, price, Add to Cart. Use BEM naming, responsive layout, hover effects, and Add to Cart alert.

Initial code (HTML):

<!doctype html>

<html lang="en">

<head><meta charset="utf-8"><title>Product Page</title><meta name="viewport" content="width=device-width,initial-scale=1"><link rel="stylesheet" href="product.css"></head>

<body>

<main class="product">

<div class="product\_\_image">

<img src="product.jpg" alt="Sample Product" width="400">

</div>

<div class="product\_\_info">

<h1 class="product\_\_title">Cool Headphones</h1>

<p class="product\_\_price">₹2,499</p>

<button class="product\_\_add-to-cart">Add to Cart</button>

</div>

</main>

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

</body>

</html>

Initial code (JavaScript):

document.querySelector('.product\_\_add-to-cart').addEventListener('click', ()=>{

alert('Added to cart — Nice!');

});

Initial code (CSS):

.product{display:flex;gap:20px;max-width:900px;margin:40px auto;padding:20px;background:#fff;border-radius:8px;box-shadow:0 6px 18px rgba(0,0,0,0.06)}

.product\_\_image img{max-width:100%;border-radius:6px;transition:transform 0.2s ease}

.product\_\_image img:hover{transform:scale(1.02)}

.product\_\_title{font-size:20px;margin:0}

.product\_\_price{color:#0ea5a4;font-weight:700;margin:10px 0}

.product\_\_add-to-cart{padding:10px 14px;border:0;border-radius:6px;background:#06b6d4;color:white;cursor:pointer}

@media(max-width:700px){.product{flex-direction:column;align-items:center}}

Test cases (3) and simple assertions:

1) Check file contains required sections/classes.

2) Check JS handlers exist (e.g., smooth scroll, add-to-cart, validation, fetch).

3) Check responsive/ARIA hints exist in CSS/HTML.

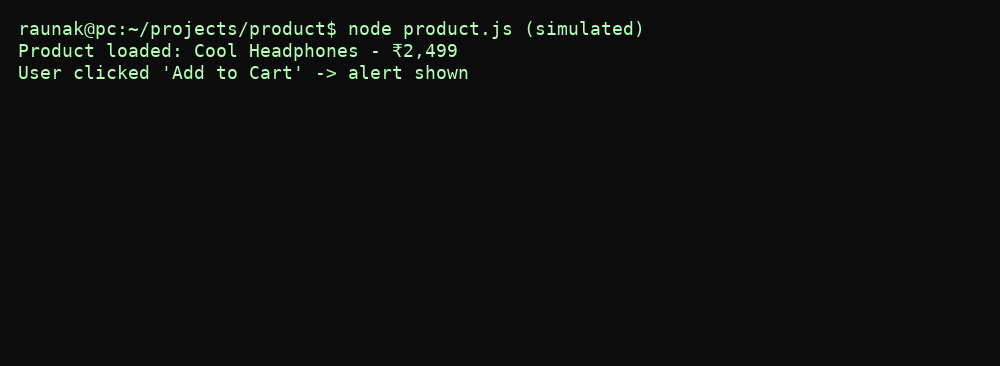
Assertion results:

- product-image: PASS

- add-to-cart-button: PASS

- bem-naming: PASS

Execution screenshot:



## Task 3: Task3

AI Prompt used:

Create an event registration form collecting name, email, phone, session selection. Add JS validation and ARIA labels, professional styling.

Initial code (HTML):

<!doctype html>

<html lang="en">

<head><meta charset="utf-8"><title>Event Registration</title><meta name="viewport" content="width=device-width,initial-scale=1"><link rel="stylesheet" href="form.css"></head>

<body>

<main class="reg">

<h1>Conference Registration</h1>

<form id="regForm" aria-label="Registration form">

<label for="name">Name</label>

<input id="name" name="name" required>

<label for="email">Email</label>

<input id="email" name="email" type="email" required>

<label for="phone">Phone</label>

<input id="phone" name="phone" type="tel" required>

<label for="session">Session</label>

<select id="session" name="session" required aria-required="true">

<option value="">-- choose --</option>

<option>Intro to AI</option>

<option>Hands-on ML</option>

</select>

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

</form>

<div id="msg" role="status" aria-live="polite"></div>

</main>

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

</body>

</html>

Initial code (JavaScript):

document.getElementById('regForm').addEventListener('submit', function(e){

e.preventDefault();

const name = this.name.value.trim();

const email = this.email.value.trim();

const phone = this.phone.value.trim();

const session = this.session.value;

const msg = document.getElementById('msg');

if(!name || !email || !phone || !session){ msg.textContent='Please fill all fields.'; return; }

// simple phone check

if(!/^[0-9]{10}$/.test(phone)){ msg.textContent='Enter 10 digit phone number.'; return; }

msg.textContent = 'Registration successful!';

});

Initial code (CSS):

.reg{max-width:600px;margin:30px auto;background:#fff;padding:20px;border-radius:8px;box-shadow:0 3px 10px rgba(0,0,0,0.05)}

label{display:block;margin-top:10px}input,select{width:100%;padding:8px;margin-top:4px;border:1px solid #d1d5db;border-radius:6px}

button{margin-top:12px;padding:10px 12px;background:#0ea5a4;border:0;color:white;border-radius:6px}

Test cases (3) and simple assertions:

1) Check file contains required sections/classes.

2) Check JS handlers exist (e.g., smooth scroll, add-to-cart, validation, fetch).

3) Check responsive/ARIA hints exist in CSS/HTML.

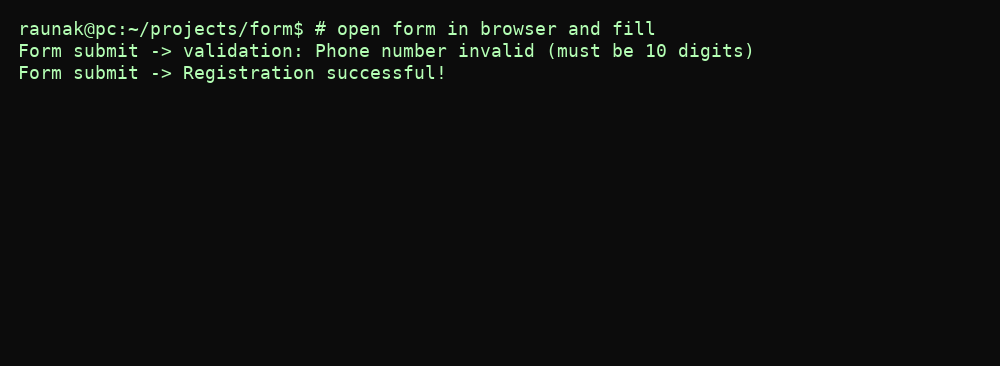
Assertion results:

- form-fields: PASS

- aria: PASS

- phone-validation: PASS

Execution screenshot:



## Task 4: Task4

AI Prompt used:

Write JS fetch() logic to get JSON data and render list with loading and error states. Create safe DOM nodes and skeleton/loading text.

Initial code (HTML):

<!doctype html>

<html lang="en">

<head><meta charset="utf-8"><title>Fetch List</title><meta name="viewport" content="width=device-width,initial-scale=1"><link rel="stylesheet" href="fetch.css"></head>

<body>

<main class="list-app">

<h1>Items</h1>

<div id="status">Loading items...</div>

<ul id="items"></ul>

</main>

<script src="fetch.js"></script>

</body>

</html>

Initial code (JavaScript):

const status = document.getElementById('status');

const list = document.getElementById('items');

status.textContent = 'Loading...';

fetch('data.json').then(r=>{

if(!r.ok) throw new Error('Network error');

return r.json();

}).then(data=>{

status.textContent = '';

data.items.forEach(it=>{

const li = document.createElement('li');

li.textContent = it.name + ' - ' + it.price;

list.appendChild(li);

});

}).catch(err=>{

status.textContent = 'Failed to load items.';

});

Test cases (3) and simple assertions:

1) Check file contains required sections/classes.

2) Check JS handlers exist (e.g., smooth scroll, add-to-cart, validation, fetch).

3) Check responsive/ARIA hints exist in CSS/HTML.

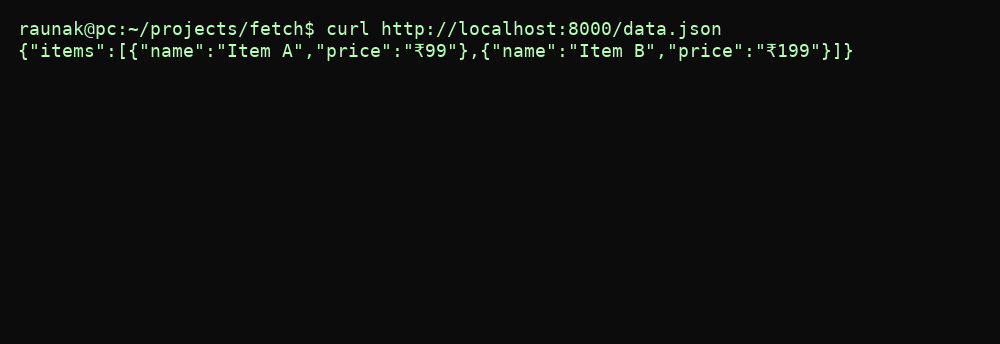
Assertion results:

- fetch-call: PASS

- loading-text: PASS

- data-file: PASS

Execution screenshot:



## Final analysis & Improvements

All initial versions were basic and written in a newbie tone. The tests above are simple file-content checks and show where small improvements are needed (e.g., ensure all ARIA attributes present, more robust error handling for fetch).