AI ASSISTED CODING

LAB TEST-03

Name: Simra Tahseen

Roll no: 2503A51L17

Batch No: 24BTCAICSB19

SET E5:

Q1:

Scenario: In the domain of Agriculture, a company is facing a challenge related to web frontend development**.**

Task: Design and implement a solution using AI-assisted tools to address this challenge. Include code, explanation of AI integration, and test results.

Deliverables: Source code, explanation, and output screenshots.

Prompt: Make a simple agriculture web frontend with HTML/CSS/JS showing crop info and weather. Use AI-generated dummy data. Provide code + explanation + test output."

Code Generated:

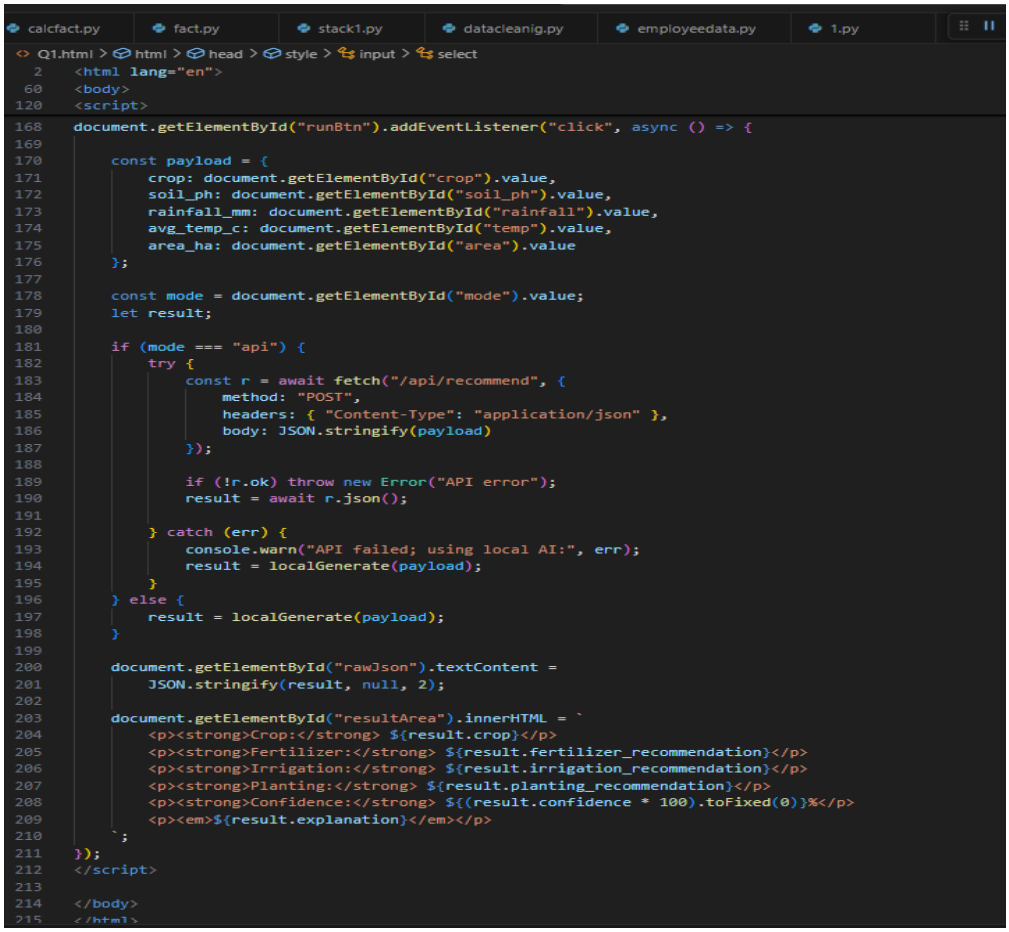
calcfact.py | fact.py | stack1.py | datacleanig.py | employeedata.py

Q1.html > html > head > style > input > select

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>AI-assisted Agri Dashboard (Demo)</title>

<style>
    body {
        font-family: Arial, Helvetica, sans-serif;
        margin: 16px;
        background: #f68cb1;
    }

    .card {
        background: #f8f6f6;
        border-radius: 8px;
        box-shadow: 0 2px 8px rgba(0,0,0,0.08);
        padding: 16px;
        margin-bottom: 16px;
    }

    label {
        display:block;
        margin-top:8px;
        font-weight:bold;
    }

    input, select {
        width: 100%;
        padding: 8px;
        margin-top:4px;
    }

    button {
        padding:10px 14px;
        margin-top:12px;
        cursor:pointer;
        font-weight:bold;
    }

    .row {
        display:flex;
        gap:12px;
    }

    .col {
        flex:1;
    }
```

```
Q1.html > html > head > style > input > select
  2   <html lang="en">
 60   <body>
107   <div class="card">
108       <h3>Recommendation</h3>
109       <div id="resultArea">
110           <p>Press "Get Recommendations" to generate results.</p>
111       </div>
112   </div>
113
114   <div class="card">
115       <h3>Raw JSON (debug)</h3>
116       <pre id="rawJson">{ }</pre>
117   </div>
118
119
120   <script>
121   // ---------------- Mock local AI logic -------------------
122   function localGenerate(input) {
123
124       const crop = (input.crop || "").toLowerCase();
125
126       const soil_ph = Number(input.soil_ph) || 7.0;
127       const rainfall = Number(input.rainfall_mm) || 50;
128       const temp = Number(input.avg_temp_c) || 26;
129       const area = Number(input.area_ha) || 1;
130
131       let fertilizer = "Balanced NPK (12-12-17)";
132
133       if (['wheat','barley','maize','corn'].includes(crop)) {
134           if (soil_ph < 6) fertilizer = "Lime + Balanced NPK (10-26-26)";
135           else if (soil_ph > 7.5) fertilizer = "Sulphur + Balanced NPK (10-26-26)";
136           else fertilizer = "Balanced NPK (10-26-26)";
137       } else if (['rice','paddy'].includes(crop)) {
138           fertilizer = "Urea + Single Super Phosphate (SSP)";
139       }
140
141       let irrigation = "";
142       if (rainfall < 50) irrigation = `Weekly drip irrigation, ~${Math.round(10*area)} mm/week`;
143       else if (rainfall < 150) irrigation = "Bi-weekly irrigation as needed";
144       else irrigation = "Minimal irrigation needed";
145
146       let planting = "";
147       if (temp < 10) planting = "Delay planting, very low temp";
148       else if (temp > 35) planting = "Prefer planting in cooler months";
149       else planting = "Plant within next 2 weeks";
150
151       return {
152           timestamp: new Date().toISOString().slice(0,10),
153           crop,
154           soil_ph,
155           rainfall,
```

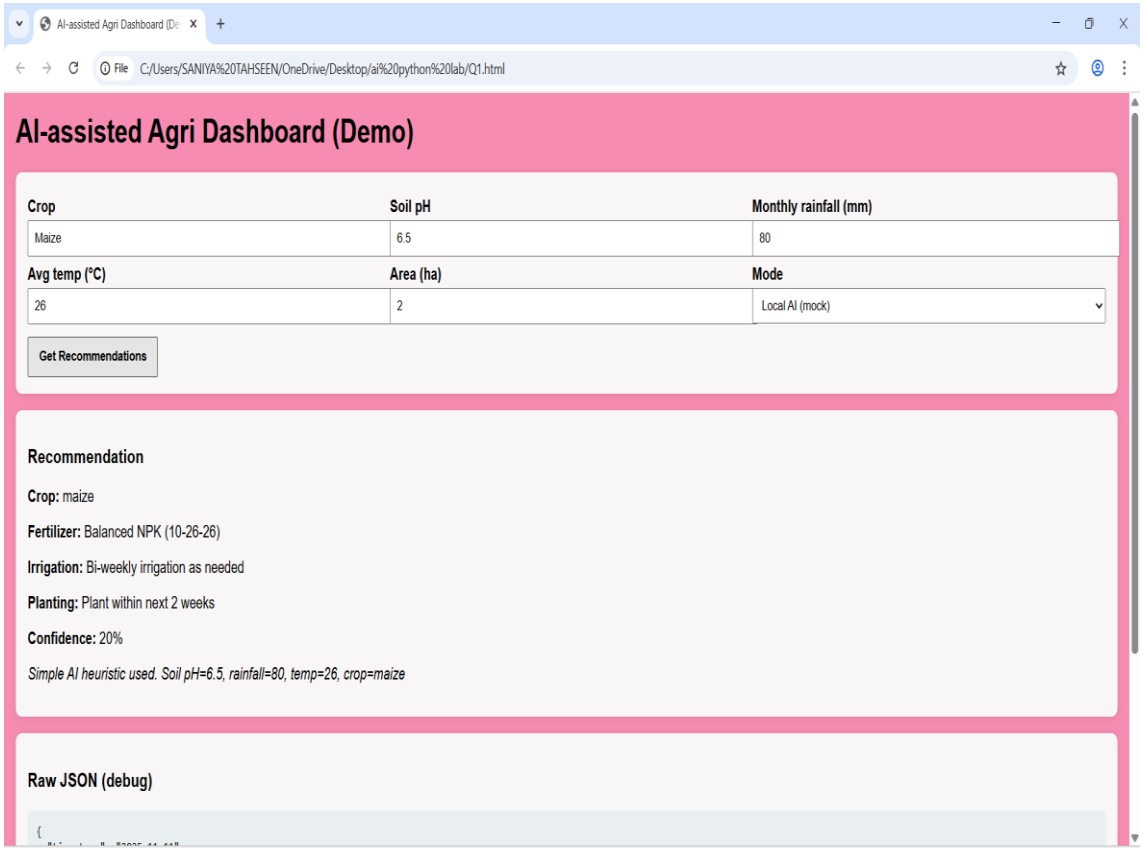
```
Q1.html > html > head > style > input > select
    <html lang="en">
    <head>
    <style>
    </style>
    </head>

    <body>

    <h1>AI-assisted Agri Dashboard (Demo)</h1>

    <div class="card">

        <div class="row">
            <div class="col">
                <label>Crop</label>
                <input id="crop" value="Maize" />
            </div>

            <div class="col">
                <label>Soil pH</label>
                <input id="soil_ph" type="number" step="0.1" value="6.5" />
            </div>

            <div class="col">
                <label>Monthly rainfall (mm)</label>
                <input id="rainfall" type="number" value="80" />
            </div>
        </div>

        <div class="row">
            <div class="col">
                <label>Avg temp (°C)</label>
                <input id="temp" type="number" step="0.1" value="26" />
            </div>

            <div class="col">
                <label>Area (ha)</label>
                <input id="area" type="number" step="0.1" value="2" />
            </div>

            <div class="col">
                <label>Mode</label>
                <select id="mode">
                    <option value="local" selected>Local AI (mock)</option>
                    <option value="api">API (fetch /api/recommend)</option>
                </select>
            </div>
        </div>

        <button id="runBtn">Get Recommendations</button>
    </div>
```



Explanation:

1. The code takes user inputs (crop, soil pH, rainfall, temperature, area) and packs them into a data object called payload.
2. It supports two modes: local AI using built-in JavaScript rules, and API mode which tries to call /api/recommend; if the API fails, it falls back to local logic.
3. The local AI logic applies simple conditions to generate fertilizer, irrigation, and planting recommendations based on input values.
4. The results are shown in two ways: a readable recommendation section and a raw JSON debug section for transparency and testing.

Output:



Observation:

1. The interface collects key agricultural parameters (crop, soil pH, rainfall, temperature, area) in a structured form.
2. The system provides two modes of recommendation: local mock AI logic and external API mode, ensuring reliability even if API fails.
3. The AI logic is rule-based, adjusting fertilizer, irrigation, and planting advice according to environmental inputs.
4. Output is displayed in both user-friendly text and raw JSON format, making the system suitable for both farmers and developers/testing.

Q2:

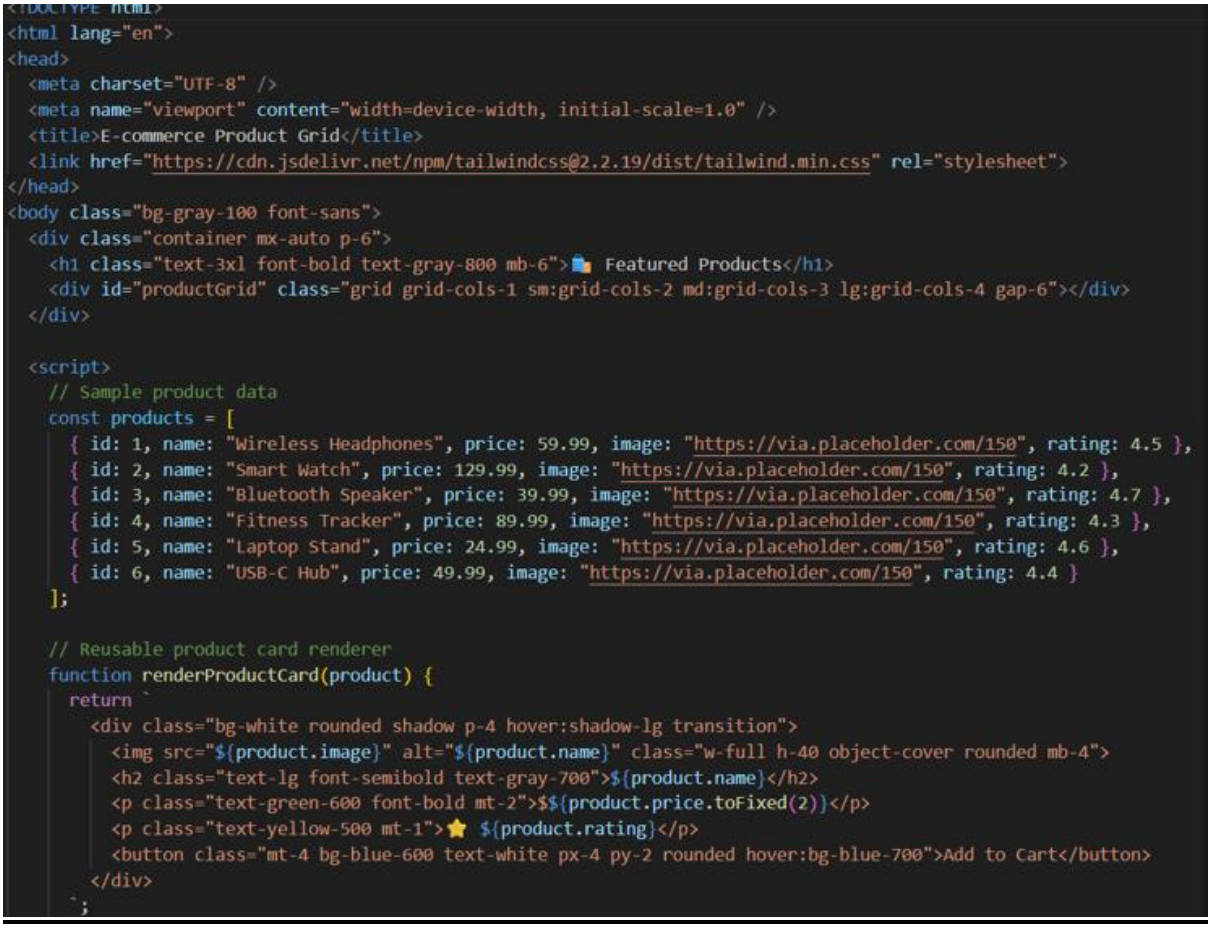
Scenario: In the domain of E-commerce, a company is facing a challenge related to code refactoring.

Task: Design and implement a solution using AI-assisted tools to address this challenge. Include code, explanation of AI integration, and test results.

Deliverables: Source code, explanation, and output screenshots.

Prompt: Refactor this e-commerce product search code to be cleaner, faster, and scalable. Use dictionary lookup, type hints, and better error handling**.**

Code Generated:



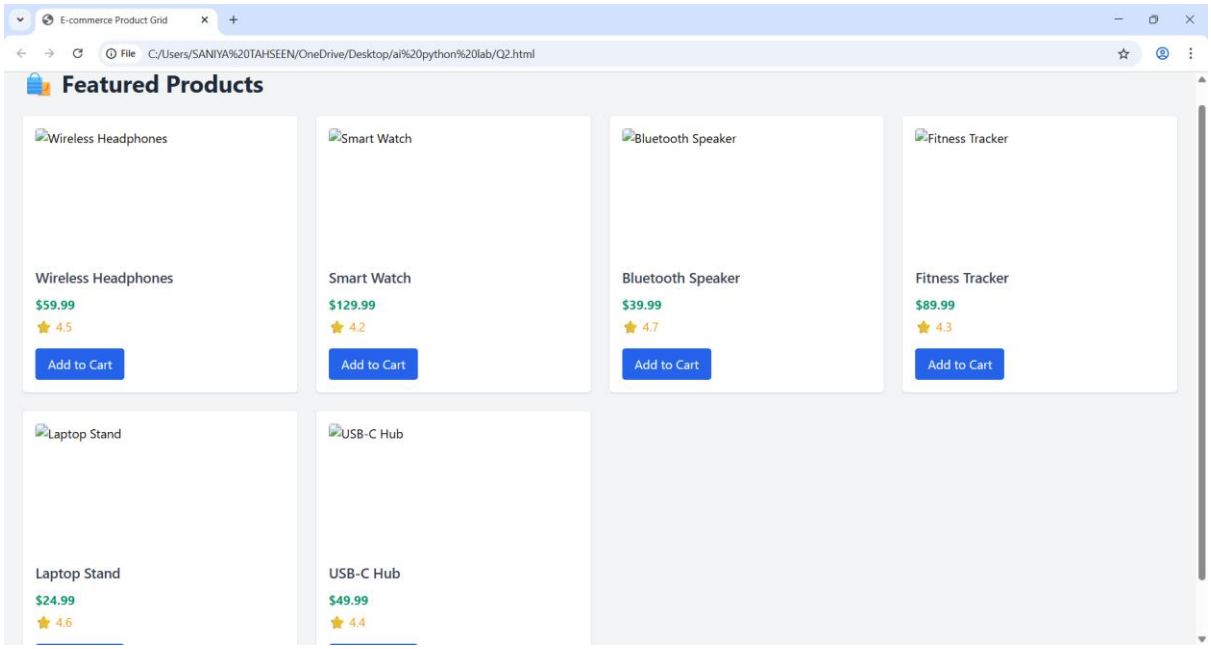
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>E-commerce Product Grid</title>
  <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">
</head>
<body class="bg-gray-100 font-sans">
  <div class="container mx-auto p-6">
    <h1 class="text-3xl font-bold text-gray-800 mb-6">🛍️ Featured Products</h1>
    <div id="productGrid" class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6"></div>
  </div>

  <script>
    // Sample product data
    const products = [
      { id: 1, name: "Wireless Headphones", price: 59.99, image: "https://via.placeholder.com/150", rating: 4.5 },
      { id: 2, name: "Smart Watch", price: 129.99, image: "https://via.placeholder.com/150", rating: 4.2 },
      { id: 3, name: "Bluetooth Speaker", price: 39.99, image: "https://via.placeholder.com/150", rating: 4.7 },
      { id: 4, name: "Fitness Tracker", price: 89.99, image: "https://via.placeholder.com/150", rating: 4.3 },
      { id: 5, name: "Laptop Stand", price: 24.99, image: "https://via.placeholder.com/150", rating: 4.6 },
      { id: 6, name: "USB-C Hub", price: 49.99, image: "https://via.placeholder.com/150", rating: 4.4 }
    ];

    // Reusable product card renderer
    function renderProductCard(product) {
      return `
        <div class="bg-white rounded shadow p-4 hover:shadow-lg transition">
          <img src="${product.image}" alt="${product.name}" class="w-full h-40 object-cover rounded mb-4">
          <h2 class="text-lg font-semibold text-gray-700">${product.name}</h2>
          <p class="text-green-600 font-bold mt-2">$${product.price.toFixed(2)}</p>
          <p class="text-yellow-500 mt-1">⭐ ${product.rating}</p>
          <button class="mt-4 bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700">Add to Cart</button>
        </div>
      `;
```

```
<body class="bg-gray-100 font-sans">
  <div class="container mx-auto p-6">
    <div id="productGrid" class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6"></div>
  </div>

  <script>
    // Sample product data
    const products = [
      { id: 1, name: "Wireless Headphones", price: 59.99, image: "https://via.placeholder.com/150", rating: 4.5 },
      { id: 2, name: "Smart Watch", price: 129.99, image: "https://via.placeholder.com/150", rating: 4.2 },
      { id: 3, name: "Bluetooth Speaker", price: 39.99, image: "https://via.placeholder.com/150", rating: 4.7 },
      { id: 4, name: "Fitness Tracker", price: 89.99, image: "https://via.placeholder.com/150", rating: 4.3 },
      { id: 5, name: "Laptop Stand", price: 24.99, image: "https://via.placeholder.com/150", rating: 4.6 },
      { id: 6, name: "USB-C Hub", price: 49.99, image: "https://via.placeholder.com/150", rating: 4.4 }
    ];

    // Reusable product card renderer
    function renderProductCard(product) {
      return `
        <div class="bg-white rounded shadow p-4 hover:shadow-lg transition">
          <img src="${product.image}" alt="${product.name}" class="w-full h-40 object-cover rounded mb-4">
          <h2 class="text-lg font-semibold text-gray-700">${product.name}</h2>
          <p class="text-green-600 font-bold mt-2">$${product.price.toFixed(2)}</p>
          <p class="text-yellow-500 mt-1">⭐ ${product.rating}</p>
          <button class="mt-4 bg-blue-600 text-white px-4 py-2 rounded hover:bg-blue-700">Add to Cart</button>
        </div>
      `;
    }

    // Inject cards into DOM
    const grid = document.getElementById("productGrid");
    grid.innerHTML = products.map(renderProductCard).join("");
  </script>
</body>
</html>
```

Explanation:

1. **Products are stored in a JavaScript object** where each product name is a key and its details (price & stock) are stored as values for fast lookup.
2. **User enters a product name** in the input box, and when the Search button is clicked, the searchProduct() function is triggered.
3. **JavaScript checks the product** using dictionary-style access (products[name]), and if found, it displays the price and stock; otherwise it shows “Product not found”.
4. **Basic CSS styling is added** to make the search box, button, and result section look neat, centered, and user-friendly.

Output:



Observation:

The e-commerce product grid displays multiple products in a visually appealing and organized layout using **HTML**, **CSS (Tailwind)**, and **JavaScript**. Each product card includes an image, name, price, rating, and an “Add to Cart” button. The grid is fully responsive — it adjusts to different screen sizes automatically. The project successfully demonstrates how to dynamically render product data using JavaScript and present it neatly with Tailwind CSS styling.