Problem Statement: Inventory Cards Web Application

Objective

Build a web application that displays inventory items as cards (3 per row) loaded from a data.json file. The application should allow users to add items to a cart by clicking the cards, update the cart quantities, and compute total costs.

Functional Requirements

- Display Inventory as Cards: Load inventory items from data.json. Show them as cards in a grid layout (3 per row). Each card must include: Item name, Category, Price per unit, Stock count, and Description.
- Add to Cart: Clicking a card should add that item to the cart list. Each click increments the quantity of that item by 1.
- Cart Behavior: Cart items are displayed one below the other. Each cart row shows: Item name, Quantity, Price for that item (quantity × price per unit). A grand total row is displayed at the bottom of the cart.
- Reduce Item Quantity: Clicking a cart row reduces the count of that item by 1. If the count reaches 0, remove the item from the cart.

Page Structure & IDs

HTML IDs (must be used):

- #grid → The inventory grid (cards container).
- #cartList → The cart items container.
- #cartCount → Displays total number of items currently in the cart.

CSS Classes (must be used for styling):

- .container → Page wrapper (centered with padding).
- .grid → Grid layout for item cards.
- .card → Individual inventory item card.
- .row → Flex row for aligning elements inside cards and cart rows.
- .category → Category label (badge/tag).
- .price → Used to style price values.
- .cart → Section wrapper for the cart.
- .cart-list → The container holding cart rows.
- .cart-row → Each individual row in the cart.
- .cart-empty → Message when the cart is empty.
- .muted → Gray text style.
- .small → Small font size text.

CSS Style Requirements

- Theme: Light background (#f9fafb) with white panels and dark text (#111827).

- Grid: 3 columns on desktop (grid-template-columns: repeat(3,1fr)), 2 on tablets, 1 on mobile.
- Cards: White background, border #d1d5db, rounded corners, hover shadow. On hover: slightly lifted (transform: translateY(-2px)), soft shadow.
- Category Badge: Small, rounded pill style, light gray background (#f3f4f6).
- Cart Rows: White background, border, rounded corners. On hover: light gray background. Display item name, quantity, and price aligned.
- Empty Cart: Message styled in muted gray text.

Example Behavior

1. The user clicks on "Notebook A5" card → It appears in the cart as:

> ***Notebook A5 x 1 ₹99***

2. Clicking it again updates to:

> ***Notebook A5 x 2 ₹198***

3. Clicking the cart row decreases count:

> ***Notebook A5 x 1 ₹99***

4. Clicking once more removes it from the cart.

5. The Grand Total row always updates to reflect current totals.

Deliverables for Students:

- index.html (with correct IDs)
- style.css (light theme styles using given classes)
- script.js (cart logic: add, decrement, remove, totals)
- data.json (sample items, at least 5 entries)

Sample Code for Students:

```
'use strict';

/**
 * ============================================================
 *  Inventory Cards — Student Starter (implement TODOs)
 *  NOTE: renderGrid() is fully implemented for you.
 *  Implement: init(), addToCart(), removeOne(), renderCart()
 *  Keep DOM structure/classes as-is (tests rely on them).
 * ============================================================
```

```
 */

/* ------------------ DOM Elements ------------------ */
const els = {
  grid: document.getElementById('grid'),
  cartList: document.getElementById('cartList'),
  cartCount: document.getElementById('cartCount'),
};

/* ------------------ App State ------------------ */
/** Cart structure: plain object keyed by id
 * {
 *   [id]: { id: number, name: string, price: number, qty: number }
 * }
 */
let CART = {};

/* ------------------ REQUIRED: init() ------------------ */
async function init() {
  // TODO: fetch data.json, call renderGrid(items) and renderCart()
}

/* ------------------ PROVIDED: renderGrid() ------------------ */
/**
 * Already implemented for you. Do not modify.
 * - Renders cards (3 per row via CSS grid)
 * - Adds click handler: clicking a card should add item to cart (+1)
 */
function renderGrid(items) {
  els.grid.innerHTML = items
    .map(
      (item) => `
  <article class="card" data-id="${item.id}" data-name="${item.name}" data-
price="${item.price}">
    <div class="row">
      <strong>${item.name}</strong>
      <span class="category">${item.category}</span>
    </div>
    <div class="row" style="margin-top:6px;">
      <span class="price">₹${item.price}</span>
      <span class="muted small">Stock: ${item.stock}</span>
    </div>
    <p class="desc">${item.description}</p>
```

```
    <p class="small muted">Click card to add to cart</p>
  </article>`
    )
    .join('');

  // Clicking a card adds that item to the cart
  els.grid.querySelectorAll('.card').forEach((cardEl) => {
    cardEl.addEventListener('click', () => {
      const id = Number(cardEl.dataset.id);
      const name = cardEl.dataset.name;
      const price = Number(cardEl.dataset.price);

      addToCart({ id, name, price }); // <--- implement below
    });
  });
}

/* ------------------ REQUIRED: addToCart() ------------------ */
/**
 * TODO: Implement addToCart(item)
 * - If item.id not present in CART → create with qty=1
 * - Else increment existing qty by 1
 * - Then call renderCart()
 */
function addToCart(item) {
  // TODO
}

/* ------------------ REQUIRED: removeOne() ------------------ */
/**
 * TODO: Implement removeOne(id)
 * - Decrement qty for that id by 1
 * - If qty reaches 0 → delete CART[id]
 * - Then call renderCart()
 */
function removeOne(id) {
  // TODO:
}

/* ------------------ REQUIRED: renderCart() ------------------ */
/**
 TODO: Implement renderCart()
  - Convert CART object → array of rows
```

```
 - Update #cartCount = sum of all item qty
 - If no rows → show "Cart is empty..." message and return
 - Build cart list HTML with per-item total and grand total:
   <div class="cart-row" data-id="...">
     <span class="name">Item Name</span>
     <span class="qty">x N</span>
     <span class="price">₹ITEM_TOTAL</span>
   </div>
   <hr>
   <div class="cart-row" style="cursor:default;">
     <span class="name">Grand Total</span>
     <span class="price">₹GRAND_TOTAL</span>
   </div>
 - Add click handler to each `.cart-row[data-id]` to call removeOne(id)
*/
function renderCart() {
  // TODO:

}


/* ------------------ OPTIONAL HELPERS (students can use) ------------------ */
/** Format as currency (kept simple to match tests) */
// function formatINR(n) { return `₹${n}`; }

/* ------------------ Example Snippets for Students ------------------ */
1) async/await + fetch JSON:
   async function loadJSON(url) {
     const res = await fetch(url);       // res.ok? res.status?
     if (!res.ok) throw new Error('HTTP ' + res.status);
     return res.json();                  // parsed JSON
   }

  2) Query DOM + event handling:
   const btn = document.querySelector('#myBtn');
   btn.addEventListener('click', (e) => { console.log('clicked'); });

  3) Array methods (map/filter/reduce):
   const arr = [1,2,3];
   const squares = arr.map(x => x * x);
   const evens = arr.filter(x => x % 2 === 0);
   const sum = arr.reduce((s, x) => s + x, 0);

  4) Template strings:
```

```
    const name = 'Notebook';
    const qty = 3;
    const html = `<div>${name} x ${qty}</div>`;

  5) Dataset usage (from DOM):
    const el = document.querySelector('.card');
    const id = Number(el.dataset.id);
    const price = Number(el.dataset.price);
 /
init();
```