Men's Fashion Store Web Application

Objective

Build a responsive, single-page e-commerce web application for men's fashion using AI IDE tools (GitHub Copilot, Cursor, Replit, or ChatGPT) to practice "AI-augmented engineering." The goal is to leverage AI for rapid prototyping and logic generation while maintaining strict adherence to fundamental frontend engineering practices using HTML, CSS, and Vanilla JavaScript (no frameworks).

Business Context

You are a development team (2 members) tasked with creating a Minimal Viable Product (MVP) for "Gentleman's Choice," a new men's fashion startup. The client needs a clean, fast, and functional storefront to showcase their collection and validate the market before investing in a complex backend system.

- **Target Users:** Modern men aged 20-35 who prefer a minimalist, hassle-free shopping experience.
- **Key Business Goal:** Maximize product visibility and streamline the "Add to Cart" process.
- **Success Criteria:**
  - Users can browse and filter products smoothly without page reloads.
  - The shopping cart accurately calculates totals and persists data across sessions.
  - The UI is fully responsive and visually consistent with a premium fashion brand.

Scope of Work (Aligned to SDLC Phases)

1. Requirement Gathering & Analysis

Activities:

- Define functional requirements for a simple e-commerce flow.
- Identify user stories focusing on browsing and cart management.
- Specify technical constraints (Vanilla JS, Mock API).

AI Tool Usage:

- **Primary Tool:** ChatGPT / Claude
- **Example Prompts:**
  - *"Act as a Product Owner. List 5 essential user stories for a men's clothing store MVP, focusing on the shopping cart logic."*
  - *"Generate a JSON schema for a clothing product with fields for size, color, price, and category."*

Deliverables:

- `requirements.md` - Functional requirements (Product Listing, Filtering, Cart Management).
- `user-stories.md` :
    1. *As a user, I want to filter products by category (Shirts, Pants) to find what I need quickly.*
    2. *As a user, I want to add items to my cart and see the total price update instantly.*
    3. *As a user, I want my cart to be saved even if I refresh the page.*

2. Planning & Estimation (Team of 2 - 3 Days)

Activities:

- Break down features into frontend components.
- Assign roles to team members.
- Define the folder structure.

Role Assignment:

- **Developer A (UI & Rendering Lead):** Focus on Layout, Product Grid, Responsive CSS, and DOM rendering logic.
- **Developer B (Logic & State Lead):** Focus on Cart Logic (CRUD), Data Filtering algorithms, and LocalStorage integration.

AI Tool Usage:

- **Example Prompts:**
    - *"Suggest a folder structure for a Vanilla JS e-commerce project to keep logic and styles organized."*
    - *"Create a task list for building a shopping cart feature using JavaScript, estimated for a 3-day sprint."*

Deliverables:

- `project-plan.md` - Task distribution.
- `tech-stack.md` : HTML5, CSS3 (Flexbox/Grid), Vanilla JS (ES6+), LocalStorage.

3. System Design

Activities:

- Design the Mock Data structure.
- Create low-fidelity wireframes for the Home Page and Cart Drawer.
- Plan the CSS architecture (Variables for colors/fonts).

AI Tool Usage:

- **Example Prompts:**

- *"Design a JavaScript object array representing 20 men's fashion items with realistic names and prices."*
- *"Suggest a color palette for a minimalist men's fashion website (Hex codes included)."*

Deliverables:

- `data-model.json` :

```
[
  {
    "id": 1,
    "name": "Classic Oxford Shirt",
    "category": "shirts",
    "price": 45.00,
    "image": "assets/shirt-1.jpg",
    "sizes": ["S", "M", "L"]
  }
]
```

- `wireframes.html` (or sketch).

4. Development

Activities:

- **Setup:** Initialize repository and file structure.
- **UI Implementation:** Build the responsive navbar, hero section, and product grid.
- **Logic Implementation:**
  - Fetch and render products from the mock data.
  - Implement `filter()` logic for categories and price.
  - Implement `addToCart()` : Handle quantity updates for existing items.
  - Implement `removeFromCart()` and dynamic total calculation.
- **Persistence:** Sync cart state with `localStorage` .

AI Tool Usage:

- **Primary Tool:** GitHub Copilot / Cursor / ChatGPT
- **Development Flow Prompts:**
  - *HTML:* "Generate a semantic HTML5 template for an e-commerce product card with an image, title, price, and 'Add to Cart' button."
  - *CSS:* "Write CSS for a responsive grid layout that shows 4 columns on desktop and 1 column on mobile."
  - *JS:* "Write a JavaScript function to format a number as currency (USD) and calculate the total price of an array of cart objects."
  - *JS:* "Refactor this event listener to use Event Delegation for better performance."

Deliverables:

- `index.html` : Main application structure.
- `css/style.css` : All styles (using CSS variables).
- `js/main.js` (or `app.js`): Application logic.
- `js/products.js` : Mock data file.

5. Testing

Activities:

- Cross-device testing (Desktop vs. Mobile).
- Logic verification (Cart totals, duplicate item handling).
- Edge case testing (Empty cart, Zero search results).

AI Tool Usage:

- **Example Prompts:**
  - *"Generate a checklist for testing a JavaScript shopping cart."*
  - *"What are common edge cases for product filtering logic in frontend development?"*

Deliverables:

- `test-plan.md` - List of test cases (e.g., "Add item -> Refresh page -> Item remains").
- `README.md` - Setup instructions and project overview.

Project Evaluation Metrics

Criteria	Excellent (9-10)	Satisfactory (5-6)
Code Structure & Quality	Code is modular (e.g., separate functions for rendering vs. logic). Consistent naming conventions (camelCase). No inline styles.	Monolithic code (one huge function). Inconsistent naming. logic mixed tightly with UI code.
Feature Implementation	**Advanced Logic:** Dynamic cart updates, smart filtering (multi-select), responsive design works perfectly.	**Basic Logic:** Cart adds items but duplicates rows instead of updating quantity. Filtering is buggy.
AI Tool Usage	Evidence of using AI to generate mock data, regex, or optimize algorithms. Code is understood and customized by developers.	Blindly copying AI code without understanding, leading to redundant or "hallucinated" function calls.
UI/UX Design	Professional aesthetic (Grid/Flexbox mastery). Interactive feedback (hover states, toast notifications).	Layout breaks on resize. Poor contrast or alignment. No feedback when buttons are clicked.

Submission Requirements

1. **Source Code:** Complete folder structure (`/css` , `/js` , `/assets`).

2. **Documentation:** `README.md` containing:
    - Project Setup Instructions.
    - List of features implemented.
    - **"AI Usage Report" section:** Briefly describe how you used AI tools to accelerate development (e.g., "Used ChatGPT to generate 50 mock products").