# NAME: AYMAN SHAHEEN

**Roll No: 00476421**

**Template: 1(E-commerce Website)**

**Class Day: Sunday morning**

# ****Marketplace Technical Foundation - Clothing E-commerce Website****

## ****1. Introduction****

### ****1.1 Project Overview****

The Clothing E-commerce Website is designed to provide a seamless online shopping experience, allowing users to browse, filter, and purchase various clothing items. The platform ensures a **pixel-perfect** frontend experience based on the **Figma design** while integrating a robust backend for managing products, orders, and user data.

### ****1.2 Purpose of the Document****

This document outlines the technical foundation of the e-commerce website, ensuring **scalability, efficiency, and professional implementation**. It includes **system architecture, API design, workflows, and data schema** to guide development.

## ****2. System Architecture****

### ****2.1 High-Level Architecture****

* **Frontend:** Built with **Next.js** and **Tailwind CSS** for a modern, responsive UI.
* **Backend:** **Sanity CMS** for managing product inventory, user data, and orders.
* **Authentication:** **Clerk** for secure user sign-in and authentication.
* **Payments:** **Stripe** for secure checkout and transaction processing.
* **Shipping API:** **ShipEngine** to track real-time order shipping status.

### ****2.2 Example Workflow****

1. Users **sign up/log in** via Clerk.
2. Users **browse clothing products** dynamically fetched from Sanity CMS.
3. Users **add products to the cart** and proceed to checkout.
4. Orders are **stored in Sanity CMS**.
5. Payments are **processed through Stripe**.
6. Shipment tracking updates are **fetched from ShipEngine**.

## ****3. Technical Requirements****

### ****3.1 Frontend Requirements****

* **Pixel-perfect UI** based on **Figma design**.
* **Responsive design** for mobile and desktop users.
* **Dynamic product listing** using **Next.js SSR (Server-side Rendering)**.
* Secure **checkout page and order confirmation process**.

### ****3.2 Backend Requirements****

* **Sanity CMS** for managing:
  + **Clothing product data** (name, price, size, stock level, images).
  + **User profiles** and authentication data.
  + **Orders and order history** tracking.

### ****3.3 API Integrations****

* **Payments:** **Stripe** for secure transactions.
* **Shipping:** **ShipEngine** to track real-time orders.
* **CMS Integration:** **Sanity CMS APIs** for managing product data.

## ****4. API Specifications****

| **Endpoint** | **Method** | **Description** | **Response Example** |
| --- | --- | --- | --- |
| /products | GET/POST | Fetch all clothing products or update product data | { "\_id": "123", "name": "T-Shirt", "price": 120 } |
| /orders | POST/GET | Create a new order or fetch order history | { "orderId": "1234", "customer": "Nasir" } |
| /shipment | GET | Fetch shipment details | { "orderId": 123, "status": "In Transit" } |

## ****5. Workflows****

### ****5.1 User Registration****

1. User **signs up/logs in** via Clerk authentication.
2. User data is **securely stored** in Sanity CMS.
3. A **confirmation email** is sent.

### ****5.2 Product Browsing****

1. Users **navigate** through clothing categories.
2. Product data is **fetched dynamically** from Sanity CMS.
3. Users **filter products** by size, price, and brand.

### ****5.3 Order Placement****

1. Users **add items to the cart** and proceed to checkout.
2. Order details are **stored in Sanity CMS**.
3. Payment is **processed via Stripe**.
4. A **confirmation email** is sent.

### ****5.4 Shipment Tracking****

1. Orders are assigned a **tracking ID**.
2. The status is **fetched from ShipEngine** and updated in Sanity CMS.
3. Users can **check their order status in real-time**.

## ****6. Data Schema Design****

### ****6.1 Clothing Product Schema (Sanity CMS)****

export default {

name: 'product',

type: 'document',

fields: [

{ name: 'name', type: 'string', title: 'Clothing Name' },

{ name: 'price', type: 'number', title: 'Price' },

{ name: 'size', type: 'string', title: 'Size' },

{ name: 'stockLevel', type: 'number', title: 'Stock Level' },

{ name: 'images', type: 'array', of: [{ type: 'image' }] }

]

};

### ****6.2 Order Schema****

export default {

name: 'order',

type: 'document',

fields: [

{ name: 'orderId', type: 'string', title: 'Order ID' },

{ name: 'customerId', type: 'string', title: 'Customer ID' },

{ name: 'status', type: 'string', title: 'Order Status' },

{ name: 'createdAt', type: 'datetime', title: 'Order Created At' }

]

};

## ****7. Technical Roadmap****

### ****Phase 1: Frontend Development****

* Implement **pixel-perfect UI** from **Figma using Next.js and Tailwind CSS**.
* Ensure **responsiveness** and accessibility.

### ****Phase 2: Backend Setup****

* Configure **Sanity CMS** for managing **clothing products, users, and orders**.

### ****Phase 3: API Integrations****

* Integrate **Stripe for payments** and **ShipEngine for shipping tracking**.

### ****Phase 4: Testing & Deployment****

* Conduct **end-to-end testing**.
* Deploy the website on **Vercel for scalability**.

## ****8. Conclusion****

This document provides a **clear technical foundation** for your **Clothing E-commerce Website**, ensuring a **scalable, efficient, and professional** implementation using **Next.js, Sanity CMS, Stripe, and ShipEngine**.