Marketplace Technical Foundation - [E-Commerce]

Project Overview

This document outlines the technical design for a clothing e-commerce platform built using **Next.js**, **Tailwind CSS**, **Sanity CMS**, and a custom API for product management and CRUD operations. The platform will allow users to browse, search, and purchase clothing products while providing an intuitive admin interface for managing product details. It will also address the integration of third-party APIs and payment gateways.

System Architecture

Frontend

- Framework: **Next.js**..
- Styling: **Tailwind CSS** for a responsive and modern UI.
- Key Pages:
    1. **Home Page**: Displays featured products and categories.
    2. **Product Listing Page**: Lists all available products.
    3. **Product Details Page**: Detailed view of a single product.
    4. **Cart Page**: Displays selected items for purchase.
    5. **Checkout Page**: Facilitates payment and order placement.
    6. **Admin Dashboard**: Manage CRUD operations on products via Sanity CMS.

Backend

- **Sanity CMS**:
    - Used for backend data storage and content management.
    - Provides CRUD operations for product management.
    - Serves API endpoints to the Next.js frontend.
- **Custom APIs**:
    - One API endpoint will be provided to fetch all products.
    - Additional endpoints will be built for cart management, user authentication, and orders.

Third-Party Integrations

- **Payment Gateway**: Integration with Stripe or PayPal for secure payments.
- **Shipping API**: For real-time shipping rates and tracking.

System Workflow

User Workflow:

- **Browsing Products**:
  - The user lands on the homepage.
  - Navigate to a product listing page via categories or search.
- **Viewing Product Details**:
  - Selects a product to view details.
  - Can add items to the cart from the product details page.
- **Cart and Checkout**:
  - Views the cart, adjusts quantities, or removes items.
  - Proceeds to checkout for payment and order placement.
- **Order Completion**:
  - Receives order confirmation.
  - Tracking information is updated via the shipping API.

Admin Workflow

1. Log into the Sanity CMS dashboard.
2. Manages products (CRUD operations):
   - Add, edit, or delete product information.
   - Upload product.
3. Publishes updates, which are reflected in the frontend via Sanity's API.

Technical Requirements

Frontend Requirements

- **Responsive Design**: Ensures compatibility across mobile, tablet, and desktop devices.
- **Smooth Navigation**: Implemented using Next.js routing.
- **Optimized Images**: Utilize Next.js 'Image' component for optimized loading.
- **State Management**: Implement react hooks (avoiding external libraries).

Backend Requirements

- **Sanity CMS Integration**:
  - Schema for clothing products (e.g., title, description, price, category, stock, and images).
  - Real-time updates to frontend via GROQ queries.
- **Custom API**:
  - One endpoint to fetch all product data.
  - Endpoints for managing orders and user authentication.
- **Scalability**:
  - Modular design to add features such as reviews, wishlists, and coupons in future phases.

API Endpoints

Endpoint	Purpose
/products	Fetch all products
/product/cart	Add items to the cart
/cart/orders/:orderId	Get order details
/order/checkout	Process payment and orders

Database Schema

Products Schema in Sanity

```
{
  "name": "product",
  "type": "document",
  "fields": [
    { "name": "title", "type": "string" },
    { "name": "description", "type": "text" },
    { "name": "price", "type": "number" },
    { "name": "category", "type": "string" },
    { "name": "stock", "type": "number" },
    { "name": "image", "type": "image" }
  ]
}
```

Visualized Architecture Diagram

```
[ User Interface (Next.js + Tailwind CSS) ]
            |
            v
[ Sanity CMS (CRUD Operations) ]
            |
            v
[ Custom APIs (Data Fetching) ]
            |
            v
[ Payment Gateway | Shipping API ]
```

Development Guidelines

- **Code Quality**: clean code with reusable components in Next.js.
- **Security**: Use HTTPS, secure API tokens, and environment variables for sensitive data.
- **Testing**: Perform unit testing for components and integration testing for APIs.
- **Performance**: Optimize images, implement lazy loading, and use server-side rendering for SEO.

Next Steps

1. **Backend Setup**:
    - Configure Sanity CMS schemas.
    - Create API endpoints for product and order management.
2. **Frontend Development**:
    - Build pages and integrate Tailwind CSS for styling.
    - Fetch data from Sanity CMS using GROQ.
3. **Integration**:
    - Connect APIs to the frontend.
    - Set up a payment gateway.
4. **Testing and Deployment**:
    - Perform end-to-end testing.
    - Deploy the application on Vercel or Netlify.
