NIKE ECOMMERCE PLATFORM

SYSTEM ARCHITECTURE FOR MY MARKETPLACE:

The Technical Foundation for my Nike E-commerce Marketplace requires a robust combination of tools, framework and instrastructure to deliver an intuitive , scalable and secure platform.

High-Level Diagram

[Frontend (Next.js)] | v [Sanity CMS] <--------> [Product Data (Mock) API] | ^ | | [Third-Party APIs] <----> [(ShipEngine) Shipment Tracking API] | | | v | [Payment Gateway (Stripe)] | v [Authentication (Clerk)]

1. FRONT-END REQUIREMENTS:

- User Interface : To build a visually engaging, responsive and user-friendly experience.

FRAMEWORK **Next.js: Next.js** serves as the framework for building a responsive and interactive user interface. It enables seamless product browsing, order management, and user authentication. It also efficiently fetches and displays real-time data from backend APIs.

STYLING **TAILWIND CSS: Tailwind Css** is a utility-first CSS framework that simplifies the styling process by providing pre-defined classes. It allows for rapid development of responsive, aesthetically pleasing designs.

LIBRARY **SHADCN: Shadcn (shadcn/ui)** is a modern, component-based UI library that focuses on building highly customizable and accessible components using **React** and **Tailwind CSS**.

- Essential Pages:
  - **HOMEPAGE**: To highlight products and promotions.
  - **PRODUCT LISTING**: To display available products with filters and sorting.

- **PRODUCT DETAILS**: To provide detail for each product, where user can get options to select color and size of their choice and also can increase or decrease product quantities.
- **ADD TO CART**: To show summary of selected items as per quantity for purchase.
- **CHECKOUT**: To provide secure form for finalizing purchases.
- **ORDER COMFIRMATION**: For acknowledgement of successful transactions.

---

2. SANITY CMS as BACKEND:

I will use **Sanity CMS** as the Database to manage and organize key information for my marketplace.

- **PRODUCT DETAILS:** To store details of my products like product_ID, product_name, product_description, product_image, product_price, product_category.
- **CUSTOMER DETAILS:** To manage information such as customer_id, customer_name, customer_details and order histories.
- **ORDER RECORDS:** To store details of placed orders, including items purchased, items_quantity, total_price and status.

PRODUCTS SCHEMA

product_ID: string	Product ID
name: string	Product Name
description: string	Detailed Product Description
price: string	Product Price
sizes: array	Different options for Sizes
colors: array	Different options for Colors
Image: string	The Product Image Url
stock: number	Product quantity in Stock

CUSTOMERS SCHEMA

customer_id: string	Customer ID
customer_name: string	Customer Name
address: string	Customer’s Shipping Address
contact_number: number	Customer Contact Number
city: string	Customer’s City
country: string	Customer’s Country
zipcode: number	Postal/zip code for the address

ORDERS SCHEMA:

customer_ID: string	Link to the Customers who placed order
products_quantity:number	Number of Products Customer selected
products: array	List of purchased Products
total_price: string	Total amount of Products
order_status: string	Order Status (e.g: ‘Pending’, ‘Shipped’)
order_date: datetime	Order Placement Date

THIRD-PARTY APIs:

- **SHIPMENT [Tracking]:** For Shipment I will use **Shipengine** . **ShipEngine** is a shipping API that allows you to track shipment, get rates and manage logistics.
- **PAYMENT[Gateway]:** I will use **Stripe** payment processing API for all online transactions. **Stripe** is a powerful payment processing platform that can be seamlessly integrated into your e-commerce project to handle online transactions.
- **BACKEND AUTHENTICATION:** I will use **Clerk** as backend authentication solution. **Clerk** is a modern user management and

authentication platform designed to simplify the integration of user accounts into web applications. It provides a comprehensive set of tools for handling user registration, authentication, and profile management.

Key Workflows for E-Commerce Platform

1. User Authentication

Process:

- Users register or log in via the frontend using **Clerk**.
- Authentication tokens are generated and stored securely.
- User profiles (name, email, preferences) are synced with **Sanity CMS** for personalization.

2. Product Exploration

Process:

- Users browse product categories or search for items using the frontend interface.
- The frontend sends requests to **Sanity CMS** via APIs to fetch product details (title, price, reviews, stock status, and images).
- Recommended or trending products are displayed dynamically based on user behavior or preferences.

---

3. Checkout and Order Finalization

Process:

- Users add items to their cart, adjust quantities, and review their order summary.
- Shipping details and user preferences (e.g., gift wrap, expedited shipping) are collected.

- The finalized order (products, quantities, payment details) is sent to **Sanity CMS**.
- Payment is processed through **Stripe**, with multi-currency and discount code support.
- Users receive an order confirmation email, and the order status is updated in the CMS.

---

4. Delivery Tracking

Process:

- Upon order dispatch, shipping information is updated in **Sanity CMS** using **ShipEngine API**.
- The user receives a tracking link and estimated delivery time via email or SMS.
- Real-time updates, such as current location and delivery milestones, are displayed in the user's account dashboard.

---

5. Stock and Wishlist Management

Process:

- Inventory levels are automatically synchronized in **Sanity CMS** when purchases are made or new stock is added.
- Products that go out of stock display a "Notify Me" button, allowing users to subscribe to stock alerts.
- Users can save unavailable products to their wishlists for later purchase.

GENERAL E-COMMERCE WORKFLOWS:

1. Product Browsing

Endpoint: /products
Workflow:

1. Users visit the "Products" or "Categories" section on the website via the frontend.
2. The frontend retrieving product information stored in **Sanity CMS**.
3. The server responds with product details like id, name, price, description, availability, category, and image links.
4. The frontend dynamically displays the product listings, incorporating features such as advanced filtering (e.g., by price range, categories, or ratings) and sorting options (e.g., "Price: Low to High").

2. Cart Management

- **Endpoint**: /cart
- **Process**:
  1. Users click "Add to Cart" on a product.
  2. The server validates stock availability through **Sanity CMS** before adding the item to the user's cart.
  3. Users can view, update (e.g., quantity adjustments), or remove items through the cart page.

3. Order Placement

- **Endpoint**: /checkout
- **Process**:
  1. When users add their product to cart, they proceed to the checkout page.
  2. They need to provide details like shipping address, contact info and displays an order summary.
  3. The backend:
     - Verifies stock availability and calculates the total amount.
     - Processes payment using **Stripe**.
     - Records the order in **Sanity CMS**.

4. Shipment and Order Interaction

- **Endpoint**: /shipment-tracking

- **Process**:
    1. Users access "My Orders" dashboard to manage their orders.
    2. A GET request is sent to the /live-shipment-tracking endpoint, connecting to **ShipEngine** for real-time location updates and estimated delivery times.
    3. Users can view live tracking updates directly on the user interface.

5.Inventory Management

- **Endpoint**: /inventory
- **Process:**

1. Inventory levels are tracked and updated in Sanity CMS.

2. Live stock information is retrieved directly from Sanity CMS.

3. Products that are unavailable are moved to the wishlist rather than the cart

4. Items in stock can be added to the cart and purchased during checkout.

API END POINTS:

ENDPOINTS	METHODS	PURPOSE	RESPONSE/PAYLOAD
/products	GET	Fetches all product details	{ “product_id”:”prod1”, “name”:”Nike Air Force 1 Mid 07”, “price”:“10,795.00”, ”sizes”:[“UK6”, “UK7”, “UK8”], “image”:https://example.com/product.jpg”,“stock”: 90, “category”: “Men’s Shoes”, “tag”: “Just In”, “colors”: [“white”, “grey”, “red”] }

/orders	POST	Creates a new order with customer and product details	{ “orderID”:“order123”, ”order_status”:”success”, “quantity”:2, “total_price”: “11,590.00” }
/shipment	GET	Tracks shipment status for an order	{ “shipment_ID”:”ship234”,”status”:”In-Transit”,”expectedDeliveryDate”20-01-2025”, “actualDeliveryDate”: “18-01-2025” }
/deliverystatus	GET	Fetch delivery informtion	{ “order_ID”:”order23”,”delivery_time”:”7days” }
/inventory	GET	Fetch real-time stock levels	{ “product_id”:”prod1”, “stock”: 20 }
/cart	POST	Add product to cart	{ “cart_ID”:”cart415”,”products”:[“Nike Air Max”] }
/wishlist	POST	Add products to wishlist	{ “wishlist_ID”:”wishlist234”,”products”:[“Nike Standard Issue”, “Nike Court Vision Low” }

SANITY SCHEMA:

```
import { Rule } from "@sanity/types";
export default {
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    {
      name: "name",
      title: "Name",
      type: "string",
    },
    {
      name: "image",
      title: "Image",
      type: "image",
      options: {
        hotspot: true,
      },
    },
    {
      name: "description",
      title: "Description",
      type: "text",
    },
    {
      name: "price",
      title: "Price",
      type: "string",
    },
    {
      name: "sizes",
      title: "Sizes",
      type: "array",
      of: [{ type: "string" }],
     /* options: {
        list: [
          { title: "UK 6 (EU 40)", value: "6" },
          { title: "UK 6.5", value: "6.5" },
          { title: "UK 7", value: "7" },
          { title: "UK 7.5", value: "7.5" },
          { title: "UK 8", value: "8" },
          { title: "UK 8.5", value: "8.5" },
          { title: "UK 9", value: "9" },
          { title: "UK 9.5", value: "9.5" },
          { title: "UK 10", value: "10" },
          { title: "UK 10.5", value: "10.5" },
          { title: "UK 11", value: "11" },
          { title: "UK 11.5", value: "11.5" },
          { title: "UK 12", value: "12" },
        ],
      }, */
    },
    {
      name: "rating",
      title: "Rating",
      type: "number",
      validation: (Rule: Rule) =>
        Rule.min(0).max(5).error("Rating must be between 0 and 5"),
    },
    {
      name: "stockQuantity",
      title: "Stock Quantity",
      type: "number",
    },
    {
      name: "discount",
      title: "Discount",
      type: "number",
    },
    {
      name: "department",
      title: "Department",
      type: "array",
      of: [{ type: "string" }],
    },
    {
      name: "totalOrders",
      title: "Total Orders",
      type: "number",
    },
    {
      name: "brand",
      title: "Brand",
      type: "string",
    },
    {
      name: "category",
      title: "Category",
      type: "string",
    },
    {
      name: "color",
      title: "Color",
      type: "array",
      of: [{ type: "string" }],
    },
    {
      name: "details",
      title: "Details",
      type: "string",
    },
    {
      name: "style",
      title: "Style",
      type: "string",
    },
    {
      name: "tag",
      title: "Tag",
      type: "string",
    },
    {
      name: "id",
      title: "ID",
      type: "string",
    },
  ],
};
```

Technical Roadmap

This document outlines the technical roadmap for the **NIKE ECOMMERCE MARKETPLACE**. It covers the development, testing, and launch phases, along with key features and workflows.

Conclusion

The Nike E-commerce Marketplace is a cutting-edge platform designed to provide users with a seamless and dynamic shopping experience. With features such as real-time inventory tracking, personalized recommendations through wishlist functionality, and express delivery updates, the platform caters to the modern shopper's expectations. The responsive design ensures accessibility across devices, while advanced filtering and sorting options allow users to browse and shop effortlessly.