# **Day 2: Marketplace Technical Foundation - Nike-Style Shoe Marketplace**

## **System Architecture Overview**

Diagram:

[Frontend (Next.js)]

│

├── [Sanity CMS] → Manages products, orders, customers

│ └── Product Data API

│

├── [Payment Gateway API] → Processes payments

│

└── [Shipment Tracking API] → Manages delivery status

Component Roles:

* Frontend (Next.js): User interface for browsing products, cart management, and checkout.
* Sanity CMS: Backend for storing product details, customer data, and order records.
* Payment Gateway API: Handles secure transactions (e.g., Stripe, PayPal).
* Shipment Tracking API: Provides real-time delivery updates (e.g., FedEx, DHL).

## **Key Workflows**

### **1. Product Browsing**

1. User visits the homepage and views shoe categories (Sports, Sneakers, Formal).
2. Frontend fetches product data from Sanity CMS via /products API.
3. Products are displayed with images, prices, sizes, and colors.

### **2. Order Placement**

1. User adds shoes to the cart.
2. Frontend sends a POST request to /orders endpoint to save order details in Sanity.
3. Payment Gateway API processes the transaction.
4. Order confirmation is stored in Sanity and displayed to the user.

### **3. Shipment Tracking**

1. After payment, shipment details are sent to the Shipment Tracking API.
2. Frontend fetches delivery status via /shipment endpoint.
3. Real-time updates (e.g., "In Transit," "Delivered") are shown to the user.

## **Sanity Schema Design**

### **1. Product Schema**

// schemas/product.js

export default {

name: 'product',

type: 'document',

title: 'Product',

fields: [

{ name: 'productId', type: 'string', title: 'Product ID' },

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

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

{ name: 'stock', type: 'number', title: 'Stock' },

{ name: 'category', type: 'string', title: 'Category', options: { list: ['Sports', 'Sneakers', 'Formal'] } },

{ name: 'sizes', type: 'array', title: 'Size Options', of: [{ type: 'string' }] },

{ name: 'colors', type: 'array', title: 'Color Options', of: [{ type: 'string' }] },

{ name: 'description', type: 'text', title: 'Description' },

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

],

};

### **2. Order Schema**

// schemas/order.js

export default {

name: 'order',

type: 'document',

title: 'Order',

fields: [

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

{ name: 'customer', type: 'reference', to: [{ type: 'customer' }], title: 'Customer' },

{ name: 'products', type: 'array', title: 'Products', of: [{ type: 'reference', to: [{ type: 'product' }] }] },

{ name: 'quantity', type: 'number', title: 'Quantity' },

{ name: 'totalPrice', type: 'number', title: 'Total Price' },

{ name: 'status', type: 'string', title: 'Status', options: { list: ['Pending', 'Shipped', 'Delivered'] } },

{ name: 'timestamp', type: 'datetime', title: 'Order Date' },

],

};

### **3. Customer Schema**

// schemas/customer.js

export default {

name: 'customer',

type: 'document',

title: 'Customer',

fields: [

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

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

{ name: 'email', type: 'string', title: 'Email' },

{ name: 'phone', type: 'string', title: 'Phone' },

{ name: 'address', type: 'text', title: 'Address' },

{ name: 'orderHistory', type: 'array', title: 'Order History', of: [{ type: 'reference', to: [{ type: 'order' }] }] },

],

};

## **API Endpoints**

| **Endpoint** | **Method** | **Description** | **Response Example** |
| --- | --- | --- | --- |
| /products | GET | Fetch all products | [{ "id": "001", "name": "Air Max", "price": 150, "category": "Sneakers", ... }] |
| /orders | POST | Create a new order | { "orderId": "123", "status": "Pending", "totalPrice": 300 } |
| /shipment/{orderId} | GET | Track shipment status | { "status": "In Transit", "ETA": "2025-01-20", "courier": "DHL" } |
| /payment | POST | Process payment | { "paymentId": "PAY-789", "status": "Completed", "transactionDate": "2025-01-18" } |

## **Industry Best Practices**

1. Sanity CMS Optimization: Use real-time previews to manage product listings efficiently.
2. Scalable Frontend: Implement Next.js static generation for fast product pages.
3. API Security: Encrypt sensitive data (e.g., payment details) using HTTPS and JWT tokens.