**Author:**

**RABIA REHMAH,**

**Online Shopping.**

**Technical Foundation and Enhanced Workflow**

**Fronted (Next.js)**

**Sanity(cms) Third party API**

**Order data Shipment tracking Payment getaway**

**System Architecture Overview**

**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)]

**Component Descriptions**

**Frontend (Next.js):**

Provides a responsive and interactive user interface for browsing products, managing orders, and handling user authentication.

Fetches and displays data from the backend APIs in real-time.

**Sanity CMS:**

Centralized backend for managing product information, user data, order records, and inventory. Exposes APIs for dynamic data communication with the frontend.

**Third-Party APIs:**

1. **Shipment Tracking API (ShipEngine):** Fetches real-time shipping updates and generates tracking details.
2. **Payment Gateway (Stripe):** Processes secure transactions and confirms payment status.

**Authentication (Clerk):**

Handles user registration, login, and session management.

Integrates with Sanity CMS to store user data securely.

# Key Workflows

**User Registration**

**Process:**

User signs up via the frontend using Clerk.

Registration details are stored in Sanity CMS.

# Key Workflows

**User Registration**

**Process:**

User signs up via the frontend using Clerk.

Registration details are stored in Sanity CMS.

**Product Browsing**

**Process:**

User navigates through product categories on the frontend.

Sanity CMS API fetches product data (name, price, stock, description, images). Dynamic product listings are displayed on the frontend.

**Order Placement**

**Process:**

User adds products to the cart and proceeds to checkout.

Order details (products, quantities, shipping address) are sent to Sanity CMS.

Payment is processed via Stripe.

A confirmation message is sent to the user's email, and the order is recorded in Sanity CMS.

**Shipment Tracking**

**Process:**

After order placement, shipment details are updated using ShipEngine.

Real-time tracking information is displayed to the user on the frontend.

**Inventory Management**

**Process:**

Product stock levels are managed in Sanity CMS.

Real-time stock updates are fetched from Sanity CMS.

Out-of-stock products are added to wishlist instead of cart.

In-stock products can be added to cart and proceed to checkout.

# Category-Specific Instructions

**General E-commerce**

**Focus on:**

Standard product browsing, cart management, wishlist, and order placement. Inventory management and real-time stock updates.

**Workflow Example:**

**Endpoint:** /products

**Method:** GET

**Purpose:** Fetch all product listings.

**API Endpoints**

**Endpoint : Method: Purpose: Response Example**

/products GETFetch all product details [ { "name": "Product Name", "slug": "product-slug",

"price": 100 } ]

/order POST Submit new order details { "orderId": 123, "status": "success" }

/shipment-tracing GETFetch real-time tracking updates { "trackingId": "AB123", "status": "In Transit" }

/delivery-status GETFetch express delivery tracking information { "orderId": 456, "deliveryTime": "30 mins" }

/inventory GETFetch real-time stock levels { "productId": 789, "stock": 50 }

/cart POST Add product to cart { "cartId": 101, "items": [...] }

/wishlist POST Add product to wishlist { "wishlistId": 202, "items": [...] }

**Technical Roadmap Development Phase:**

**Authentication:**

Implement user registration and login using Clerk.

Integrate Clerk with Sanity CMS for user data storage.

**Product Management:**

Create mock API for product data.

Store product data in Sanity CMS.

Fetch and display product data on dynamic frontend pages.

Cart and Wishlist:

Implement add-to-cart functionality with real-time stock checks.

Allow out-of-stock products to be added to wishlist.

Display total bill and proceed to checkout button on cart page.

**Payment Integration:**

Integrate Stripe for secure payments.

Use Stripe test account for development.

Handle payment success and failure scenarios.

**Shipment Tracking:**

Integrate ShipEngine for shipment tracking.

Generate tracking numbers and display on the frontend.

Allow users to track their orders in real-time.

**Inventory Management:**

Create API for real-time stock updates in Sanity CMS. Update stock levels upon order placement.

Prevent out-of-stock products from being added to cart.

**Testing Phase:**

**End-to-End Testing:**

Test all workflows, including user registration, product browsing, cart management, checkout, and shipment tracking.

Validate API responses and ensure data accuracy.

**Security Audits:**

Conduct security audits for sensitive data handling, including user authentication and payment processing.

**Launch Phase:**

**Deployment:**

Deploy the platform on a cloud hosting service (e.g., Vercel, Netlify).

Monitor user feedback and optimize for performance.

**Post-Launch:**

Collect user feedback for continuous improvement.

Optimize API performance and frontend loading times.

Scale infrastructure based on traffic and demand.

**Conclusion:**

This technical foundation outlines the architecture, workflows, and API endpoints for the Bandage Online Shopping platform. By following the enhanced workflow and technical roadmap, the platform will provide a seamless eCommerce experience with robust authentication, inventory management, and shipment tracking features.