DAY 2 PLANNING THE TECHNICAL FOUNDATION

Technical Requirements

Frontend Requirements:

Pages and Features:

Home Page:

Hero section with the latest Nike shoes.

Categories for easy navigation (e.g., Running, Sports, Casual).

Promotional banners for discounts and new arrivals.

Product Listing Page:

Filters (e.g., size, price range, color, category).

Sort options (e.g., by price, popularity, or newest arrivals).

Product Details Page:

High-resolution images with zoom functionality.

Detailed descriptions (material, features, etc.).

Customer reviews and ratings.

Add-to-cart button with size selection.

Cart Page:

Editable cart (change quantity, remove items).

Display subtotal, taxes, and final total.

Proceed to checkout button.

Checkout Page:

Secure form for billing and shipping information.

Payment method selection (e.g., Credit/Debit Card, PayPal).

Order summary for user verification.

Order Confirmation Page:

Display confirmation message with order details.

Provide an estimated delivery date.

User Account Page:

Login/Signup functionality (via email or social logins like Google).

Order history and tracking.

Wishlist for saving products.

User Experience:

Mobile-first responsive design to ensure smooth browsing across devices.

Fast-loading pages with optimized assets for better performance.

Intuitive navigation for effortless user flow.

Dynamic Search:

Real-time search bar to quickly find Nike shoes based on keywords.

Accessibility:

Ensure WCAG-compliant design (e.g., proper color contrast, keyboard navigation).

Backend Requirements:

Sanity CMS Schemas:

Product Schema:

Fields: id, name, price, stock, sizes, category, images, description, rating.

Order Schema:

Fields: orderId, customerName, address, products, totalAmount, paymentStatus, shipmentStatus.

Customer Schema:

Fields: id, name, email, password, orderHistory, wishlist.

Product Inventory Management:

Maintain real-time stock updates when users place orders.

Security:

Implement role-based access for admins to manage products, orders, and customers.

Use secure authentication methods (e.g., OAuth for social logins).

Scalability:

Design the backend to handle a growing number of users and products.

Third-Party Integrations:

Payment Gateway API:

Use services like Stripe to securely process payments.

Support multiple payment methods (e.g., cards, digital wallets).

Shipment Tracking API:

Real-time updates on order delivery status.

Send tracking links to customers via email.

Email/SMS Notifications:

Notify users about order confirmations, shipment details, and delivery status.

System Architecture Flow

Frontend (Next.js):

Users interact with the interface to:

Browse products.

Add items to the cart.

Place orders.

Track shipments.

Sanity CMS (Backend Database):

Stores and manages:

Product details.

Customer data.

Orders.

Functionality:

Sends product data to the frontend for display.

Receives order details from the frontend and stores them.

Payment Gateway API (Strip):

Processes user payments securely.

Sends payment confirmation back to the frontend.

Shipment Tracking API:

Provides real-time updates about delivery status.

Sends the tracking information to the frontend.

Workflow

User (Frontend) → Product Data Request → Sanity CMS

^ |

| v

User (Frontend) → Order Details → Sanity CMS

^ |

| v

User (Frontend) → Payment Info → Payment Gateway API → Payment Confirmation → User (Frontend)

^ |

| |

| |

| |

| v

User (Frontend) → Tracking Data Request → Shipment Tracking API → Tracking Updates → User (Frontend)

API Requirements

1. Product APIs

GET /products

Purpose: Fetch the list of all available Nike shoes.

Response Example:[

{

“id”: 1,

```
  “name”: “Nike Air Max”,
  “price”: 120,
  “sizes”: [7, 8, 9, 10],
  “category”: “Running”,
  “image”: “url_to_image”,
  “rating”: 4.5,
  “stock”: 50
 }
]
```


GET /products/{id}

Purpose: Fetch details of a specific shoe.

Path Parameter: id (Product ID).

Response Example:


```
{
 “id”: 1,
 “name”: “Nike Air Max”,
 “price”: 120,
 “sizes”: [7, 8, 9, 10],
 “description”: “Comfortable and stylish running shoes”,
 “image”: “url_to_image”,
 “rating”: 4.5,
 “reviews”: [
  {“user”: “John”, “comment”: “Great shoes!”, “rating”: 5}
 ]
}
```


2. Order APIs

POST /orders

Purpose: Save a new order in the system.

Payload Example:


```
{
  “customerId”: 101,
  “products”: [
    {“productId”: 1, “quantity”: 2},
    {“productId”: 3, “quantity”: 1}
  ],
  “totalAmount”: 300,
  “paymentStatus”: “Pending”,
  “address”: “123 Main Street, NY”
}
```


Response Example:


```
{
  “orderId”: 2001,
  “status”: “Order Placed”,
  “paymentLink”: “url_to_payment_gateway”
}
```


GET /orders/{orderId}

Purpose: Fetch details of a specific order.

Path Parameter: orderId (Order ID).

Response Example:


```
{
  “orderId”: 2001,
  “customerName”: “John Doe”,
  “products”: [
    {“name”: “Nike Air Max”, “quantity”: 2, “price”: 120}
```

```
 ],
 “totalAmount”: 300,
 “shipmentStatus”: “In Transit”,
 “expectedDelivery”: “2025-01-25”
}
```


3. **Payment APIs**

POST /payments

Purpose: Process payment for an order.

Payload Example:


```
{
 “orderId”: 2001,
 “paymentMethod”: “Credit Card”,
 “amount”: 300
}
```


Response Example:


```
{
 “paymentId”: “abc123”,
 “status”: “Success”,
 “transactionDetails”: {
  “transactionId”: “txn5678”,
  “amount”: 300,
  “timestamp”: “2025-01-17T10:30:00Z”
 }
}
```


4. **Shipment APIs**

GET /shipment/{orderId}

Purpose: Fetch real-time shipment tracking updates.

Path Parameter: orderId (Order ID).

Response Example:

{

“shipmentId”: “shp001”,

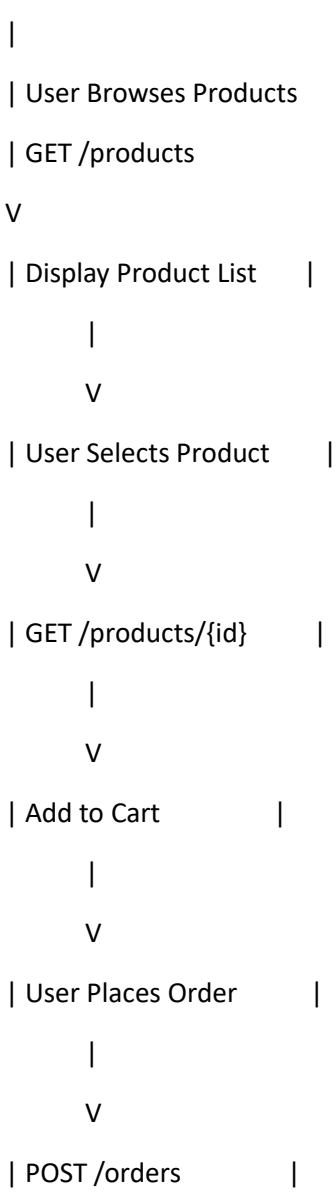
“status”: “In Transit”,

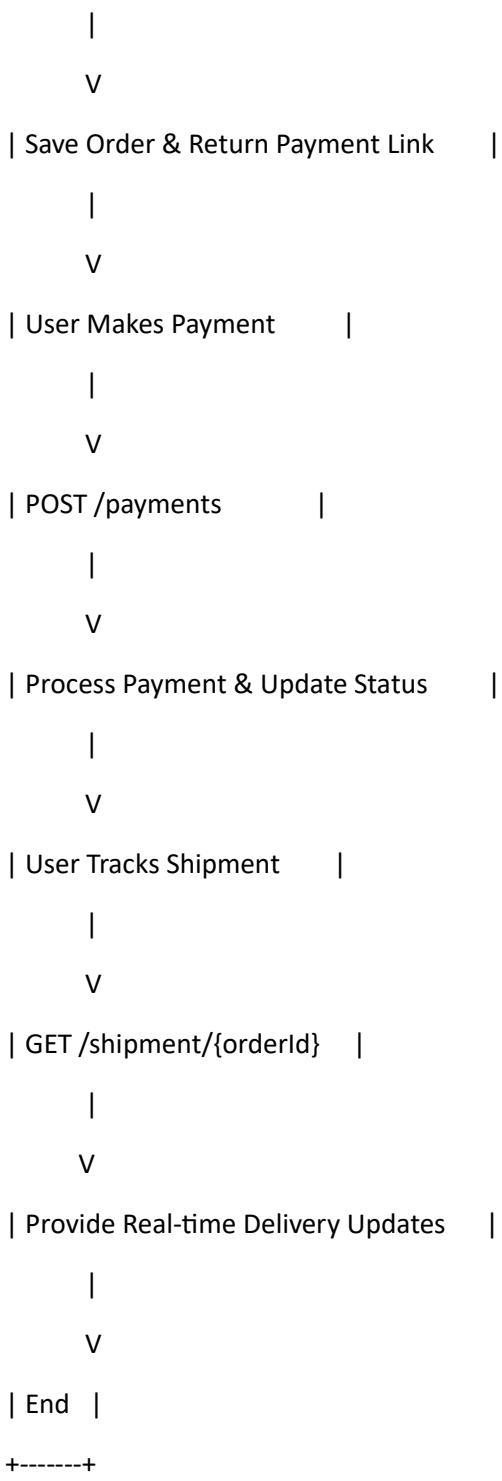
“location”: “New York Warehouse”,

“estimatedDelivery”: “2025-01-25”

}

Workflow





Technical Documentation

System Architecture Diagram: Visual representation of interactions.

Workflows: Step-by-step description of user interactions, such as browsing shoes, adding to the cart, and completing orders.

API Endpoints: List of endpoints with their purposes, methods, and expected responses.