Marketplace Technical Foundation - Nike Clone

1. Define Technical Requirements

Frontend Requirements

- I need to create a **user-friendly interface** to ensure customers' smooth browsing and purchasing experience.
- The website will be **responsive**, ensuring it works seamlessly on both **mobile** and **desktop** devices.
- These are the pages I will focus on:
    - **Home Page**: A showcase for featured products and promotions.
    - **Product Listing**: Displaying all available athletic footwear and apparel.
    - **Product Details**: Showing detailed information like price, stock, and product images.
    - **Cart**: Where users can review and modify their selected items.
    - **Checkout**: Handling secure payments and capturing shipping details.
    - **Order Confirmation**: Confirming successful purchases with order details.

Backend with Sanity CMS

- I will use **Sanity CMS** to **store and manage product data**, including names, prices, images, and stock levels.
- Sanity CMS will also **manage customer orders**, storing order history and customer information.
- I'll handle **checkout data**, such as shipping addresses and payment status, here.
- I need to ensure the **database schemas** align with the business's goals.

Why sanity is ideal for our project?

I chose **Sanity CMS** for my Nike Clone Marketplace project because it offers:

1. **Flexible Schema**: I can customize data structures for products, orders, and customers.
2. **Real-Time Collaboration**: Makes team collaboration easier.
3. **Powerful API**: Allows smooth data fetching between the backend and frontend.
4. **Scalability**: Handles growing content without performance issues.
5. **Easy Integration**: Works seamlessly with **Next.js**.
6. **User-Friendly Interface**: Simplifies content management for non-technical users.

These features make Sanity a great fit for a dynamic, scalable e-commerce platform.

Third-Party APIs

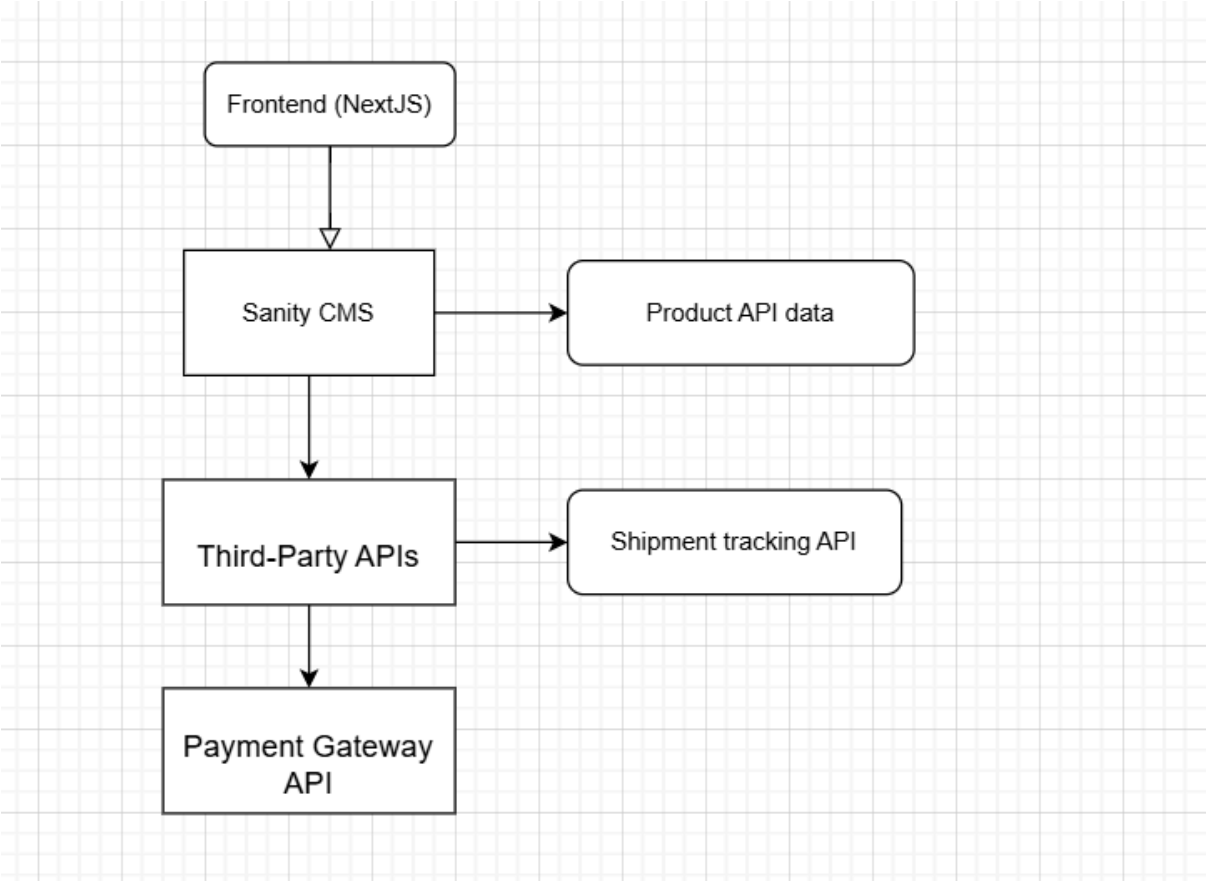
- **Shipment Tracking API**: To provide users with real-time updates on their order status.
- **Payment Gateway API**: To facilitate secure transactions with multiple payment methods.
- **Authentication API**: For handling user sign-ups and logins.
- **Analytics API**: To track user behavior and order trends for improved decision-making.

2. Design System Architecture

System Flow:

1. A **user visits the site** and browses the available products.
2. The **frontend (Next.js)** fetches the product data from **Sanity CMS**.
3. Once the user adds an item to the **cart**, I'll store the selection temporarily.
4. At **checkout**, the user's details and order information will be sent to **Sanity CMS**.
5. Payments will be processed via the **Payment Gateway API**.
6. After payment confirmation, an **order confirmation** will be displayed.
7. The **Shipment API** will fetch real-time delivery updates, which I will display to the user.

System Architecture Diagram[



Key Workflows

1. **User Registration**: Users sign up → Data stored in Sanity CMS → Confirmation sent.
2. **Product Browsing**: User views products → Sanity API fetches data → Displayed on frontend.
3. **Order Placement**: User adds items to cart → Proceeds to checkout → Order stored in Sanity CMS.
4. **Shipment Tracking**: Order status updates via API → Displayed in user dashboard.

3. Plan API Requirements

Product API

- **Endpoint:** `/products`
- **Method:** `GET`
- **Description:** Fetches all available products from Sanity CMS.

Response Example:

```
 {
  "id": 1,
```

```
  "name": "Nike Air Max",
  "price": 150,
  "stock": 20,
  "image": "url-to-image"
}
```

-

Order API

- **Endpoint:** `/orders`
- **Method:** `POST`
- **Description:** Creates a new order in Sanity CMS.

Payload Example:

```
 {
  "customerId": 101,
  "products": [{ "id": 1, "quantity": 2 }],
  "paymentStatus": "Pending"
}
```

-

Payment API

- **Endpoint:** `/payments`
- **Method:** `POST`
- **Description:** Processes a payment.

Payload Example:

```
 {
  "orderId": 1001,
  "amount": 200,
  "status": "Paid"
}
```

-

Shipment API

- **Endpoint:** `/shipment`
- **Method:** `GET`
- **Description:** Tracks the order status.

Response Example:

```
 {
  "shipmentId": 555,
  "status": "Out for delivery"
```

}

4. Write Technical Documentation

System Architecture Overview

- **Frontend (Next.js)**: I'll handle user interactions and display.
- **Sanity CMS**: It will store product, order, and customer data.
- **Payment Gateway API**: I'll use this to handle secure financial transactions.
- **Shipment API**: This will help me track orders in real-time and update users on their status.

Key Workflows

1. **Product Browsing**: The frontend fetches product data from Sanity CMS and displays it.
2. **Order Placement**: Order details are stored in Sanity CMS, and payment is processed.
3. **Shipment Tracking**: The Shipment API provides real-time order status.

Sanity CMS Schema Example

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'stock', type: 'number', title: 'Stock Level' },
    { name: 'image', type: 'image', title: 'Product Image' }
  ]
};
```