# HACKATHON 3

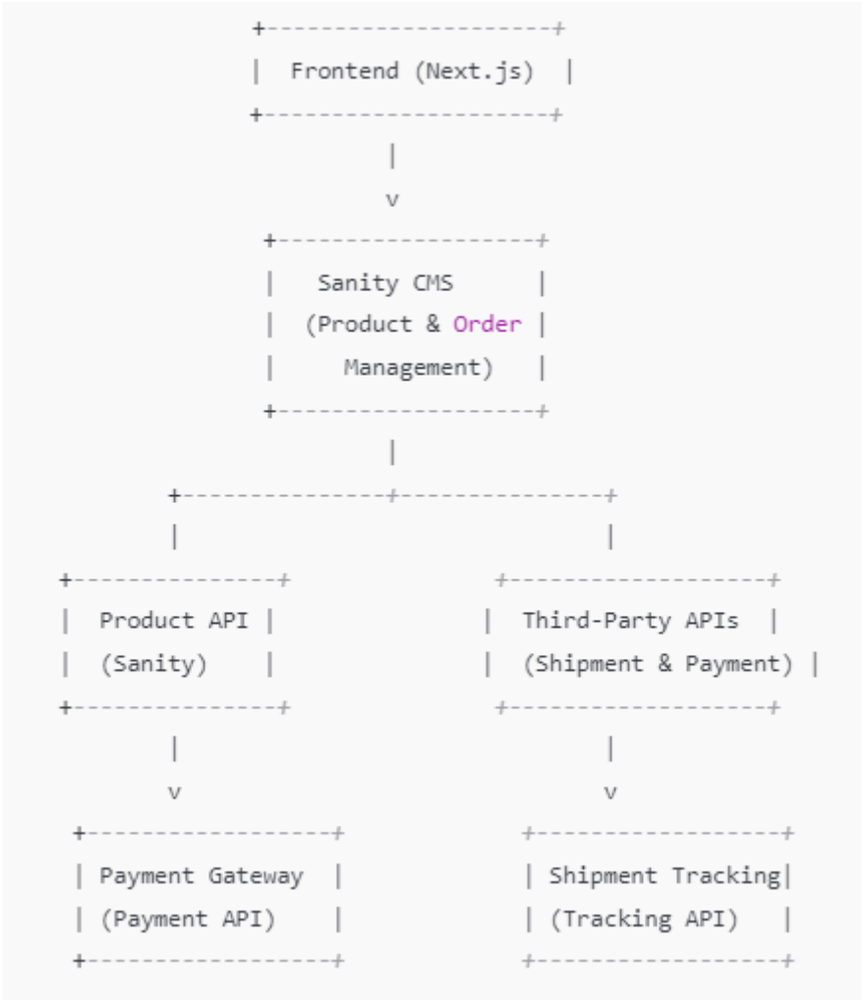
# DAY 2

# PLANNING THE TECHNICAL FOUNDATION


## Sharmeen Asif

## Roll number: 00200801


### System Architecture Diagram




## Explanation of System Components

### Frontend (Next.js)

- **Purpose**: Acts as the user interface where users browse products, manage their cart, and complete checkout.
- **Responsibilities**:
  - Fetch product listings from **Sanity CMS** via an API.
  - Send order details to **Sanity CMS**.
  - Integrate with **Payment Gateway API** to process payments.

- Request shipment status from **Shipment Tracking API**.

## 2. Sanity CMS

- **Purpose**: Manages and stores product data, customer information, and order records.
- **Responsibilities**:
  - Serve as the central content management system for storing product details, stock, and pricing.
  - Handle order creation and management (stores customer details and order status).
  - Product API: Provides product data (ID, name, price, stock, image).
  - Order Management: Receives new order data and stores it.

## 3. Product API (Sanity CMS)

- **Purpose**: Allows the frontend to fetch product data dynamically.
- **Responsibilities**:
  - Provides product details such as name, description, price, stock, and images to the frontend when requested via a GET request to /products.

## 4. Third-Party APIs

- These are external services integrated to manage non-internal functionalities, such as shipment tracking and payment processing.

### a. Shipment Tracking API

- **Purpose**: Provides real-time shipment tracking for customer orders.
- **Responsibilities**:
  - Fetch current order status, tracking number, expected delivery date, and estimated arrival times.

### b. Payment Gateway API

- **Purpose**: Processes payment transactions securely.
- **Responsibilities**:
  - Handle customer payment details and process transactions.
  - Provide payment status (success or failure) to the frontend.

## 5. Payment Gateway

- **Purpose**: Facilitates payment transactions during checkout.
- **Responsibilities**:
  - Works with **Payment Gateway API** to securely handle customer payment data and authorize payments.
  - Sends a success or failure response back to the frontend to update the order status.

## 6. Shipment Tracking

- **Purpose**: Provides real-time order delivery information.
- **Responsibilities**:
  - Fetches the shipment status and estimated delivery time from the **Shipment Tracking API**.
  - Displays real-time updates to users about their orders.

---

## Data Flow Overview

1. **User Interaction**:
   - A user browses the product catalog in the frontend (Next.js), which sends requests to the **Product API** (Sanity) to fetch and display product data.
2. **Order Placement**:
   - The user adds products to their cart and proceeds to checkout. The frontend sends a **POST request** to **Sanity CMS** to create a new order.
3. **Payment Processing**:
   - Once the order is confirmed, the frontend interacts with the **Payment Gateway API** to process the transaction. The payment status (success or failure) is received and communicated back to the user.
4. **Shipment Tracking**:
   - After payment is processed, the frontend makes a request to the **Shipment Tracking API** to retrieve real-time delivery updates for the order.

This system architecture clearly outlines how components interact, providing a clear understanding of the data flow and integration points between **Frontend (Next.js)**, **Sanity CMS**, and **Third-Party APIs** (Shipment and Payment).

# Workflow for your Next.js marketplace project based on the provided example:

### 1. User Registration

- **Step 1**: User signs up on the frontend.
- **Step 2**: User data (name, email, etc.) is sent to **Sanity CMS** to store the user details.
- **Step 3**: Sanity CMS stores user data in the database.
- **Step 4**: A confirmation email or message is sent back to the user confirming successful registration.

---

### 2. Product Browsing

- **Step 1**: User navigates to the product catalog on the frontend (Next.js).
- **Step 2**: Frontend sends a GET request to the **Sanity API** to fetch all product listings, including product details (name, price, stock, image).
- **Step 3**: Sanity CMS returns product data to the frontend.
- **Step 4**: Product data is displayed on the frontend for the user to browse.
- **Step 5**: User selects a product to view more details.

---

### 3. Order Placement

- **Step 1**: User adds items to the cart.
- **Step 2**: User proceeds to checkout by clicking on the "Checkout" button.
- **Step 3**: Frontend collects order details (product ID, quantity, customer info) and sends a **POST request** to **Sanity CMS** to create an order record.
- **Step 4**: **Sanity CMS** processes the order, stores the order details (including user info, items, and order status).

- **Step 5**: Frontend redirects user to the payment page.

---

## 4. Payment Processing

- **Step 1**: User enters payment details on the frontend (payment gateway integration).
- **Step 2**: Frontend sends payment details to the **Payment Gateway API** for processing.
- **Step 3**: Payment Gateway API processes the payment securely and returns the payment status (success or failure).
- **Step 4**: Based on the payment response, the frontend updates the order status (order completed or failed).
- **Step 5**: If payment is successful, **Sanity CMS** updates the order status to "paid" and sends a confirmation email to the user.

## 5. Shipment Tracking

- **Step 1**: Once the order is paid, frontend sends a request to the **Shipment Tracking API** to get the real-time shipment status.
- **Step 2**: **Shipment Tracking API** fetches shipment data and returns order status (e.g., "In Transit", "Out for Delivery").
- **Step 3**: Shipment status (including ETA and current location) is displayed on the frontend to the user.
- **Step 4**: The user can track the order until it is delivered.

---

## 6. Order Confirmation

- **Step 1**: After shipment tracking information is fetched, frontend displays a summary of the order with delivery details.
- **Step 2**: User receives an email confirming the successful order placement, payment, and estimated delivery date.

---

This workflow represents the end-to-end process in your marketplace, ensuring that data flows smoothly from the user's actions (browsing products, placing orders, and tracking shipments) to the backend (Sanity CMS, APIs).

# API requirements plan for your Next.js marketplace project, based on your current Sanity CMS schema:

### 1. Endpoint: /products

- **Method**: GET
- **Description**: Fetch all available products from Sanity.
- **Response Example**:

```
[
 {
  "id": 1,
  "name": "Product A",
  "price": 100,
  "stock": 50,
  "image": "https://image.url/product-a.jpg",
  "description": "A description of Product A"
 },
 {
  "id": 2,
  "name": "Product B",
  "price": 75,
  "stock": 30,
  "image": "https://image.url/product-b.jpg",
  "description": "A description of Product B"
 }
]
```


---

## 2. Endpoint: /orders

- **Method**: POST
- **Description**: Create a new order in Sanity.
- **Payload Example**:


```
{
 "customerInfo": {
  "name": "John Doe",
  "email": "johndoe@example.com",
  "address": "123 Main St, City, Country"
 },
 "products": [
  {
   "productId": 1,
   "quantity": 2
  },
  {
   "productId": 2,
   "quantity": 1
  }
 ],
 "paymentStatus": "paid"
}
```


- **Response Example**:


```
{
 "orderId": 12345,
 "status": "created",
 "customerInfo": {
  "name": "John Doe",
  "email": "johndoe@example.com",
  "address": "123 Main St, City, Country"
 },
 "products": [
  {
   "productId": 1,
   "name": "Product A",
   "price": 100,
   "quantity": 2
  },
  {
   "productId": 2,
   "name": "Product B",
   "price": 75,
```

```
      "quantity": 1
    }
  ],
  "totalAmount": 275
}
```


---

## 3. Endpoint: /shipment

- **Method**: GET
- **Description**: Track the shipment status of an order via a third-party API.
- **Response Example**:


```
{
  "shipmentId": "SH123456789",
  "orderId": 12345,
  "status": "In Transit",
  "expectedDeliveryDate": "2025-02-01",
  "trackingUrl": "https://tracking.url/sh123456789"
}
```


## 4. Endpoint: /register

- **Method**: POST
- **Description**: Register a new user.
- **Payload Example**:


```
{
  "name": "John Doe",
  "email": "johndoe@example.com",
  "password": "securepassword123"
}
```


- **Response Example**:


```
{
  "userId": "12345",
  "status": "registered",
  "message": "Registration successful, please verify your email."
}
```


---

## 5. Endpoint: /login

- **Method**: POST
- **Description**: User login.
- **Payload Example**:


```
{
  "email": "johndoe@example.com",
  "password": "securepassword123"
}
```


- **Response Example**:


```
{
  "userId": "12345",
  "status": "authenticated",
  "token": "JWT_TOKEN"
}
```

---

## 6. Endpoint: /payment

- **Method**: POST
- **Description**: Process payment for an order.
- **Payload Example**:

```
{
  "orderId": 12345,
  "paymentMethod": "credit_card",
  "paymentDetails": {
    "cardNumber": "4111111111111111",
    "expiryDate": "12/25",
    "cvv": "123"
  }
}
```

- **Response Example**:

```
{
  "orderId": 12345,
  "status": "payment_successful",
  "transactionId": "TX123456789"
}
```

---

## 7. Endpoint: /products/:id

- **Method**: GET
- **Description**: Fetch details of a single product by its ID.
- **Response Example**:

```
{
  "id": 1,
  "name": "Product A",
  "price": 100,
  "stock": 50,
  "image": "https://image.url/product-a.jpg",
  "description": "A detailed description of Product A"
}
```

---

## 8. Endpoint: /cart

- **Method**: POST
- **Description**: Add products to the shopping cart.
- **Payload Example**:

```
{
  "userId": "12345",
  "items": [
    {
      "productId": 1,
      "quantity": 2
    }
  ]
}
```

- **Response Example**:

```
{
  "cartId": "67890",
  "items": [
    {
      "productId": 1,
      "name": "Product A",
      "quantity": 2,
      "price": 100
    }
  ]
}
```

## 9. Endpoint: /cart/:id

- **Method**: GET
- **Description**: Get the user's cart details.
- **Response Example**:

```
{
  "cartId": "67890",
  "userId": "12345",
  "items": [
    {
      "productId": 1,
      "name": "Product A",
      "quantity": 2,
      "price": 100
    }
  ],
  "totalAmount": 200
}
```

## 10. Endpoint: /checkout

- **Method**: POST
- **Description**: Finalize the checkout and place an order.
- **Payload Example**:

```
{
  "userId": "12345",
  "cartId": "67890",
  "shippingAddress": "123 Main St, City, Country",
  "paymentStatus": "paid"
}
```

- **Response Example**:

```
{
  "orderId": 12345,
  "status": "created",
  "message": "Order placed successfully."
}
```

---

These API endpoints should handle the major functionalities for  marketplace, including product browsing, order placement, user management, payment processing, and shipment tracking. The responses ensure the data required for each workflow is returned in a clear and consistent format.

# NEXT.JS Marketplace Project Map