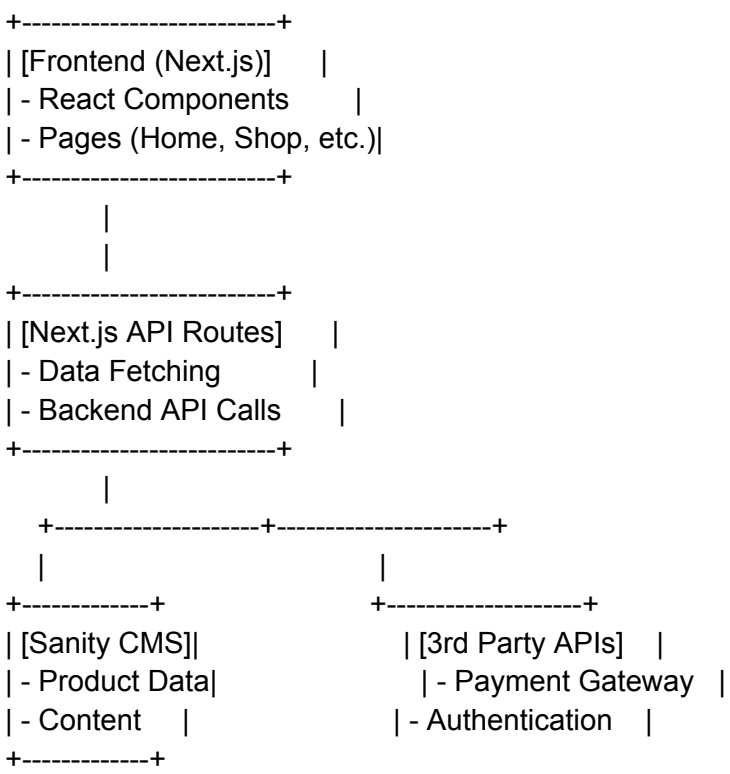
Hackathon Day 2

Planning the Technical Foundation

Design System Architecture



Frontend (Next.js):

- React components and pages for the UI (e.g., Home, Shop).
- Handles user interaction and displays dynamic content.

Next.js API Routes:

- Server-side logic for dynamic data fetching (e.g., product details).
- Handles API calls to fetch data from **Sanity CMS** or **3rd Party APIs**.

Sanity CMS:

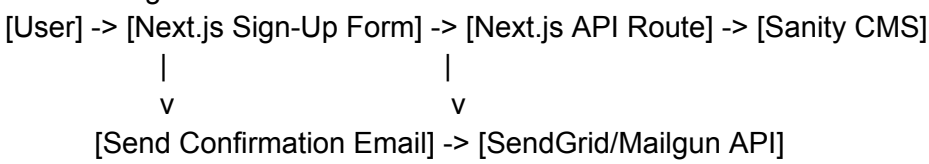
- Manages dynamic content like product data, categories, etc.
- Provides a headless CMS to manage and fetch content easily.

3rd Party APIs:

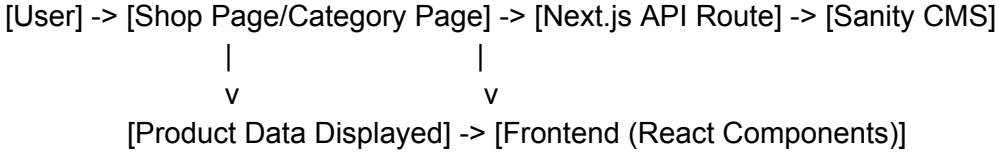
- Integrates with external services like **payment gateways** (e.g., Stripe), **authentication** (e.g., Auth0), and **analytics** (e.g., Google Analytics).

Workflows

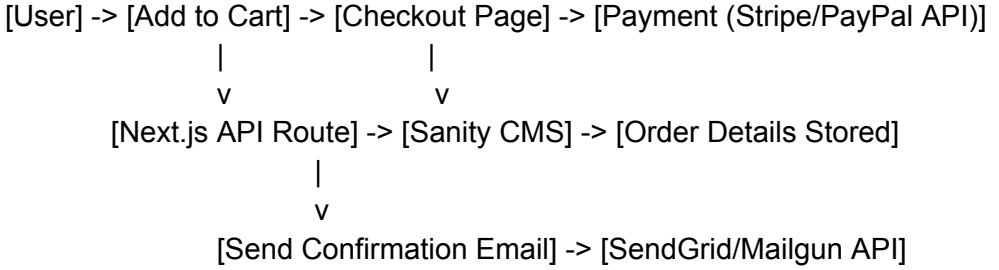
1. User Registration:



2. Product Browsing:



3. Order Placement:



4. Shipment Tracking:
[User] -> [Order Status Page] -> [Next.js API Route] -> [3rd Party Shipment API (Shippo/EasyPost)]
|
v
[Shipment Tracking Details Displayed]

1. User Registration Workflow:

Steps:

1. **User Sign-Up**:
    - User navigates to the **Signup Page**.
    - Enters personal details (e.g., name, email, password).
    - User submits the form.
2. **Data Storage**:
    - The user data is sent to the **Next.js API Routes**.
    - **Next.js API** processes the data and saves it in **Sanity CMS** (or any backend system you're using for user management).
3. **User Confirmation**:
    - After successful registration, a confirmation email is triggered via a third-party API (e.g., **SendGrid** or **Mailgun**).
    - The user receives a confirmation email with account activation or welcome details

2. Product Browsing Workflow:

Steps:

1. **User Views Product Categories**:
    - The user navigates to the **Product Categories Page** or **Shop Page** on the frontend.
    - The page triggers an API call to **Next.js API Routes**.
2. **Data Fetching from Sanity CMS**:
    - **Next.js API** queries **Sanity CMS** for product data (categories, names, prices, images).
3. **Products Displayed**:
    - The product data is sent to the **frontend** (React components) and displayed to the user on the **Shop Page** or **Category Page**.

3. Order Placement Workflow:

Steps:

1. **User Adds Items to Cart**:
    - The user selects products and adds them to the **Shopping Cart** on the frontend.
2. **Proceed to Checkout**:
    - The user clicks on **Checkout** and enters shipping and payment details.
    - Payment is processed via **3rd Party API** (e.g., **Stripe** or **PayPal**).
3. **Order Saved in Sanity**:
    - After successful payment, the order details (items, total price, user information, shipping address) are stored in **Sanity CMS**.
4. **Order Confirmation**:
    - A confirmation email with order details is sent to the user via a **3rd Party Email API** (e.g., **SendGrid**).

4. Shipment Tracking Workflow:

Steps:

1. **User Views Order Status**:
    - The user navigates to the **Order Status Page** to check the shipment details.
2. **Fetch Shipment Status**:
    - The order ID is sent to a **3rd Party API** (e.g., **Shippo**, **EasyPost**) to get the latest shipping status.
3. **Display Shipment Status**:
    - The shipment tracking details (e.g., dispatched, in transit, delivered) are fetched and displayed on the **Order Status Page** on the frontend.

Plan API Requirements

---

1. Product Management Schema

```
{
  "product": {
    "id": "string",
    "title": "string",
    "description": "string",
    "price": "number",
    "oldPrice": "number",
```

```
    "image": "string",
    "category": "string",
    "stock": "number",
    "colors": ["string"],
    "department": "string",
    "rating": "number",
    "reviews": [
      {
        "userId": "string",
        "rating": "number",
        "comment": "string",
        "date": "string"
      }
    ]
  }
}
```

---

2. Inventory Management Schema

```
{
  "inventory": {
    "productId": "string",
    "quantityInStock": "number",
    "quantitySold": "number",
    "quantityReserved": "number"
    "lastUpdated": "string"
  }
}
```

---

3. Payment Integration Schema

```
{
  "payment": {
    "orderId": "string",     "userId": "string",
    "paymentMethod": "string",
    "paymentStatus": "string",
    "paymentDate": "string",
```

```
    "amountPaid": "number",
    "transactionId": "string"
  }
}
```

---

4. Shipment Tracking Schema

```
{
  "shipment": {
    "orderId": "string",
    "shipmentId": "string",
    "carrier": "string",
    "status": "string",
    "estimatedDelivery": "string"
    "actualDelivery": "string",
    "trackingUrl": "string"
  }
}
```