DAY 2:PLANNING THE TECHNICAL FOUNDATION

TECHNICAL REQUIREMENTS

Frontend Requirements

- User Interface (UI): A clean and interactive design for product browsing, cart, and checkout pages.
- Responsive Design: Ensure the platform is mobile-friendly, adaptable to different screen sizes.
- Key Pages:
    - Homepage
    - Product Listing
    - Product Details
    - Cart
    - Checkout
    - Order Confirmation

Backend with Sanity CMS:

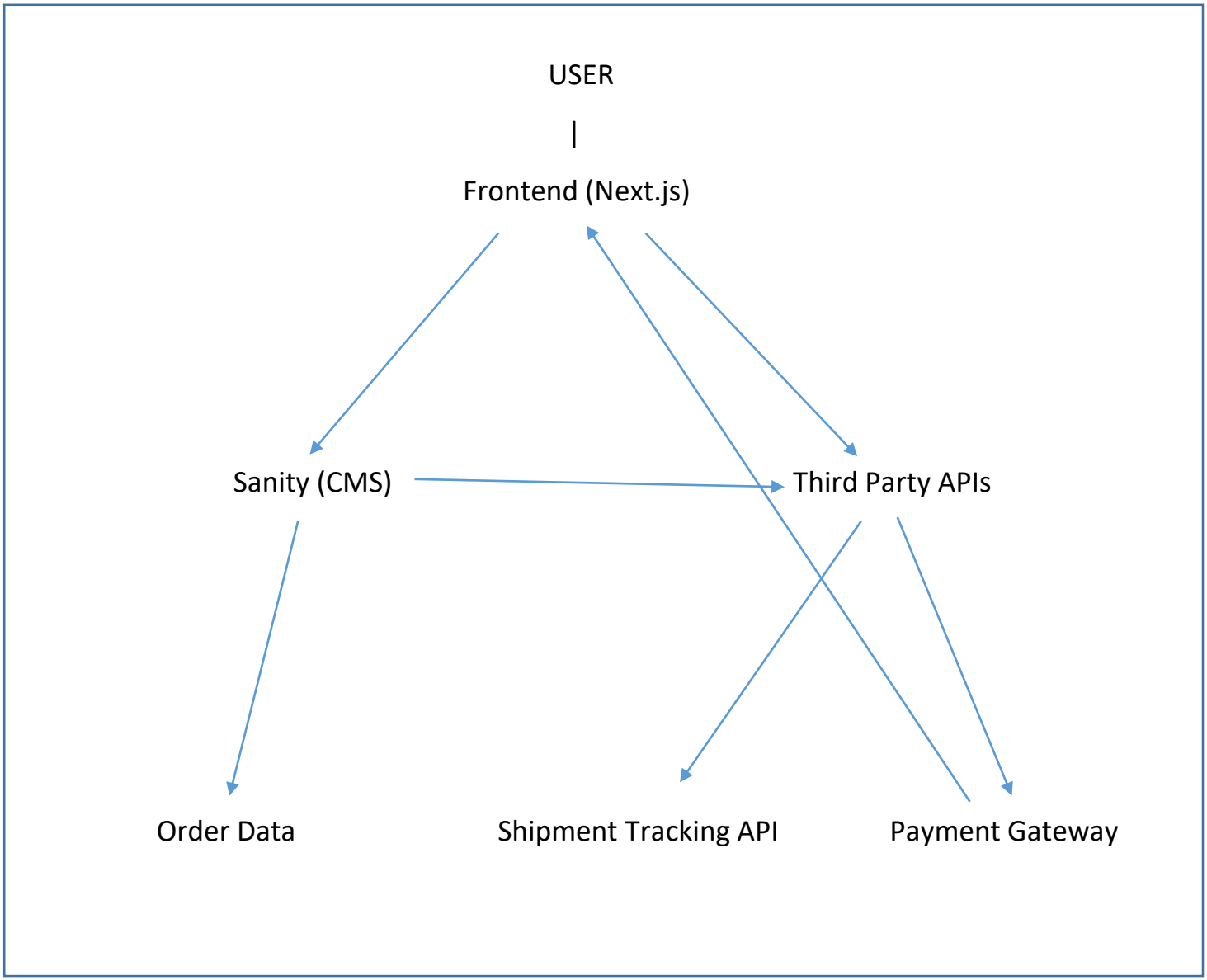
- Use Sanity CMS to manage product data, user profiles, and order records.
- Data Schema: Design schemas for entities such as Products, Customers, and Orders.

Third Party API Integrations:

- Shipment Tracking API: Real-time tracking of products.

- Payment Gateway API: Secure payment handling.
- Stock Management API (optional): Real-time stock management for inventory updates.

SYSTEM ARCHITECTURE



User: Performs actions like browsing products or placing an order.

Frontend (Next.js): Sends user actions to the backend and third-party APIs.

Sanity CMS: Manages and stores all order and product data.

Third Party APIs: Handle shipment tracking and payment processing.

Payment Gateway: Processes payments and sends confirmation back.

Order Data: Stored in Sanity CMS for backend reference.

KEY WORKFLOWS

1. **User Registration:**
    - User signs up -> Data stored in **Sanity CMS** -> Confirmation sent to user.
2. **Product Browsing:**
    - User select product category -> Data fetched from **Sanity CMS API** -> Products displayed dynamically on the frontend.
3. **Order Placement:**
    - User adds items to cart -> Checkout process -> Order details stored in **Sanity CMS.**
4. **Shipment Tracking:**
    - Order status updates -> Data fetched via **Third-Party Shipment API** -> User sees real-time tracking.

PLAN API REQUIREMENTS

1. Product Management:

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

```
{
 "id": 1,
 "name": "Product A",
 "price": 100,
 "stock": 10
}
```

2. Order Management:

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

```
{
 "customerInfo": {
  "name": "John Doe",
  "email": "john@gmail.com"
 },
 "productDetails": [
  {
   "productId": 1,
   "quantity": 2
  }
 ],
 "paymentStatus": "Success"
}
```

- **Response Example:**

```
{
  "orderId": 456,
  "status": "Created",
  "message": "Order confirmed"
}
```

3. Shipment Tracking:

- **Endpoint:** /shipment
- **Method:** GET
- **Description:** Track order status via third-party API.
- **Response Example:**

```
{
  "shipmentId": 123,
  "orderId": 456,
  "status": "In Transit",
  "expectedDeliveryDate": "2025-01-21"
}
```

---

4. Sanity Schema Definitions

□ Product Schema:

- Used for managing product details (name, price, stock, image).

□ Customer Schema:

- Used for storing customer details (name, email, address).

□ Order Schema:

- Stores order-related data, such as product details, customer information, and payment status.

5. Technical Roadmap

Here is a high-level roadmap for the project:

1. **Step 1: Setup Sanity CMS and Design Schemas**
    - Set up Sanity CMS and design schemas for Products, Customers, and Orders.
2. **Step 2: Frontend Development**
    - Develop the frontend in Next.js, implementing features such as product browsing and checkout.
3. **Step 3: API Integration**
    - Integrate third-party APIs for payment and shipment tracking.
4. **Step 4: Testing & Optimization**
    - Test all components, ensuring everything works together smoothly.