

# Day 2: Transitioning to Technical Planning

## 1. Technical Requirements

### Key Components:

#### 1. Frontend (Next.js)

**Role:** Delivers a user-friendly interface for browsing products, placing orders, and tracking shipments.

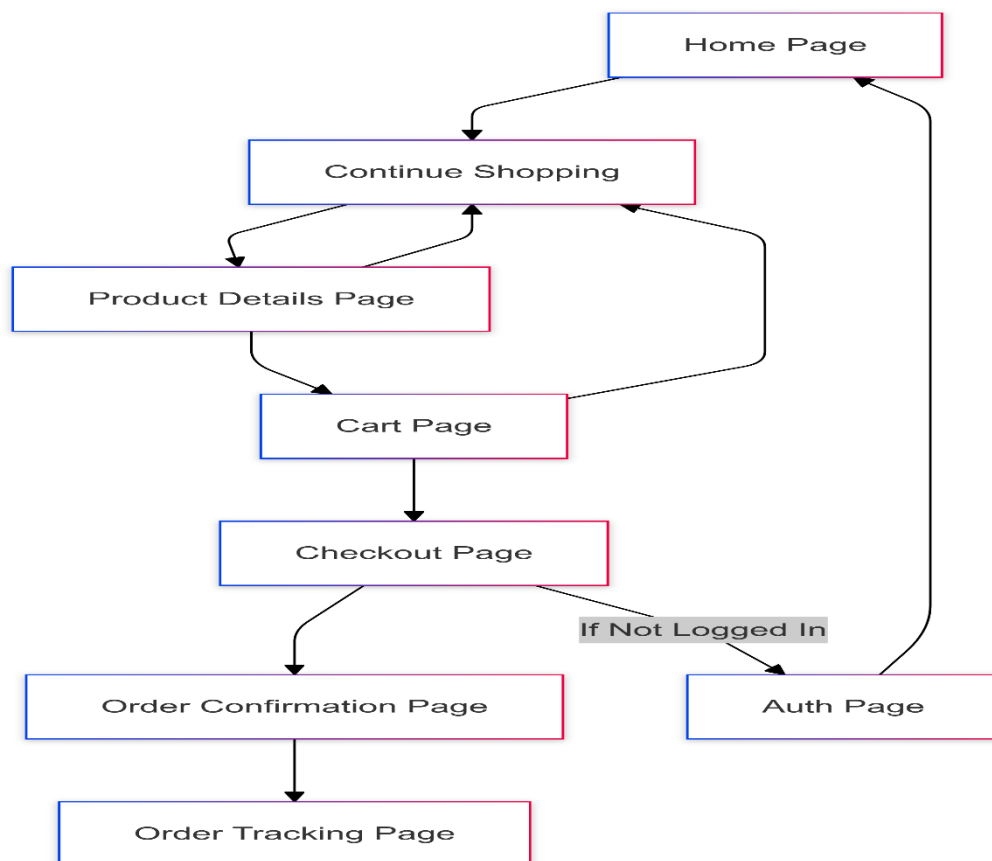
**Interactions:**

- **User Authentication:** Sends user actions to backend (Sanity CMS).
- **Data Requests:** Requests product data and dynamically displays it.
- **Order Processing:** Places an order and processes payment for it.

**Requirements:**

- **Responsive Design:** Optimized for mobile and desktop.
- **UI Framework:** Uses *Shadcn* for consistent, reusable, and aesthetically pleasing components such as buttons, modals, forms, and more.
- **Essential Pages:** Home, Product Listing, Product Details, Cart, Auth, Checkout, Order Confirmation.

**FLOW:**



#### 2. Sanity CMS

**Role:** Acts as content management and central data hub for platform.

**Responsibilities:**

- Fetch and store product data, customer data, order data.

- Maintains order and customer information.
- Integrates different external APIs for shipment tracking and payments.

### 3. Third-Party APIs

**Role:** Supports shipment tracking and payment processing.

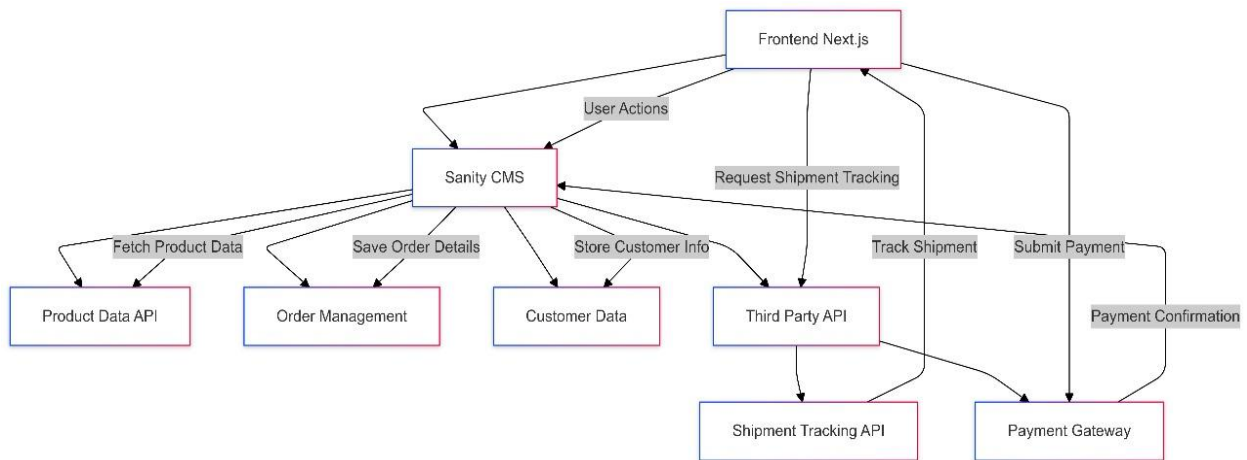
**Integration:** Connects to APIs for real-time updates and secure payments.

**Support:** Enhances backend capabilities with additional services.

----- X ----- X ----- X -----

## 2. System Architecture

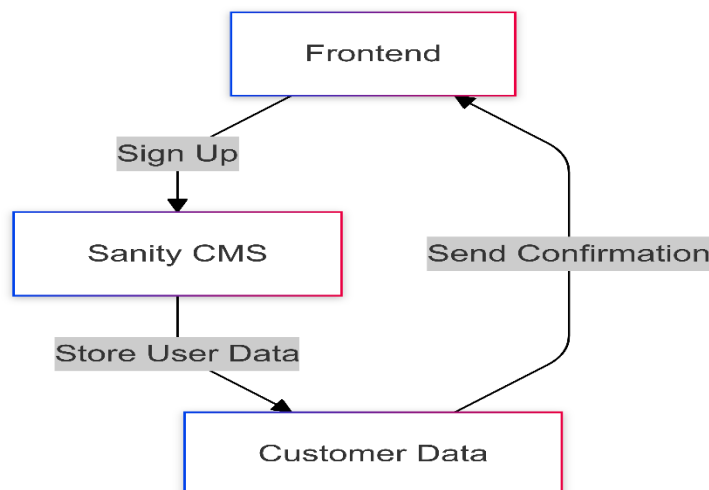
This is how the data will flow through the process:



### Key Workflows to Include:

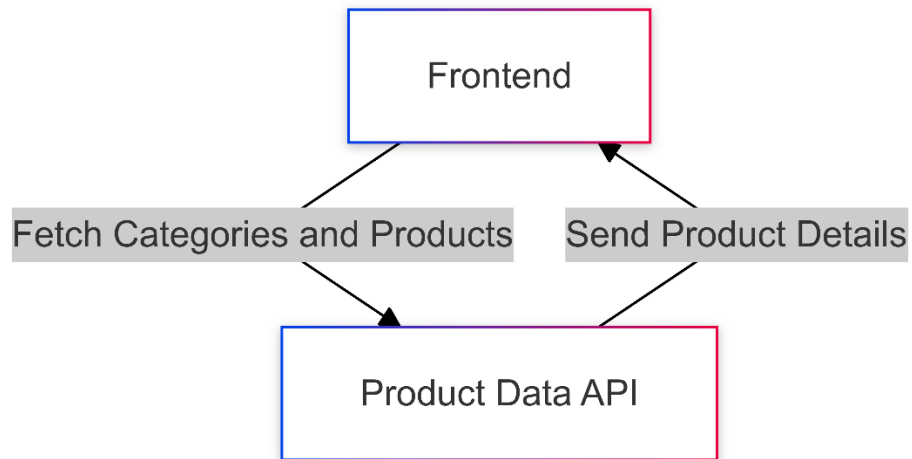
#### 1. User Registration:

- User registration details are sent from the frontend to Sanity CMS, which stores the data and sends a confirmation to the user.



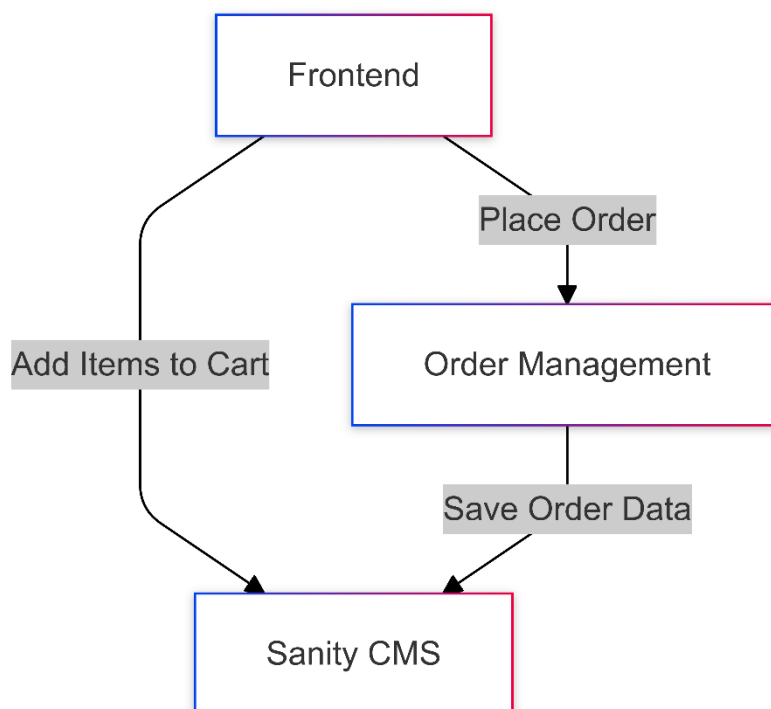
## 2. Product Visit:

- The product fetching request is made by the frontend along the Product Data API path, which passes through Sanity CMS.



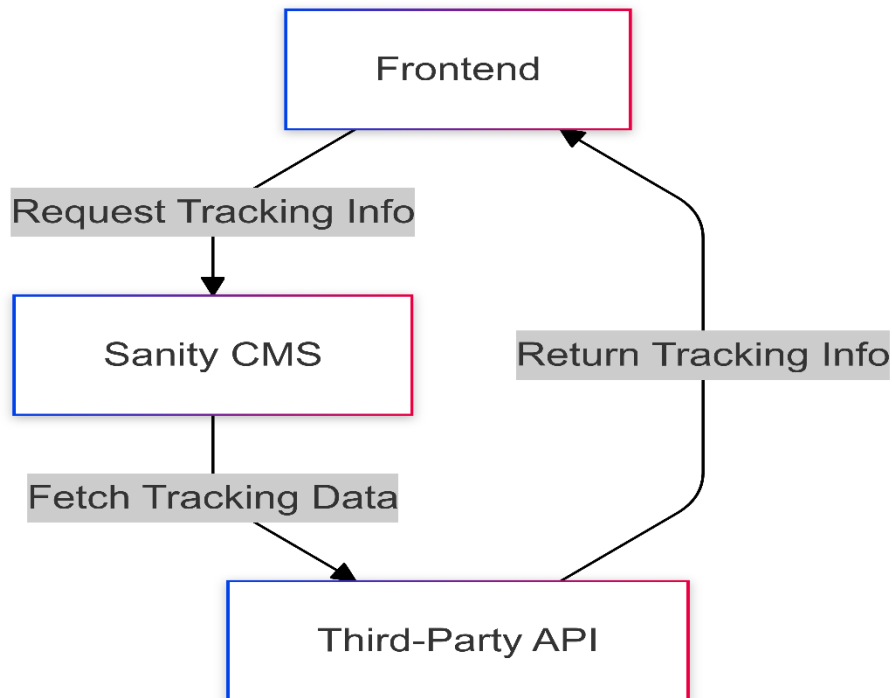
## 3. Order Placement:

- The customer adds all products to the cart and places the orders, which will go to Sanity CMS for storage purposes.



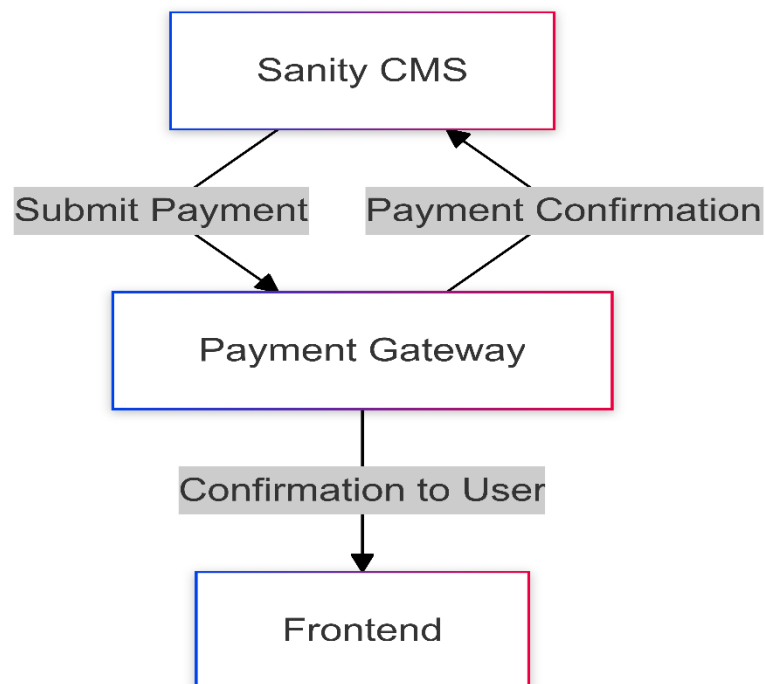
#### 4. Tracking:

- The system calls a third-party API for shipment updates and displays it to the customer.



#### 5. Payments:

- Payments are processed through the Payment Gateway and then confirmed to both the customer and Sanity CMS.



### 3. API Requirements

This document provides details about the API endpoints available in the Sanity CMS system. Each endpoint includes information about its method, purpose, and a response example.

Endpoint	Method	Purpose	Response Example
/products	GET	Fetch all available product details from Sanity CMS.	{ "id": 1, "name": "Product A", "price": 100, "stock": 50, "image": "productA.jpg" }
/products/:id	GET	Fetch details of a specific product by ID.	{ "id": 1, "name": "Product A", "description": "High-quality running shoes", "price": 100, "stock": 50, "images": [ "image1.jpg", "image2.jpg" ] }
/orders	POST	Create a new order in Sanity CMS.	{ "orderId": 456, "status": "Order Placed" }
/orders/:id	GET	Fetch details of a specific order by ID.	{ "orderId": 456, "customerId": 123, "products": [ { "productId": 1, "quantity": 2 }, { "productId": 2, "quantity": 1 } ], "totalAmount": 250, "status": "Shipped" }
/customers/:id	GET	Fetch customer details by ID.	{ "customerId": 123, "name": "John Doe", "email": "john.doe@example.com", "orderHistory": [ 456, 789 ] }
/shipment/:orderId	GET	Track the status of a specific order shipment.	{ "shipmentId": 789, "orderId": 456, "status": "In Transit", "expectedDeliveryDate": "2025-01-20" }
/payments	POST	Process payment for an order.	{ "paymentId": 789, "status": "Completed" }

### 4. Technical Documentation

#### 1. System Architecture

The purpose of the system architecture document is to show the design of the system and the interaction of the components within the marketplace system. It also explains how the frontend application, Sanity CMS, and different APIs work together to facilitate a good experience for the user.

##### Key Components:

- Frontend (Next.js): Presents an easy to use interface for product search, ordering, and tracking of the orders.
- Sanity CMS: Serves as a single content management platform where product, customer and order information is stored and organized.

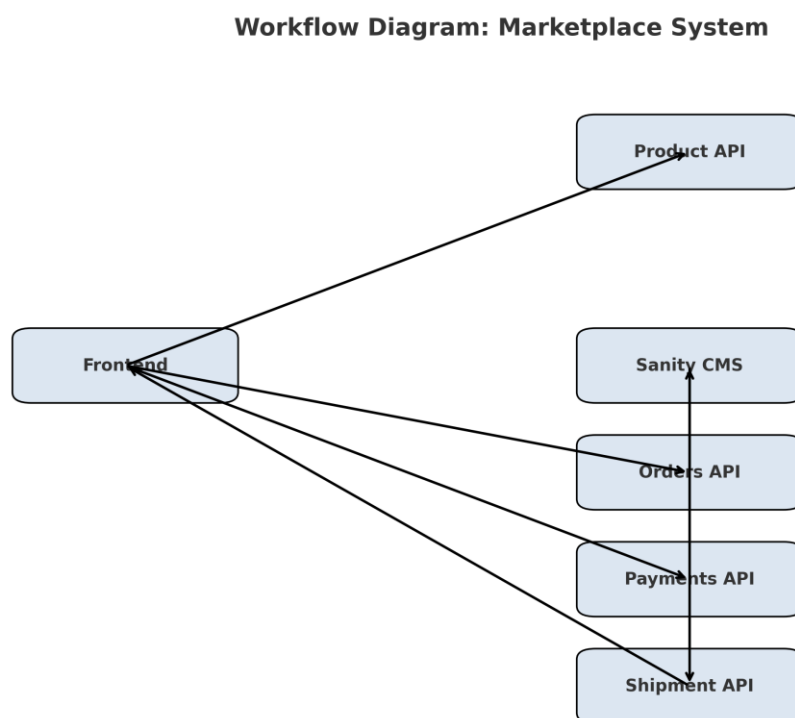
- Third-Party APIs: Provides different services like shipment tracking and payment services and other integrated services.

### **Workflow Highlights:**

- Requests from a user are collected at the frontend and forwarded to the relevant APIs.
- Sanity CMS is effective in performing as a data warehouse and bridges the frontend with the external services.
- Notifications and payments are a guarantee of a good experience for the user.

## **2. Workflow Diagram**

The diagram below illustrates the workflows in the marketplace system, including interactions between the frontend, Sanity CMS, and third-party APIs.



### **Key Workflows:**

1. **User Registration:** Details sent from the frontend to Sanity CMS, which stores data and confirms registration.
2. **Product Browsing:** Frontend requests product data from the Product API via Sanity CMS and displays it dynamically.
3. **Order Placement:** Customers place orders via the Orders API, with data stored in Sanity CMS.
4. **Payments:** Payment information is processed through the Payments API and confirmed to the customer and CMS.
5. **Shipment Tracking:** Shipment updates fetched via the Shipment API are displayed to the user.

### 3. Schema Design:

#### **Products**

```
export const products = {  
  name: "products",  
  type: "document",  
  title: "Products",  
  fields: [  
    { name: "productId", type: "number", title: "Product ID" },  
    { name: "name", type: "string", title: "Name" },  
    { name: "price", type: "number", title: "Price", options: { precision: 2 } },  
    { name: "stock", type: "number", title: "Stock" },  
    { name: "category", type: "string", title: "Category" },  
    { name: "tags", type: "array", title: "Tags", of: [{ type: "string" }] },  
    { name: "description", type: "text", title: "Description" },  
    { name: "images", type: "array", title: "Images", of: [{ type: "image" }] },  
    { name: "brand", type: "string", title: "Brand" },  
    { name: "size", type: "array", title: "Size", of: [{ type: "string" }] },  
    { name: "color", type: "array", title: "Color", of: [{ type: "string" }] },  
  ],  
};
```

#### **Orders**

```
export const orders = {  
  name: "orders",  
  type: "document",  
  title: "Orders",  
  fields: [  
    { name: "orderId", type: "number", title: "Order ID" },  
    { name: "customerId", type: "number", title: "Customer ID" },  
    { name: "status", type: "string", title: "Status" },  
    { name: "timestamp", type: "datetime", title: "Timestamp" },  
    { name: "totalAmount", type: "number", title: "Total Amount", options: { precision: 2 } },  
  ],  
};
```

```
};
```

## ***Order Products***

```
export const orderProducts = {  
  name: "order_products",  
  type: "document",  
  title: "Order Products",  
  fields: [  
    { name: "orderId", type: "number", title: "Order ID" },  
    { name: "productId", type: "number", title: "Product ID" },  
    { name: "quantity", type: "number", title: "Quantity" },  
  ],  
};
```

## ***Customers***

```
export const customers = {  
  name: "customers",  
  type: "document",  
  title: "Customers",  
  fields: [  
    { name: "customerId", type: "number", title: "Customer ID" },  
    { name: "name", type: "string", title: "Name" },  
    { name: "contactInfo", type: "string", title: "Contact Info" },  
    { name: "address", type: "text", title: "Address" },  
  ],  
};
```

## ***Shipments***

```
export const shipments = {  
  name: "shipments",  
  type: "document",  
  title: "Shipments",  
  fields: [  
    { name: "shipmentId", type: "number", title: "Shipment ID" },  
    { name: "orderId", type: "number", title: "Order ID" },  
  ],  
};
```



```
    { name: "status", type: "string", title: "Status" },
    { name: "deliveryDate", type: "datetime", title: "Delivery Date" },
    { name: "trackingNumber", type: "string", title: "Tracking Number" },
  ],
};
```

















## ***Payments***

```
export const payments = {
  name: "payments",
  type: "document",
  title: "Payments",
  fields: [
    { name: "paymentId", type: "number", title: "Payment ID" },
    { name: "orderId", type: "number", title: "Order ID" },
    { name: "amount", type: "number", title: "Amount", options: { precision: 2 } },
    { name: "status", type: "string", title: "Status" },
    { name: "paymentMethod", type: "string", title: "Payment Method" },
  ],
};
```

## ***Delivery Zones***

```
export const deliveryZones = {
  name: "delivery_zones",
  type: "document",
  title: "Delivery Zones",
  fields: [
    { name: "zoneId", type: "number", title: "Zone ID" },
    { name: "zoneName", type: "string", title: "Zone Name" },
    { name: "coverageArea", type: "array", title: "Coverage Area", of: [{ type: "text" }] },
    { name: "assignedDrivers", type: "array", title: "Assigned Drivers", of: [{ type: "text" }] },
  ],
};
```

#### 4. *Technical Roadmap:*

-   **Define Technical Requirements**
-   **Design System Architecture**
-   **Develop Data Schemas**
-   **Build Frontend Interfaces**
-   **Integrate Sanity CMS**
-   **Connect Third-Party APIs**
-   **Test and Refine**
-   **Deploy and Launch**