# Define Technical Requirements:

## Frontend Requirements:

* **User-Friendly Interface**: A design that helps users easily browse products.
* **Responsive Design**: A platform that works well on both mobile and desktop.

### Essential Pages:

* Homepage
* Product Listing
* Product Details
* Cart
* Checkout
* Order Confirmation

## Backend Requirements (Sanity CMS):

* **Product Data Management**: Use Sanity CMS to manage products, customer details, and orders.
* **Schema Design**: Design schemas in Sanity that align with business goals.

### Third-Party APIs:

* **Shipment Tracking**: Integrate a third-party API for real-time order tracking.
* **Payment Gateway**: Integrate a payment gateway (e.g., Stripe, PayPal) for secure payment processing.

# Design System Architecture:

### Diagram:

[Frontend (Next.js)]  
|  
[Sanity CMS] ---------> [Product Data API]  
|  
[Third-Party API] -----> [Shipment Tracking API]  
|  
[Payment Gateway]

### Key Workflows:

**User Registration**:  
User signs up → Data is saved in Sanity → Confirmation is sent to the user.

**Product Browsing**:  
User views products → Sanity API fetches data → Products are displayed.

**Order Placement**:  
User adds items to the cart → Proceeds to checkout → Order is saved in Sanity.

**Shipment Tracking**:  
Order status is fetched via a third-party API → Displayed to the user.

# Plan API Requirements:

### API Endpoints:

| **Endpoint** | **Method** | **Description** | **Response Example** |
| --- | --- | --- | --- |
| /products | GET | Fetch all products | { "id": 1, "name": "Product A", "price": 100 } |
| /orders | POST | Create a new order | { "orderId": 123, "status": "Pending" } |
| /shipment | GET | Track order status | { "shipmentId": 456, "status": "In Transit" } |

# Write Technical Documentation:

### System Architecture Overview:

* **Frontend**: Next.js (User Interface)
* **Backend**: Sanity CMS (Product Data Management)
* **Third-Party APIs**: Shipment Tracking, Payment Gateway

### Key Workflows:

**User Registration**:  
User signs up → Data is saved in Sanity → Confirmation is sent to the user.

**Product Browsing**:  
User views products → Sanity API fetches data → Products are displayed.

**Order Placement**:  
User adds items to the cart → Proceeds to checkout → Order is saved in Sanity.

**Shipment Tracking**:  
Order status is fetched via a third-party API → Displayed to the user.

### API Endpoints:

| **Endpoint** | **Method** | **Description** | **Response Example** |
| --- | --- | --- | --- |
| /products | GET | Fetch all products | { "id": 1, "name": "Product A", "price": 100 } |
| /orders | POST | Create a new order | { "orderId": 123, "status": "Pending" } |
| /shipment | GET | Track order status | { "shipmentId": 456, "status": "In Transit" } |

### Sanity Schema Example:

#### 1. Product Schema:

javascript

CopyEdit

export default {

name: 'product',

type: 'document',

title: 'Product',

fields: [

{

name: 'name',

type: 'string',

title: 'Product Name',

},

{

name: 'price',

type: 'number',

title: 'Price',

},

{

name: 'stock',

type: 'number',

title: 'Stock Level',

},

{

name: 'category',

type: 'string',

title: 'Category',

options: {

list: [

{ title: 'Electronics', value: 'electronics' },

{ title: 'Clothing', value: 'clothing' },

{ title: 'Groceries', value: 'groceries' },

{ title: 'Home Appliances', value: 'home-appliances' },

],

},

},

{

name: 'tags',

type: 'array',

title: 'Tags',

of: [{ type: 'string' }],

options: {

layout: 'tags',

},

},

{

name: 'image',

type: 'image',

title: 'Product Image',

},

],

};

#### 2. Order Schema:

javascript

CopyEdit

export default {

name: 'order',

type: 'document',

title: 'Order',

fields: [

{

name: 'orderId',

type: 'string',

title: 'Order ID',

},

{

name: 'customer',

type: 'reference',

to: [{ type: 'customer' }],

title: 'Customer',

},

{

name: 'products',

type: 'array',

title: 'Products',

of: [

{

type: 'object',

fields: [

{

name: 'product',

type: 'reference',

to: [{ type: 'product' }],

title: 'Product',

},

{

name: 'quantity',

type: 'number',

title: 'Quantity',

},

],

},

],

},

{

name: 'status',

type: 'string',

title: 'Status',

options: {

list: [

{ title: 'Pending', value: 'pending' },

{ title: 'Shipped', value: 'shipped' },

{ title: 'Delivered', value: 'delivered' },

],

},

},

{

name: 'timestamp',

type: 'datetime',

title: 'Order Date and Time',

},

],

};

#### 3. Customer Schema:

javascript

CopyEdit

export default {

name: 'customer',

type: 'document',

title: 'Customer',

fields: [

{

name: 'customerId',

type: 'string',

title: 'Customer ID',

},

{

name: 'name',

type: 'string',

title: 'Name',

},

{

name: 'contactInfo',

type: 'object',

title: 'Contact Info',

fields: [

{

name: 'email',

type: 'string',

title: 'Email',

},

{

name: 'phone',

type: 'string',

title: 'Phone',

},

],

},

{

name: 'address',

type: 'string',

title: 'Delivery Address',

},

{

name: 'orderHistory',

type: 'array',

title: 'Order History',

of: [{ type: 'reference', to: [{ type: 'order' }] }],

},

],

};

#### 4. Shipment Schema:

javascript

CopyEdit

export default {

name: 'shipment',

type: 'document',

title: 'Shipment',

fields: [

{

name: 'shipmentId',

type: 'string',

title: 'Shipment ID',

},

{

name: 'order',

type: 'reference',

to: [{ type: 'order' }],

title: 'Order',

},

{

name: 'status',

type: 'string',

title: 'Status',

options: {

list: [

{ title: 'In Transit', value: 'in-transit' },

{ title: 'Delivered', value: 'delivered' },

],

},

},

{

name: 'deliveryDate',

type: 'datetime',

title: 'Delivery Date',

},

],

};

# Collaborate and Refine:

### Group Discussions:

Brainstorm with peers to improve plans.

### Peer Review:

Share technical plans for feedback.

### Version Control:

Use GitHub to track changes.

# Submission Guidelines:

### Repository Submission:

* Create a folder named "Documentation" in your repository.
* Upload your technical document, diagrams, and schemas.

### Document Structure:

* Follow the industry-standard format.
* Include collaboration notes if applicable.

### File Naming Convention:

* Use clear names, e.g., SystemArchitecture\_Day2.pdf, APIEndpoints.xlsx, SanitySchema.js.

# What’s Next?

**Day 3**: Use the provided API as a reference or create your own schema in Sanity for GET, POST, UPDATE, PATCH, and DELETE operations.

### Focus Areas:

* **Q-Commerce**: Real-time inventory updates and SLA tracking.
* **Rental eCommerce**: Rental duration, deposit handling, and condition tracking.
* **General eCommerce**: Product browsing, cart management, and order placement.

