Day 2 Activities: Transitioning to Technical Planning for My Q-Commerce Website

Technical Requirements

The first step is to translate my business goals into clear technical requirements for my Q-commerce website. For each feature identified on Day 1, I will outline the following:

Frontend Requirements:

- **User-Friendly Interface:**
    - I need a simple, intuitive interface that allows users to browse products and place orders quickly. The goal is to reduce any friction in the user experience and make the shopping process as smooth as possible.
    - Products should have clear images, brief descriptions, prices, and importantly, delivery time estimates to ensure users make fast decisions.
- **Responsive Design:**
    - The website must be responsive and load efficiently on both mobile and desktop devices. As most users will likely shop from their mobile phones, it's essential that the site performs well across all devices.
- **Essential Pages:**
    - **Home:** A page that showcases fast delivery times, featured products, and current offers.
    - **Product Listing:** A section where users can browse products with filters for categories, price, and delivery time.
    - **Product Details:** A page that includes clear descriptions, high-quality images, prices, and delivery time estimates.
    - **Cart:** A page displaying the items in the cart, estimated delivery times, and an easy way to proceed to checkout.
    - **Checkout:** A simplified, single-page checkout with payment options (credit/debit, wallets, COD) and quick delivery choices.
    - **Order Confirmation:** A page that confirms the order with real-time tracking options and expected delivery time.

Sanity CMS as Backend:

- **Product Data Management:**

    - I will use Sanity CMS to manage all product data, including details like product name, price, availability, delivery times, and images.

- **Customer and Order Data Management:**
    - Sanity will also store customer details, order history, and preferences, ensuring a smooth process for returning users.

- **Schema Design:**
    - **Product Schema:** I will design the schema with fields for product name, price, stock level, categories, delivery time, and images.
    - **Order Schema:** The schema will store order details, status, payment information, and timestamps.
    - **Customer Schema:** This schema will store basic customer details, such as name, email, phone, address, and order preferences.

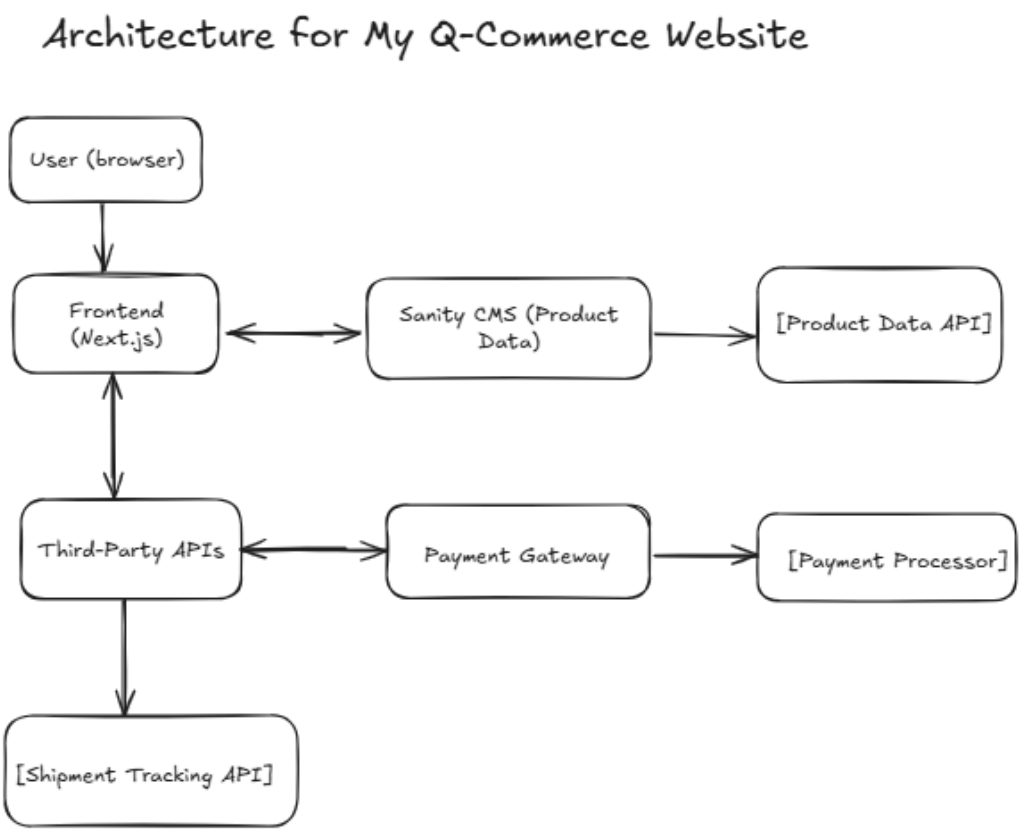
Third-Party APIs:

- **Shipment Tracking API:**
    - I will integrate shipment tracking APIs to provide real-time tracking of the orders for the customers.

- **Payment Gateway API:**
    - I'll use payment gateway APIs like Stripe or PayPal to securely process payments. Customers will have multiple payment options like credit/debit cards, wallets, and COD.

- # 2. Design System Architecture for My Q-Commerce Website

- components for my Q-commerce website will interact. The architecture includes how frontend, backend (Sanity CMS), third-party APIs, and payment gateways integrate together to create a seamless user experience.



-

Components Breakdown:

1. **Frontend (Next.js):**
    - **Role:** This is where users interact with the website. It's responsible for displaying product listings, allowing users to add products to their cart, and providing a seamless checkout experience.
    - **Functions:**
        - Dynamic product display
        - Cart management

        User checkout and payment form
2. **Sanity CMS (Backend):**

- **Role:** Sanity CMS is used to manage product data, customer information, and order records. It acts as the content management system and stores all product and order details.
- **Functions:**
  - Product listings and details
  - Customer profile data
  - Order history and tracking

3. **Product Data API:**
   - **Role:** This API fetches product information from **Sanity CMS** and serves it to the frontend, displaying product details to users dynamically.
   - **Functions:**
     - Provide product listings, pricing, images, descriptions
     - Handle queries related to stock availability and product variations
4. **Payment Gateway & Processor:**
   - **Role:** The payment gateway securely handles payment transactions, while the payment processor manages the financial processing.
   - **Functions:**
     - Accept and process credit card and digital wallet payments
     - Handle secure transactions via providers like PayPal, Stripe, etc.
     - Send payment success or failure status back to the frontend
5. **Third-Party APIs (Shipment Tracking & Other Services):**
   - **Role:** These external services help with additional functionalities like shipment tracking and other logistics needs.
   - **Functions:**
     - **Shipment Tracking API:** Fetches the latest updates on order shipments, such as delivery status or expected delivery time.
     - **Other APIs:** For additional services like tax calculations, currency exchange, or customer service chatbots.

How Components Interact:

1. **User Browsing:**
   - The user visits the frontend of the website, which displays product categories and individual product details. The **Frontend** makes API calls to **Sanity CMS** to fetch

product data, and the **Product Data API** returns the necessary information to display the products.

2. **Adding to Cart & Checkout:**
    - The user adds products to their cart and proceeds to checkout. The **Frontend** sends the order details to **Sanity CMS** where the order is saved in the database.
    - During checkout, the **Payment Gateway** is used to securely process the user's payment (via credit card or PayPal).
    - Once payment is processed, a confirmation is sent back to the **Frontend** and the **Order is recorded** in **Sanity CMS**.
3. **Shipment Tracking:**
    - After the user completes the purchase, the **Shipment Tracking API** is used to track the order's delivery status.
    - The user can view real-time shipment updates on the **Frontend** (e.g., "In Transit" or "Delivered").
4. **Payment Confirmation:**
    - Payment details are processed through the **Payment Gateway** and once successful, the user is notified via the **Frontend**, and the order is marked as completed in **Sanity CMS**.

Data Flow

1. **User Logs In:**
    - The frontend fetches user data from **Sanity CMS** to display order history and profile information.
2. **Product Browsing:**
    - User browses products → Frontend fetches product data from **Sanity CMS** using the **Product Data API**.
3. **Order Placement:**
    - User adds items to cart → Frontend sends order details to **Sanity CMS** → Order is saved in CMS.
4. **Payment:**
    - User enters payment details → Payment Gateway processes transaction → Confirmation sent back to user.
5. **Shipment Tracking:**
    - After order confirmation, **Shipment Tracking API** provides real-time updates on delivery status, shown on the frontend.

API Requirements for Q-Commerce Website

List of API endpoints based on the data schema and required functionalities:

---

1. Product Management

Endpoint Name: /products

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


```
[
 {
  "id": 1,
  "name": "Fresh Apples",
  "price": 5.99,
  "stock": 100,
  "image": "https://example.com/images/apples.jpg"
 },
 {
  "id": 2,
  "name": "Organic Milk",
  "price": 2.99,
  "stock": 50,
  "image": "https://example.com/images/milk.jpg"
 }
]
```


---

2. Order Management

Endpoint Name: /orders

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

- **Payload Example:**

```
{
 "customerId": 123,
 "items": [
  { "productId": 1, "quantity": 2 },
  { "productId": 2, "quantity": 1 }
 ],
 "paymentStatus": "Paid",
 "deliveryAddress": "123 Main Street, City, Country"
}
```

- **Response Example:**

```
{
 "orderId": 456,
 "status": "Order Created",
 "totalAmount": 14.97
}
```

---

3. Delivery Tracking

Endpoint Name: /express-delivery-status

- **Method:** GET
- **Description:** Fetch real-time delivery updates for perishable items.
- **Response Example:**

```
{
 "orderId": 789,
 "status": "In Transit",
 "ETA": "15 mins"
}
```

---

4. Shipment Tracking

Endpoint Name: /shipment

- **Method:** GET

- **Description:** Track order shipment status via a third-party API.
- **Response Example:**


```
{
  "shipmentId": "SHIP123",
  "orderId": 456,
  "status": "Out for Delivery",
  "expectedDeliveryDate": "2025-01-17T14:00:00Z"
}
```


---

4. Shipment Tracking

Endpoint Name: /shipment

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


```
{
  "shipmentId": "SHIP123",
  "orderId": 456,
  "status": "Out for Delivery",
  "expectedDeliveryDate": "2025-01-17T14:00:00Z"
}
```


5. Customer Registration

Endpoint Name: /register

- **Method:** POST
- **Description:** Create a new customer profile in Sanity CMS.
- **Payload Example:**


```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "password": "securePassword123",
  "phone": "+1234567890",
```

```
  "address": "123 Main Street, City, Country"
}
```

- **Response Example:**

```
{
  "customerId": 123,
  "status": "Registration Successful"
}
```

---

6. Payment Processing

Endpoint Name: /payment

- **Method:** POST
- **Description:** Process payment through the Payment Gateway.
- **Payload Example:**

```
{
  "orderId": 456,
  "amount": 14.97,
  "paymentMethod": "Credit Card",
  "cardDetails": {
    "cardNumber": "4111111111111111",
    "expiryDate": "12/25",
    "cvv": "123"
  }
}
```

- **Response Example:**

```
{
  "transactionId": "TXN789",
  "status": "Payment Successful"
}
```

---

7. Product Categories

Endpoint Name: /categories

- **Method:** GET
- **Description:** Fetch all product categories from Sanity CMS.

```
[
 { "id": 1, "name": "Fruits & Vegetables" },
 { "id": 2, "name": "Dairy Products" },
 { "id": 3, "name": "Bakery" }
]
```

---

8. User Login

Endpoint Name: /login

- **Method:** POST
- **Description:** Authenticate the user and generate an access token.
- **Payload Example:**

```
{
 "email": "john.doe@example.com",
 "password": "securePassword123"
}
```

- **Response Example:**

```
{
 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
 "status": "Login Successful"
}
```

---

9. Order History

Endpoint Name: /orders/history

- **Method:** GET
- **Description:** Retrieve past orders for a logged-in customer.
- **Response Example:**

```
[
 {
  "orderId": 456,
```

```
  "date": "2025-01-15T10:00:00Z",
  "items": [
   { "productId": 1, "name": "Fresh Apples", "quantity": 2, "price": 5.99 },
   { "productId": 2, "name": "Organic Milk", "quantity": 1, "price": 2.99 }
  ],
  "totalAmount": 14.97,
  "status": "Delivered"
 }
]
```


Marketplace Technical Foundation - Food Tuck

1. System Architecture Overview

Architecture Diagram

```
[User (Browser)]
      |
[Frontend (Next.js)]  ↔ [Sanity CMS (Product Data)]
      |                       |
      |                  [Product Data API]
      |
[Third-Party APIs] ↔ [Payment Gateway] ↔ [Payment Processor]
      |
[Shipment Tracking API]
```

Component Descriptions

- **Frontend (Next.js):** Provides a responsive and user-friendly interface for browsing products, managing carts, and completing orders.

- **Sanity CMS:** Acts as the backend for managing product, customer, and order data, offering a flexible schema structure.
- **Product Data API:** Fetches and updates product details in real-time.
- **Third-Party APIs:** Handles shipment tracking, payment processing, and additional logistics.
- **Payment Gateway & Processor:** Securely processes payments and returns transaction statuses.
- **Shipment Tracking API:** Tracks real-time delivery status and estimated arrival times.

---

2. Key Workflows

1. Product Browsing

- **Steps:**
    1. User navigates to the homepage.
    2. Frontend sends a request to /products API.
    3. Sanity CMS responds with product data (name, price, stock, image).
    4. Products are displayed dynamically on the frontend.

2. Order Placement

- **Steps:**
    1. User adds products to the cart.
    2. At checkout, user details and payment info are submitted via /orders API.
    3. Sanity CMS saves the order data.
    4. Payment is processed via the Payment Gateway.
    5. Confirmation is displayed to the user.

3. Express Delivery Tracking

- **Steps:**
    1. User places an order for perishable items.
    2. Frontend requests /express-delivery-status API for real-time updates.
    3. Third-Party API responds with delivery status and ETA.
    4. Status is displayed on the user dashboard.

---

3. API Endpoints

Endpoint	Method	Purpose	Response Example
/products	GET	Fetch all available products.	[{ "id": 1, "name": "Fresh Apples", "price": 5.99, "stock": 100, "image": "url" }]
/orders	POST	Create a new order.	{ "orderId": 456, "status": "Order Created", "totalAmount": 14.97 }
/express-delivery-status	GET	Fetch real-time delivery updates.	{ "orderId": 789, "status": "In Transit", "ETA": "15 mins" }
/shipment	GET	Track shipment for an order.	{ "shipmentId": "SHIP123", "orderId": 456, "status": "Out for Delivery", "expectedDelivery": "2025-01-17" }
/payment	POST	Process user payment securely.	{ "transactionId": "TXN789", "status": "Payment Successful" }

---

4. Sanity Schema Example

Product Schema

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'stock', type: 'number', title: 'Stock Level' },
    { name: 'image', type: 'image', title: 'Product Image' }
  ]
};
```

Order Schema

```
export default {
  name: 'order',
  type: 'document',
  fields: [
```

```
    { name: 'customerId', type: 'string', title: 'Customer ID' },
    { name: 'items', type: 'array', of: [{ type: 'object', fields: [
      { name: 'productId', type: 'string', title: 'Product ID' },
      { name: 'quantity', type: 'number', title: 'Quantity' }
    ] }] },
    { name: 'totalAmount', type: 'number', title: 'Total Amount' },
    { name: 'status', type: 'string', title: 'Order Status' }
  ]
};
```

Customer Schema

```
export default {
  name: 'customer',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Customer Name' },
    { name: 'email', type: 'string', title: 'Email Address' },
    { name: 'phone', type: 'string', title: 'Phone Number' },
    { name: 'address', type: 'object', title: 'Address', fields: [
      { name: 'street', type: 'string', title: 'Street' },
      { name: 'city', type: 'string', title: 'City' },
      { name: 'zipCode', type: 'string', title: 'ZIP Code' }
    ]}
  ]
};
```

Delivery Schema

```
export default {
  name: 'delivery',
  type: 'document',
  fields: [
    { name: 'orderId', type: 'string', title: 'Order ID' },
    { name: 'status', type: 'string', title: 'Delivery Status' },
```

```
  { name: 'ETA', type: 'string', title: 'Estimated Time of Arrival' },
  { name: 'trackingId', type: 'string', title: 'Tracking ID' }
 ]
};
```

Payment Schema

```
export default {
 name: 'payment',
 type: 'document',
 fields: [
  { name: 'orderId', type: 'string', title: 'Order ID' },
  { name: 'transactionId', type: 'string', title: 'Transaction ID' },
  { name: 'status', type: 'string', title: 'Payment Status' },
  { name: 'amount', type: 'number', title: 'Amount Paid' },
  { name: 'paymentMethod', type: 'string', title: 'Payment Method' }
 ]
};
```

5. TECHNICAL ROADMAP

1. **Phase 1: Setup and Initialization**
    - Configure Next.js for frontend development.
    - Set up Sanity CMS with schemas for products and orders.
    - Integrate basic APIs for product data.
2. **Phase 2: Core Features**
    - Implement product browsing, cart functionality, and order placement.
    - Integrate Payment Gateway for secure transactions.
3. **Phase 3: Advanced Features**
    - Add real-time delivery tracking using Shipment Tracking API.
    - Optimize frontend performance for better user experience.
4. **Phase 4: Testing and Deployment**
    - Conduct end-to-end testing of workflows.

- Deploy the platform on a cloud provider (e.g., Vercel, Netlify).