DAY 2
PLANNING THE TECHNICAL FOUNDATION

Prepared by
Urooj Sadiq

SHOP.COM E-Commerce Website

1. Introduction

SHOP.COM is an e-commerce platform designed to feature clothes for men, women, and kids. This document outlines the technical requirements, architecture, and integration details for the website's development.

---

2. Technical Requirements

Frontend Requirements

User-Friendly Interface

- The website should provide a simple and intuitive interface that allows users to:
  - Browse and filter products based on categories (Men, Women, Kids).
  - View product details, including images, price, and description.
  - Add products to the shopping cart with ease.

Responsive Design

- The design will be responsive, ensuring the website is accessible on all devices,

Essential Pages

- **Home Page**:
  - Display featured products, categories, and promotional offers.
  - Provide navigation to all major sections of the website.
- **Product Listing Page**:
  - List all available products for each category.
  - Include filter options such as size, color, price, and brand.
- **Product Details Page**:
  - Show detailed information for each product, including high-resolution images, size options, colors, and product specifications.
  - Provide an option to add the product to the cart.
- **Cart Page**:
  - Users can view all items in their cart, update quantities, remove items, and proceed to checkout.
- **Checkout Page**:
  - Collect shipping details, billing information, and payment options.
  - Provide a review of the order before submitting.

Sanity CMS as Backend

Sanity CMS for Data Management

- Product details such as name, description, price, availability, and images will be stored and managed within Sanity CMS and categories(Men, Women, Kids) will be maintained to organize the content.
- Sanity CMS will store customer details, including name, shipping address, and order history. Also order records will be created automatically when a customer completes a purchase.

Designing Schemas in Sanity

- **Product Schema**:
  - Define schema for products with attributes like name, description, price, images, available sizes, and colors.
- **Customer Schema**:
  - Include necessary fields such as name, email, shipping address, and phone number.
- **Order Schema**:
  - Define the structure for order data to track customer purchases, including the products ordered, order status, and delivery details.

Third-Party APIs

Shipment Tracking API

Integrate a third-party shipment tracking API (e.g., ShipEngine, Shippo) to provide customers with real-time updates on the status of their orders. Customers should be able to track their shipment by entering the tracking number.

Payment Gateway API

Integrate with popular payment gateways like **Stripe** for processing customer payments. Supported Payment Methods include Credit/Debit Cards, PayPal, Other online payment options

---

3. Architecture and Flow

Frontend Flow:

- The frontend will be developed using ReactJS or Next.js for a dynamic, fast, and responsive experience.
- All product data and customer details will be fetched from the backend through Sanity CMS and displayed on the frontend.

Backend Flow:

- Sanity CMS will serve as the backend database, providing content and data management.
- Third-party APIs (e.g., for payments, shipment tracking) will be integrated as needed to extend the functionality of the platform.

Data Flow:

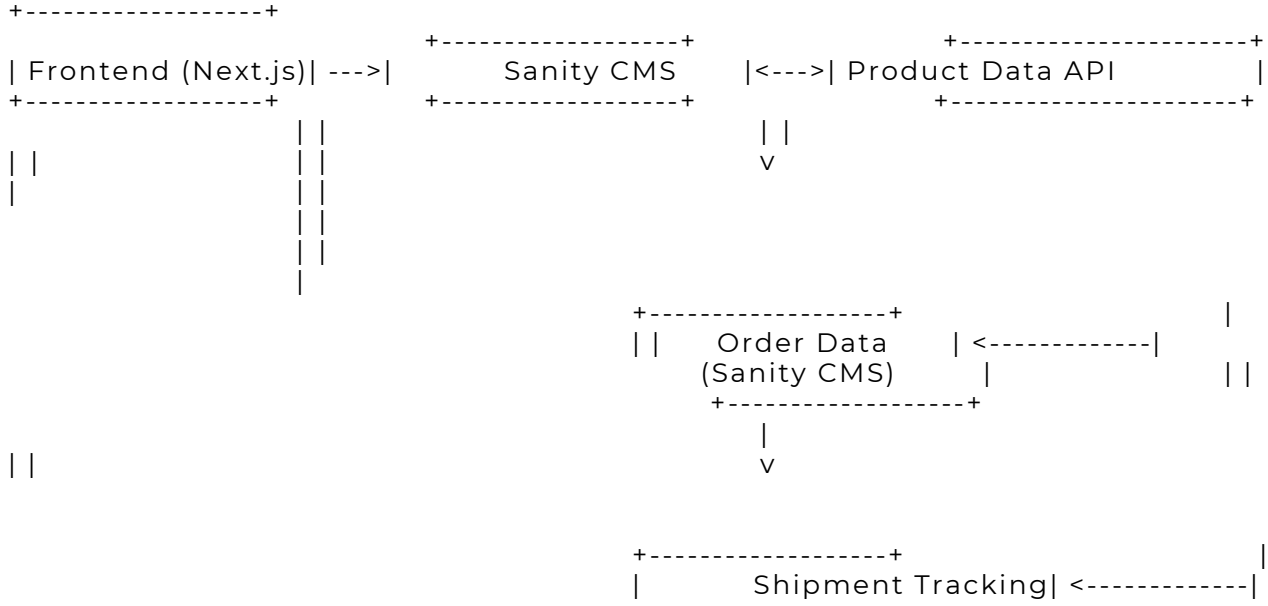
1. **User Browsing**:
    - A user visits the website and browses the product listing page, where products are fetched from the Sanity CMS database.
2. **Cart and Checkout**:
    - Products are added to the shopping cart. Once ready to checkout, the user proceeds to the checkout page to enter their shipping details.
3. **Order Confirmation**:

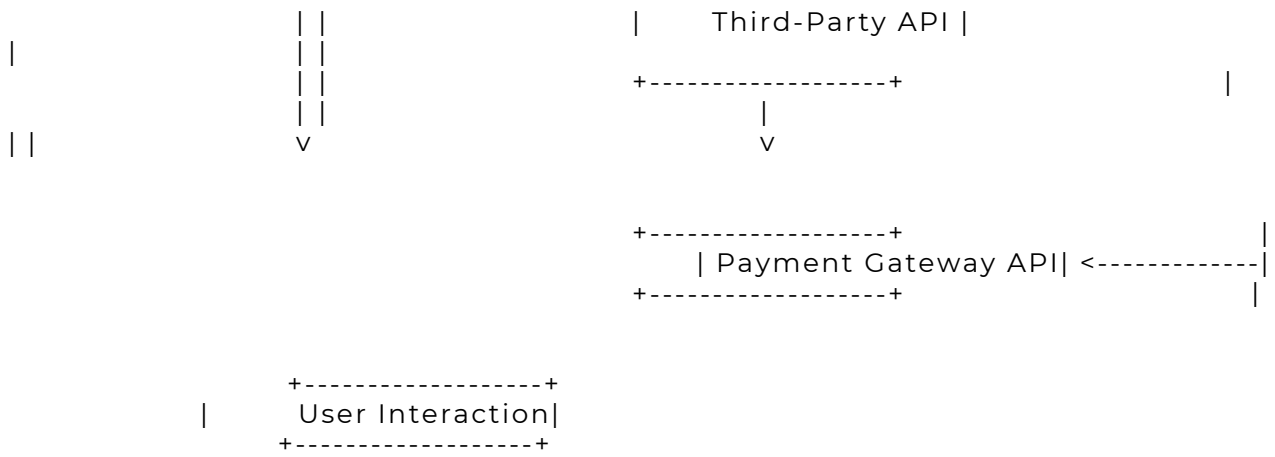
Upon successful payment, an order record is created in the Sanity CMS, and an order confirmation page is shown to the user with their purchase summary.

---

4. System Architecture

High-Level Diagram:





Key Workflows:

---

1. User Registration(Optional):

- **Flow**:
  - The user signs up on the website.
  - The data (name, email, etc.) is stored in **Sanity CMS**.
  - A confirmation email or message is sent to the user.
- **Architecture**:
  - Frontend (Next.js) communicates with Sanity CMS to store user data.
  - A confirmation is sent through an email service integration.

---

2. Product Browsing:

- **Flow**:
  - The user browses the product categories (Men, Women, Kids).
  - The **Frontend (Next.js)** makes a request to the **Product Data API** from **Sanity CMS** to fetch the product data.
  - Products are dynamically displayed on the frontend based on the response.
- **Architecture**:
  - The **Frontend** calls the **Sanity CMS API** for product data.
  - The **Sanity CMS API** fetches data from the **Product Data API** and returns it to the frontend.

---

3. Order Placement:

- **Flow**:

o The user adds items to the cart and proceeds to checkout. o Order details are captured and stored in **Sanity CMS**. o A confirmation of the order is shown on the frontend.

- **Architecture**:
  - o The **Frontend (Next.js)** makes an API call to **Sanity CMS** to record the order data.
  - o **Sanity CMS** stores the order information in the **Order Data API**.

---

4. Shipment Tracking:

- **Flow**:
  - o The user checks the status of their order.
  - o Shipment tracking details are fetched from a **Third-Party API**.
  - o The shipment status is displayed to the user in real-time.
- **Architecture**:
  - o The **Frontend (Next.js)** communicates with the **Third-Party Shipment Tracking API** to retrieve the shipment status.
  - o The **Third-Party API** provides the tracking information, which is displayed on the frontend.

---

5. Payment Processing:

- **Flow**:
  - o The user enters payment details at checkout.
  - o Payment details are securely processed through the **Payment Gateway API** (e.g., Stripe).
  - o The payment is confirmed, and a successful payment message is displayed to the user.
- **Architecture**:
  - o The **Frontend (Next.js)** sends the payment request to the **Payment Gateway API**.
  - o The **Payment Gateway API** processes the payment and returns a success or failure response.
  - o Payment status is saved in **Sanity CMS** for order records.

---

5. API Endpoints

Endpoint	Method	Purpose	Response Example

/products	GET	Fetches all product details	{ "id": 1, "name": "Product A", "price": 100 }
/orders	POST	Creates a new order	{ "orderId": 789, "status": "Success" }
/shipment	GET	Tracks order status via shipment API	{ "shipmentId": "A123456789", "status": "In Transit", "ETA": "2025-01-20" }
/payment	POST	Processes a payment for an order	{ "transactionId": "txn_123456789", "status": "Successful" }

Key Workflows:

- **User Browsing Products**:
  - The frontend calls the /products endpoint to fetch all products.
  - The product list is displayed to the user with details like name, price, and stock.
- **Order Placement**:
  - Once a user adds products to the cart and proceeds to checkout, the frontend sends a POST request to /orders to store the order data in Sanity CMS.
  - The order confirmation is returned with an orderId and status.
- **Shipment Tracking**:
  - The user checks the status of their order, and the frontend calls the /shipment endpoint to get the shipment details from a third-party API.
  - The user sees the current shipment status and expected delivery date.
- **Payment Processing**:
  - After the user confirms the order, the frontend sends a POST request to /payment to process the payment.
  - Payment status (successful or failed) is returned, and the user sees confirmation on their screen.

---

6. Sanity Schema Example

Product Schema

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: id, type: 'string', title: 'ProductID},
    { 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' }, { name: description, type: 'string',
    title: 'Description' }, { name: 'size', type: 'number', title: 'Size' }, {
    name: 'image', type: 'image', title: 'Product Image' }, { name:
    'discount', type: 'number', title: 'Category' }, { name: 'colors',
    type: 'string', title: 'Colors' }, { name: 'rating', type: 'number',
    title: 'Rating' }, { name: 'reviews', type: 'array', title: 'Reviews' },


  ]
};
```

Order Schema

```
export default {
name: 'order',
type: 'document',
fields: [
    { name: 'customerId',
      type: 'reference',
      to: { type: 'customer' },
      title: 'Customer'
    },
    { name: 'products',
      type: 'array',
      of: [{ type: 'reference', to: { type: 'product' } }]
    },
    { name: 'paymentStatus',
      type: 'string',
      title: 'Payment Status'
    },
    { name: 'shippingAddress',
      type: 'object',
      fields: [
      { name: 'street', type: 'string', title: 'Street' },
      { name: 'city', type: 'string', title: 'City' },
      { name: 'state', type: 'string', title: 'State' },
      { name: 'zipCode', type: 'string', title: 'Zip Code' }
    ]}
  ]
};
```

Shipment Schema

```
export default {
  name:   'shipment',
  type:    'document',
  fields: [
      {
        name: 'shipmentId',
        type: 'string',
        title: 'Shipment ID'
      },
```

```
      {
        name: 'orderId',
        type: 'reference',
        to: [{ type: 'order' }],
        title: 'Order'
      },
      {
        name:     'status',     type:
        'string',   title:   'Shipment
        Status'
      },
      {
        name: 'expectedDeliveryDate',
        type: 'datetime',
        title: 'Expected Delivery Date'
      },
    ],
 };
```

Payment Schema

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