Technical Documentation – Ecommerce Website

Overview

This document describes the architecture of the marketplace website. It includes the frontend, backend, and third-party API integrations.

- **Frontend**: Built using Next.js, providing a responsive and user-friendly interface across 7 pages:
  - Homepage
  - Products Page
  - Single Product Page
  - Cart
  - Contact Us
  - FAQ
  - About Us
- **Backend**: Powered by Sanity CMS to manage products, orders, and customer data.
- **Third-Party APIs**: Used for shipment tracking and secure payment processing.
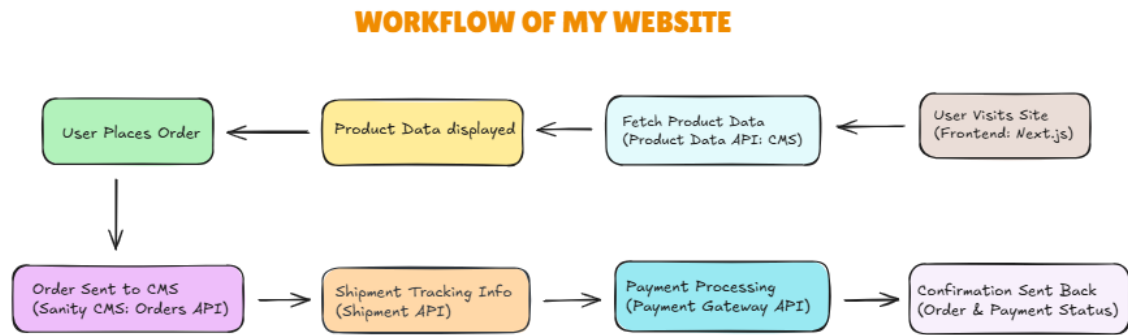
Interaction Flow

1. Users interact with the **Frontend** to browse products, add items to the cart, and place orders.
2. The **Frontend** communicates with the **Backend** (Sanity CMS) to fetch and update data.
3. Third-party APIs are integrated for shipment tracking and payment processing.

API Specification Document

Endpoint Name	Method	Description	Request Example	Response Example
/products	GET	Fetch product details	/products?page=1	{ "id": 1, "name": "Sofa", "price": 200, "stock": 5 }
/orders	POST	Save order details	{ "customerId": 123, "items": [1, 2] }	{ "orderId": 123, "status": "Order Placed" }
/shipment	GET	Get shipment tracking info	/shipment?orderId=123	{ "shipmentId": 456, "status": "In Transit" }
/payment-status	GET	Check payment confirmation	/payment-status?paymentId=789	{ "paymentId": 789, "status": "Success" }

Workflow Diagram

User Places Order ← Product Data displayed ← Fetch Product Data (Product Data API: CMS) ← User Visits Site (Frontend: Next.js)

Order Sent to CMS (Sanity CMS: Orders API) → Shipment Tracking Info (Shipment API) → Payment Processing (Payment Gateway API) → Confirmation Sent Back (Order & Payment Status)
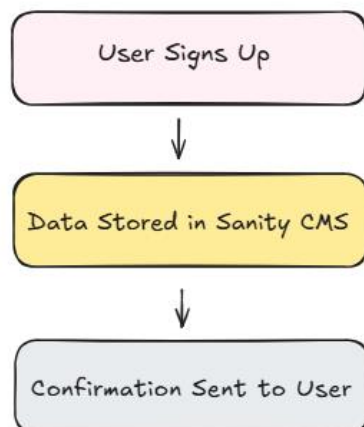
Key Workflows

1) User Registration:

User signs up -> Data stored in Sanity CMS -> Confirmation sent to user.

USER REGISTRATION WORKFLOW

User Signs Up

↓

Data Stored in Sanity CMS

↓

Confirmation Sent to User
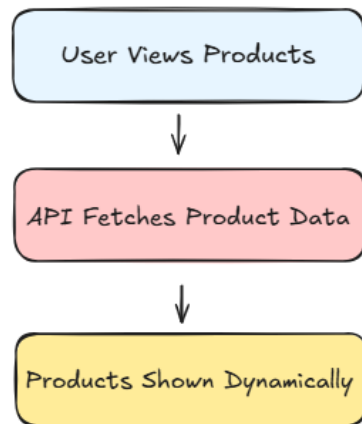
2) Product Browsing:

User views product categories -> Sanity API fetches data -> Products displayed dynamically.

PRODUCT BROWSING WORKFLOW
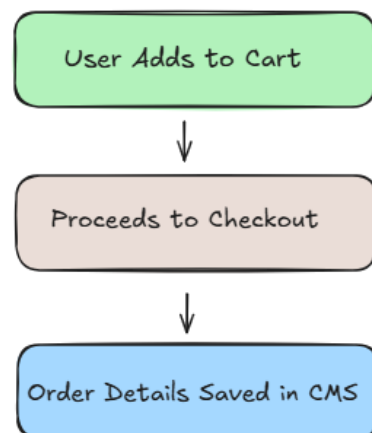
```
┌─────────────────────┐
│  User Views Products │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│ API Fetches Product Data │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│ Products Shown Dynamically │
└─────────────────────┘
```

3) Order Placement:

User adds items to cart -> Proceeds to checkout -> Order details saved in Sanity CMS.

ORDER PLACEMENT WORKFLOW

```
┌─────────────────────┐
│   User Adds to Cart  │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│ Proceeds to Checkout │
└─────────────────────┘
           │
           ▼
┌─────────────────────┐
│ Order Details Saved in CMS │
└─────────────────────┘
```

4) Shipment Tracking:

Order status updates fetched via third-party API -> Displayed to the user.

SHIPMENT TRACKING WORKFLOW

```
┌─────────────────────────┐
│   User Checks Status    │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│ API Fetches Real-Time Data │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│  Status Displayed to User  │
└─────────────────────────┘
```
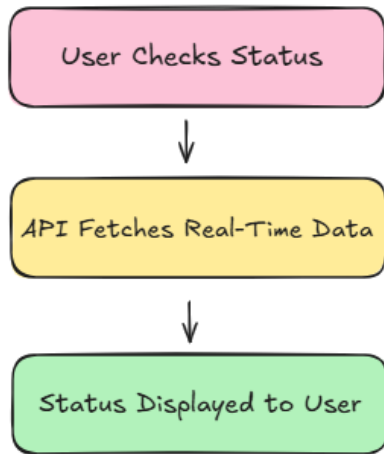
Data Schema Design

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: 'category', type: 'string', title: 'Category' },

   { name: 'image', type: 'image', title: 'Product Image' }

  ]

};
```

Technical Roadmap

1. **Frontend Development**:
   - Build and design 7 pages: Homepage, Products Page, Single Product Page, Cart, Contact Us, FAQ, About Us.
   - Ensure responsive design for all devices.
2. **Backend Integration**:
   - Configure Sanity CMS for data management (products, orders, customers).
3. **API Implementation**:
   - Develop endpoints for fetching products, saving orders, tracking shipments, and handling payments.
4. **Testing and Optimization**:
   - Perform end-to-end testing of user workflows.
   - Optimize performance and fix bugs.
5. **Launch**:
   - Deploy the system and monitor for feedback.