Marketplace Technical Foundation

Comforty

Shabnam Wahid

Marketplace Technical Foundation

1. Define Technical Requirements

Frontend Requirements:

User-friendly Interface: Provide an intuitive and easy-to-navigate UI for browsing products.

Responsive Design: Ensure the website adapts for both mobile and desktop users.

Essential Pages: Implement the following pages:

Home

Product Listing

Product Details

Cart

Checkout

Order Confirmation

Backend Requirements (Sanity CMS):

Sanity CMS as Back-end: Use Sanity CMS for managing product data, customer details, and order records.

Schema Design: Develop schema in Sanity CMS to align with the marketplace's business goals. Examples include product details, order records, and customer information.

Third-Party API s:

API Integrations:

Shipment Tracking API: Fetch real-time order updates.

Payment Gateway API: Securely process payments

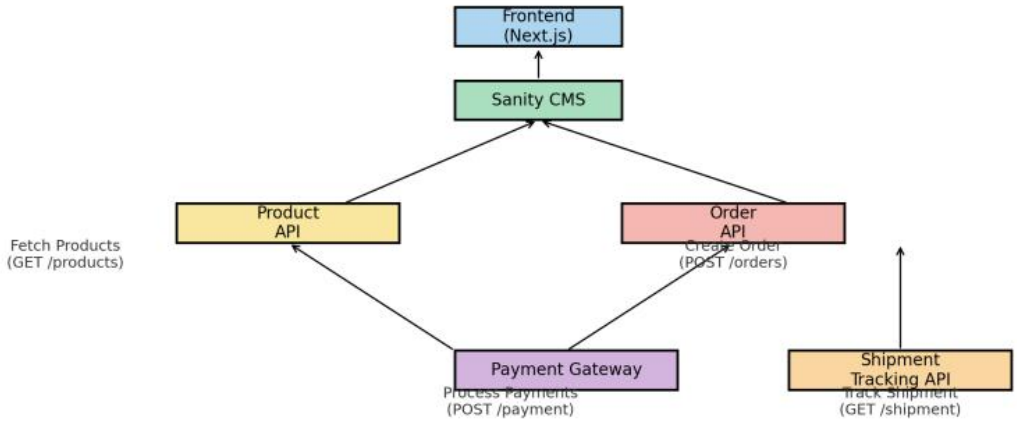
Data Utilization: Ensure API s return the necessary data for seamless front-end integration.

2. Design System Architecture

High-Level Overview:

The architecture includes components and their interactions, ensuring seamless data flow between the front-end, back-end, and external services.

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



Example Data Flow:

A user browses the marketplace via the front-end (Next.js).

The front-end makes requests to the **Product Data API** (backed by Sanity CMS) to display products dynamically.

Order Details are stored in Sanity CMS after checkout.

Shipment Tracking API fetches real-time status updates for orders.

Payments are processed via the **Payment Gateway API** and confirmations are recorded in Sanity CMS.

Workflows:

User Registration:

User signs up.

Data stored in Sanity CMS.

Confirmation sent to the user.

Product Browsing:

User navigates through product categories.

Product data fetched via Sanity CMS API and displayed on the front-end.

Order Placement:

User adds items to the cart and checks out.

Order data stored in Sanity CMS.

Shipment Tracking:

Real-time updates fetched from Third-Party API s and displayed on the front-end.

3. Plan API Requirements

API Endpoints:

General API Examples:

Fetch Products

Endpoint: `/products`

Method: GET

Description: Retrieve product details from Sanity CMS.

Response: `{ "id": 1, "name": "Product A", "price": 100, "stock": 20, "image": "url" }`

Create Order

Endpoint: `/orders`

Method: POST

Description: Submit new order details to Sanity CMS.

Payload: `{ "customer": {}, "products": [], "paymentStatus": "Paid" }`

Response: `{ "orderId": 123, "status": "Success" }`

Shipment Tracking

Endpoint: `/shipment`

Method: GET

Description: Fetch shipment details via Third-Party API.

Response: `{ "shipmentId": 456, "status": "In Transit", "ETA": "2 days" }`

4. Write Technical Documentation

System Architecture Overview:

A diagram illustrating the interaction between the frontend, Sanity CMS, and third-party API s.

Description of how each component works within the system.

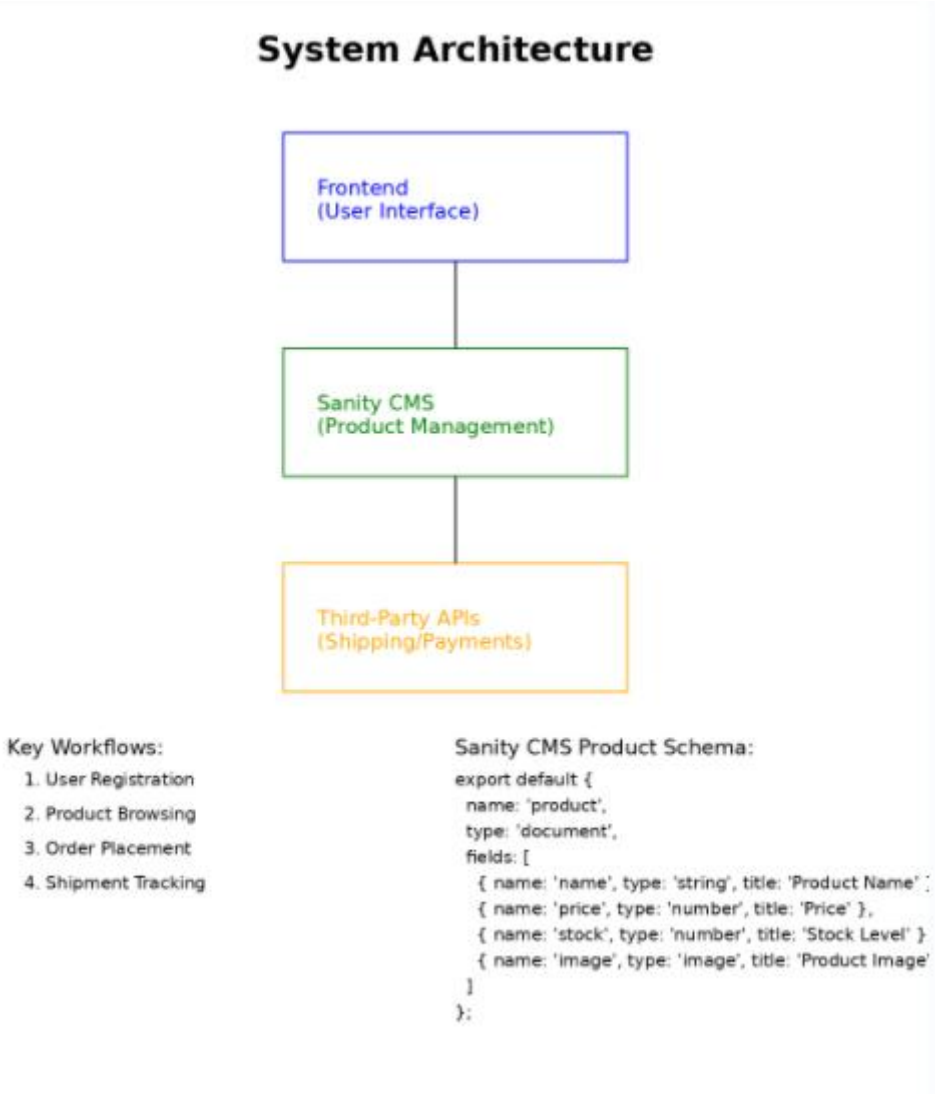
Key Workflows:

User Registration

Product Browsing

Order Placement

Shipment



Trackin

Sanity CMS 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' }
  ]
};
```

API Specification Document

A detailed table of endpoints, methods, payloads, and responses.

Endpoint	Method	Description	Status Code
`/products`	**GET**	Fetch all available products	200 OK
`/products/:id`	**GET**	Fetch a product by its ID	200 OK
`/order`	**POST**	Create a new order	201 Created
`/order/:id`	**GET**	Get details of an order by ID	200 OK
`/shipment`	**GET**	Get shipment information	200 OK

.