# MARKETPLACE TECHNICAL FOUNDATION - [SHOP.CO]

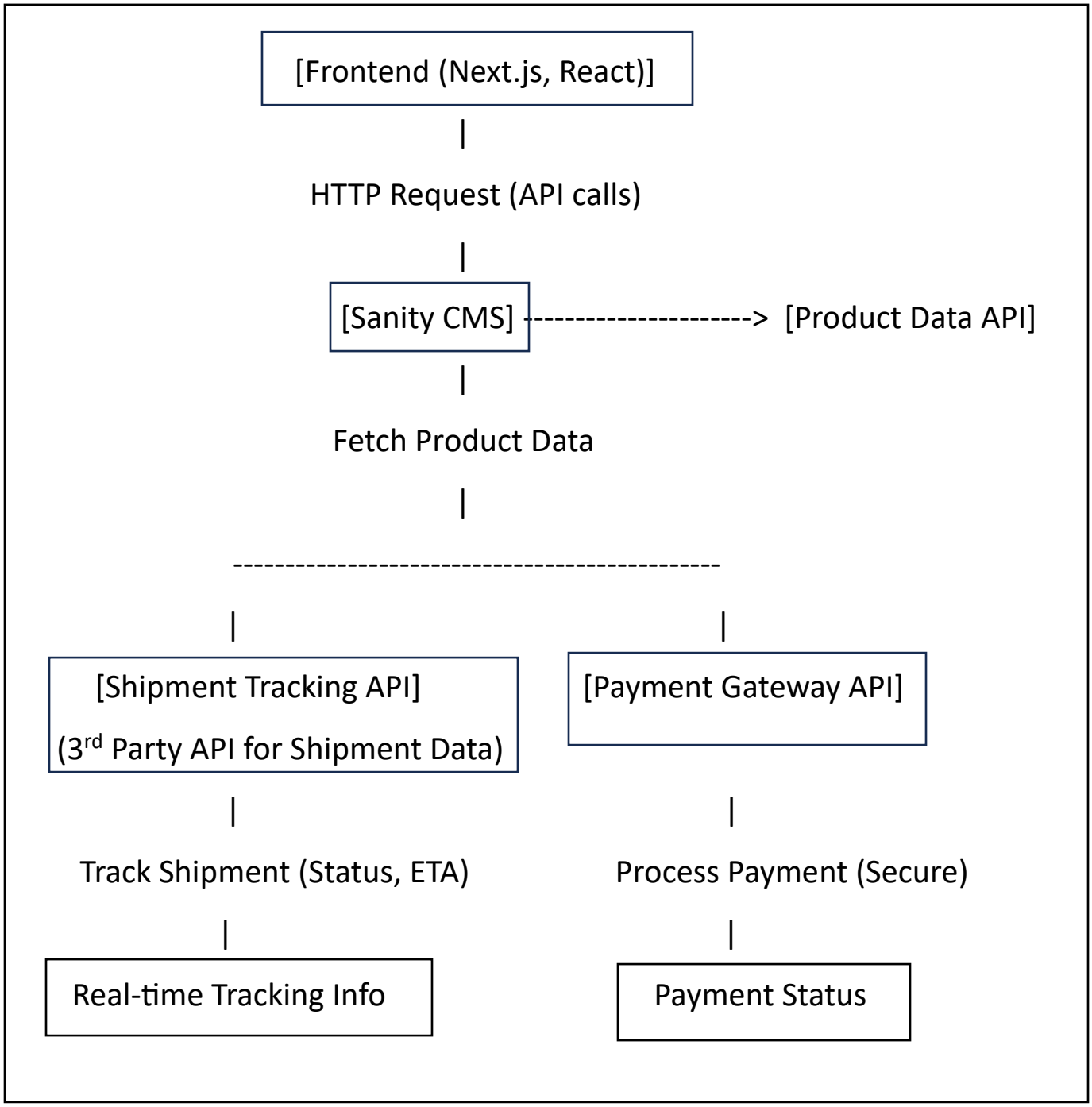
## 1.Define Technical Requirements

- Frontend Requirements:
    - User-Friendly Interface: Design a seamless and engaging interface for users to easily browse, filter, and select products.
    - Responsive Design: Ensure the marketplace is accessible across devices (desktop, tablet, and mobile).
    - Essential Pages:
        - Home Page
        - Product Listing Page
        - Product Details Page
        - Cart Page
        - Checkout Page
        - Order Confirmation Page
- Backend with Sanity CMS:
    - Use Sanity CMS to store and manage product data, customer details, and order records.
    - Data Schema Design in Sanity CMS to reflect the core entities from Day 1 (products, orders, customers).
    - Sanity will also handle updating inventory and product status.

- Third-Party API Integration:
    - Shipment Tracking API: Fetch real-time shipment updates.
    - Payment Gateway API: Securely process payments.

- Stock Management API (if required): Integrate for real-time stock updates (if not managed by sanity).

## 2.Design System Architecture



**Explaination:**

1. Frontend (Next.js):
    - The user's interface built with Next.js is responsible for displaying product listings, details, and managing the shopping cart and checkout flow.
    - The frontend interacts with Sanity CMS to fetch product data dynamically.
2. Sanity CMS:
    - Act as the central Content Management System (CMS), storing and managing all products, orders, and customer data.
    - The frontend sends API requests to Sanity CMS, which responds with product details and inventory updates.
3. Third-Party API Integrations:
    - Shipment Tracking API: Tracks shipment status, and other details for customer orders.
    - Payment Gateway API: Securely processes payments and returns the payment status to the frontend and Sanity CMS.
4. Product Data API:
    - Sanity CMS exposes an API endpoint to fetch product data, which is requested by the frontend when displaying products to the user.
5. User Flow:
    - When a user adds items to their cart and proceeds to checkout, their order details are saved in Sanity CMS, and payment is processed via the Payment Gateway API.
    - Shipment information is fetched in real-time using Shipment Tracking API.

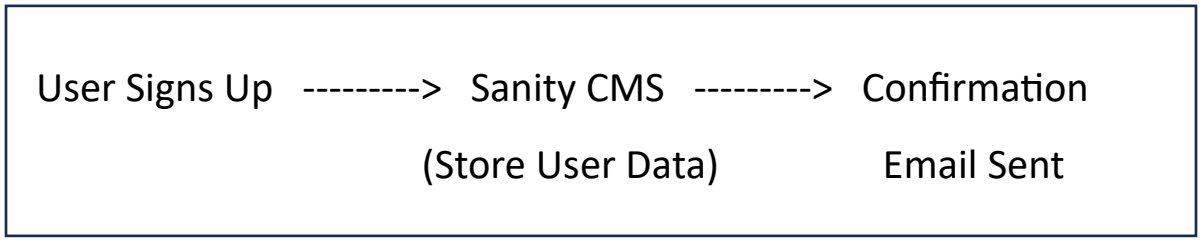
## 3. Key WorkFlows

These workflows demonstrate the sequence of actions between users and the system.

User Registration Workflow:

1. User Sign Up:
   - User inputs personal details.
   - Sanity CMS stores user data (name, email, etc).
   - Confirmation is sent to the user's email.

Workflow Diagram:




Product Browsing Workflow:

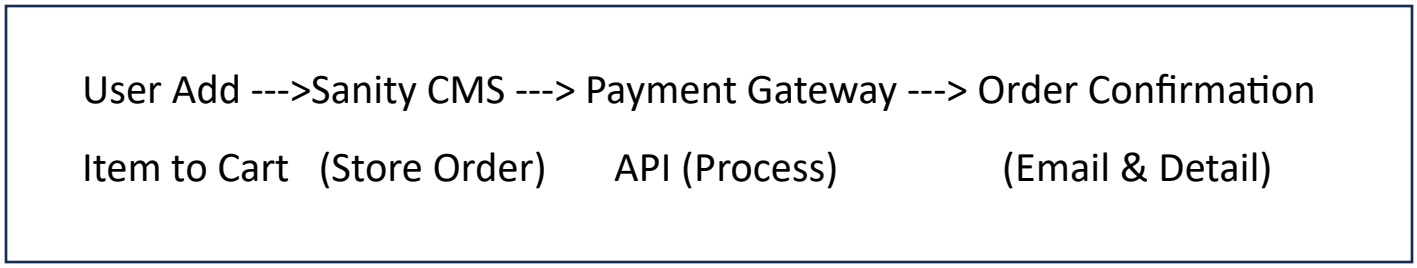
1. User Browses Products:
   - Frontend requests product data from Sanity CMS.
   - Sanity CMS returns product details (name, price, stock).
   - Products are displayed dynamically.

Workflow Diagram:

Order Placement Workflow:

1. User Places Order:
   - User adds items to the cart and proceeds to checkout.
   - Order details are saved in Sanity CMS.
   - Payment is processed via Payment Gateway API.

Workflow Diagram:




Shipment Tracking Workflow:

1. User Tracks Shipment:
   - After order placement, the system fetches shipment tracking from the Shipment Tracking API.
   - Real-time delivery updates are shown to the user.

Workflow Diagram:

## 4. API Design and Documentation

**API Endpoints:**

1. /products (GET):
    - Purpose: Fetch all data from Sanity CMS.
    - Response:

```
{
“id” : 1,
“name” : “Product A”,
“price” : 100,
“stock” : 10,
“image” : “url-to-image”
}
```

2. /orders (POST):
    - Purpose: Place a new order.
    - Payload:

```
{
“customerInfo” : { “name” : “John Doe” , “email” :
“john@example.com” },
“productDetails” : [{ “productId” : 1, “quantity” : 2 }],
“paymentStatus” : “Success”
}
```

    - Response:

```
{
“orderId” : 12345,
“status” : “Confirmed”
}
```

3. /shipment (GET):
    - Purpose: Track shipment status.
    - Response:


```
{
“orderId” : 12345,
“status” : “In Transit”,
“ETA” : “2025-01-18 14:00”
}
```