Marketplace Technical Foundation – HEKTO

Introduction

Our **e-commerce** platform aims to provide **luxury Chairs and Electronics.** This document outlines the technical foundation required to build a scalable, userfriendly marketplace.

Define Technical Requirements:

1. Frontend Requirements:

- ***User-friendly Interface***
- ***Responsive Design*** - Ensure seamless navigation and browsing on mobile, tablet, and desktop devices.
- ***Essential Pages:*** ○ Home:
- Showcase featured collections, seasonal trends, and a prominent search bar.
- Include sections for Latest Collection, Top Categories, Featured Products, and Trending Products.
    - About:
        - Highlight the brand story, mission, values, and what sets your home décor business apart.
    - Contact Us:
        - Provide a contact form for customer inquiries (fields for name, email, and message).
        - Include phone numbers, email addresses, and a physical address (if applicable).
        - Embed a Google Map for location visibility.
    - Product Listing/Shop:
        - Offer filtering options (e.g., by price).
    - Product Details:
        - Highlight product features such as dimensions, materials, variations and care instructions.
        - Provide customer reviews and ratings.
    - Cart:
        - Include an estimated shipping cost calculator.
    - Checkout:
        - Enable guest checkout and account creation options.
        - Include secure payment fields and shipping address validation.
    - Order Confirmation:
        - Show order summary with delivery timeframes and tracking details.

2. Sanity CMS as Backend:

Sanity CMS for managing product data, orders, and customer details.

Here's a detailed schema design for **Product**, **Order**, **Customer**, **Payment**, **Shipment**, and **Delivery Zone** using Sanity CMS:

[Product] export

interface Product {

_id: string; name: string; description: string; price: number;

images: { _type: 'image'; asset: { _ref: string; _type: 'reference' }

}[]; dimensions: string; material: string; stock: number; category:

string; tags: string[];

reviews: number

}

[Order] export

interface Order {

_id: string; productID:

string; Quantity:

number totalAmount:

number; orderDate:

string

}

[Customer] export

interface Customer {

_id: string;

name: string;

email: string;

phone: string;

address: string;

}

[Payment] export

interface Payment {

```
_id: string;  order: Order;  paymentMethod:
'Credit Card' | 'PayPal' | 'Stripe' ;  status: 'Pending'
| 'Completed' | 'Failed';  transactionId: string;
Amount: number;
paymentDate: string;
}
```

[Shipment] export

```
interface Shipment {
_id: string;  order: Order;  trackingNumber: string;
status: 'Pending' | 'In Transit' | 'Delivered' |
'Cancelled';  estimatedDelivery: string;
}
```

[Delivery Zone] export

```
interface DeliveryZone {
_id: string;
zoneName: string;
coverageareas: string[];
shippingCost: number;
carrier/assignedDrivers: string;
}
```

3. Third-Party APIs:

- **Shipment Tracking**:
  Integrate APIs like Shippo to provide real-time tracking updates.
- **Payment Gateways**:
  Use Stripe and Use-Shopping-Cart for payment gateway.

Design System Architecture:

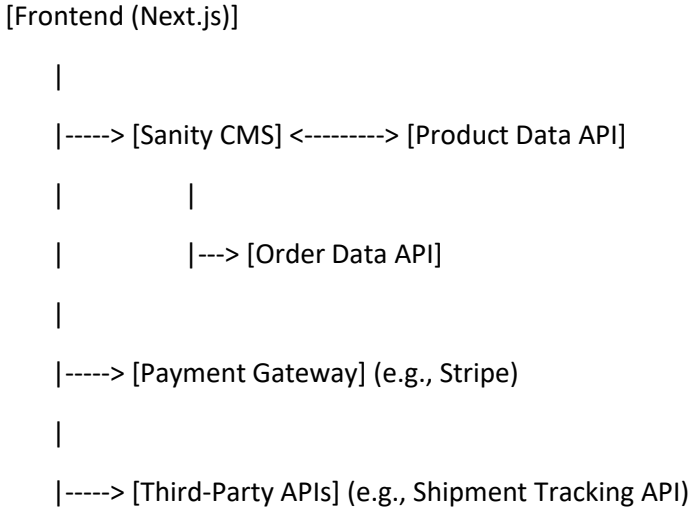
1. Key Workflows:

- **User Registration**
  - **Frontend (Next.js)** → User fills out the registration form
  - **Sanity CMS** → User data is stored in the CMS (e.g., name, email, password).

- **Confirmation** → A confirmation email is sent to the user.

- **Product Browsing**
  - **Frontend (Next.js)** → User browses product categories.
  - **Product Data API (Sanity CMS)** → Fetches product data (e.g.,name, images, descriptions, prices) from Sanity CMS.
  - **Frontend (Next.js)** → Displays product listings dynamically on the website.
- **Order Placement**
  - **Frontend (Next.js)** → User adds items to the cart and proceed to checkout.
  - **Frontend (Next.js)** → Order details (items, quantities, user information) are sent to Sanity CMS.
  - **Sanity CMS** → Stores the order details in the database.
  - **Payment Gateway** → Securely processes the payment and confirms the transaction.
- **Shipment Tracking**
  - **Sanity CMS** → Updates the order with shipping details (e.g., tracking number, carrier).
  - **Third-Party APIs (Shipment Tracking API)** → Fetches real-time shipment status.
  - **Frontend (Next.js)** → Displays shipment status (e.g., "In Transit", "Delivered") to the user.

High-level Diagram:



Plan API Requirements:

1. Endpoint: /products

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

```
[
  {
    "_id": "prod_001",
    "name": "Wooden Coffee Table",
    "description": "A stylish wooden coffee table.",
    "price": 150,
    "images": [
      { "_type": "image", "asset": { "_ref": "image_ref_001", "_type": "reference" } }
    ],
    "dimensions": "120x60x45 cm",
    "material": "Wood",
    "stock": 25,
    "category": "Living Room",
    "tags": ["sale", "new arrival"],
    "reviews": 4.5
  }
]
```


2. Endpoint: /order

- **Method:** POST
- **Description:** Create a new order.
- **Payload:**


```
{
  "productID": "prod_001",
  "Quantity": 2,
  "totalAmount": 300,
  "orderDate": "2025-01-16"
}
```


Response Example:


```
{
  "status": "Success",
  "message": "Order created successfully.",
  "order": {
    "_id": "order_001",
    "productID": "product_001",
    "quantity": 2,
    "totalAmount": 600,
    "orderDate": "2025-01-16"
  }
}
```


4. Endpoint: /payment

- **Method:** POST
- **Description:** Process payment for an order.
- **Payload:**


```
{
  "order": {
    "_id": "order_001",
    "totalAmount": 150
  },
  "paymentMethod": "Credit Card",
  "status": "Completed",
  "transactionId": "txn_001",
  "Amount": 150,
  "paymentDate": "2025-01-16"
}
```


Response Example:


```
{
  "paymentStatus": "Success",
  "transactionId": "txn_001",
  "message": "Payment has been successfully processed."
}
```


5. Endpoint: /shipment •

Method: GET

- **Description**: Track the shipment status for an order.
- **Response Example**:

```
{
  "_id": "ship_001",
  "order": {
    "_id": "order_001",
    "totalAmount": 150
  },
  "trackingNumber": "track_001",
  "status": "In Transit",
  "estimatedDelivery": "2025-01-20"
}
```

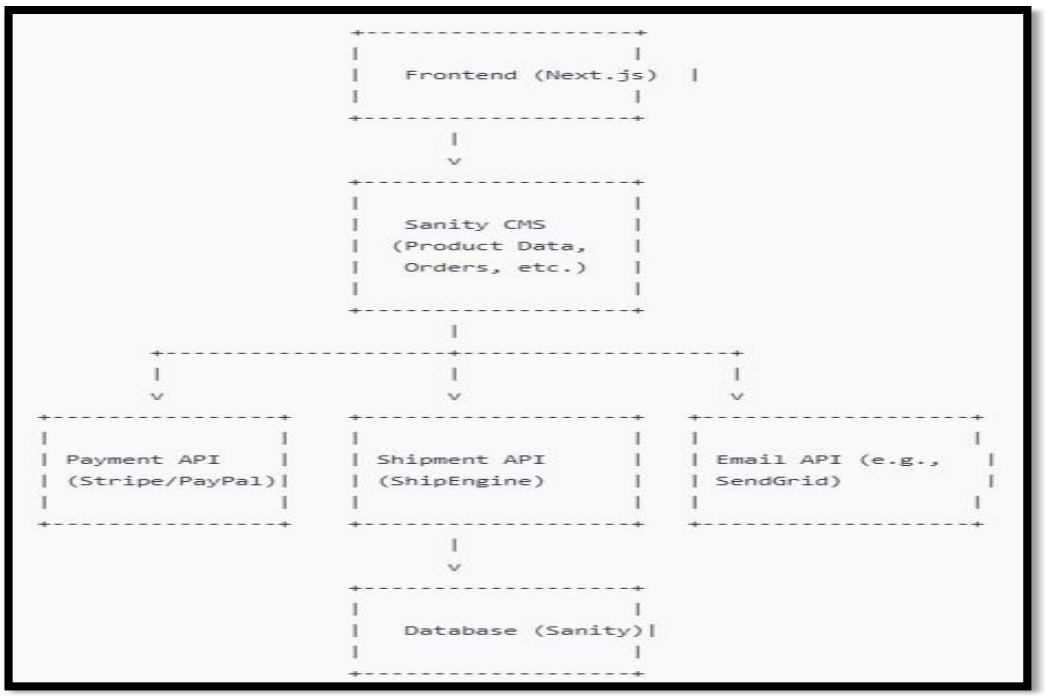
6. Endpoint: /delivery-zone •

Method: GET

- **Description**: Fetch all delivery zones and their details.
- **Response Example**:

```
[
  {
    "_id": "zone_001",
    "zoneName": "Zone 1",
    "coverageareas": ["City A", "City B"],
    "shippingCost": 20,
    "carrier": "Carrier 1",
    "assignedDrivers": "Driver 1, Driver 2"
  }
]
```

Write Technical Documentation:



Explanation of the Diagram:

1. **Frontend (Next.js)**:
    - The user interacts with the frontend, which is built using Next.js. The frontend displays products, manages the cart, and handles the checkout process.
    - It communicates with Sanity CMS to fetch product data and manage orders.
    - It also interacts with third-party APIs for payment processing and shipment tracking.
2. **Sanity CMS**:
    - Sanity acts as the backend database and CMS. It stores product data, customer information, and order details.
    - The frontend makes API requests to Sanity to fetch and display product details, add orders, and track inventory.
3. **Third-Party APIs**:
    - **Payment API** (e.g., Stripe or PayPal): Handles payment transactions when users make purchases.
    - **Shipment API** (e.g., ShipEngine, AfterShip): Tracks the shipment status of orders, including real-time updates on delivery. ○ **Email API** (e.g., SendGrid): Sends email notifications to customers for order confirmations, shipping updates, etc.
4. **Database (Sanity)**:
    - Sanity CMS also functions as a database for storing the products, orders, and other relevant data. It is tightly integrated with the frontend and third-party services.