Market Place Technical Foundation

Introduction:

Our e-commerce platform is designed to offer high-end home decor items, including vases, designer chairs, elegant mirrors, and premium lamps. This document details the technical framework necessary to create a scalable, user-friendly online marketplace.

Define Technical Requirements:

1. Frontened Requirements:

The frontend should deliver an intuitive, visually appealing, and seamless user experience across all devices. The design must prioritize usability, accessibility, and responsiveness to ensure optimal performance on mobile, tablet, and desktop platforms.

Key Features

1.User-Friendly Interface:

- Clean, modern, and visually engaging design that aligns with the brand identity.
- Intuitive navigation with clear calls-to-action (CTAs) to enhance user engagement.

2.Responsive Design:

- Fully responsive layout to ensure consistent functionality and aesthetics across all devices (mobile, tablet, and desktop).
- Optimized performance for fast loading times and smooth browsing.

Essential Pages

1. Home Page

- Hero Section: Display high-quality visuals of featured collections, seasonal trends, and promotional banners.
- **Search Bar:** Prominent and accessible search functionality with auto-suggestions for quick product discovery.
- **Sections to Include:**

i. **Latest Collection:** Highlight new arrivals with eye-catching visuals.

ii. **Top Categories:** Showcase popular product categories for easy navigation.

iii. **Featured Products:** Curate a selection of best-selling or exclusive items.

iv. **Trending Products:** Display currently popular items based on user activity.

2. About Page

- **Brand Story:** A compelling narrative about the brand's origins, mission, and values.
- **Unique Selling Points (USPs):** Clearly communicate what sets the brand apart in the home décor industry.
- **Visual Elements:** Include high-quality images or videos that reflect the brand's aesthetic and ethos.

3. Contact Us Page

- **Contact Form:** A simple yet comprehensive form with fields for name, email, and message.
- **Contact Information:** Display phone numbers, email addresses, and a physical address (if applicable).
- **Google Map Integration:** Embed an interactive Google Map for location visibility.

4. Product Listing/Shop Page

- **Filtering Options:** Allow users to filter products by price, category, color, material, and other relevant attributes.
- **Grid/List View:** Provide the option to switch between grid and list views for better user preference.

5. Product Details Page

- **Product Information:** Display detailed descriptions, including dimensions, materials, variations, and care instructions.
- **High-Quality Media:** Include multiple high-resolution images and a 360-degree view or zoom feature.
- **Customer Reviews and Ratings**: Showcase user-generated reviews and ratings to build trust and credibility.
- **Add to Cart/Wishlist:** Clear CTAs for adding products to the cart or wishlist.

6. Cart Page

- **Cart Summary:** Display a clear breakdown of items, quantities, and prices.

- **Shipping Cost Calculator:** Integrate an estimated shipping cost calculator based on the user's location.
- **Proceed to Checkout:** A prominent CTA to move to the checkout process.

7. Checkout Page

- **Guest Checkout:** Enable users to checkout without creating an account.
- **Secure Payment Gateway:** Integrate trusted payment options with secure fields for card details.
- **Address Validation:** Automatically validate shipping addresses to prevent errors.
- **Order Summary:** Display a concise summary of the order, including items, quantities, and total cost.

8. Order Confirmation Page

- **Order Summary:** Provide a detailed summary of the purchase, including product details, shipping address, and payment method.
- **Delivery Timeframes:** Clearly communicate expected delivery dates.
- **Tracking Details:** Include a tracking number and link to track the order's status.

2.Sanity CMS Schema Design:

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:

[Products]

```
export interface Product {

  _id: string;

  _type: 'product';

  name: string;

  slug: { _type: 'slug'; current: string };

  description: string;

  price: number;
```

```
  images: { _type: 'image'; asset: { _ref: string; _type: 'reference' } }[];
  dimensions: string;
  material: string;
  stock: number;
  category: { _type: 'reference'; _ref: string };
  tags: string[];
  reviews: { _type: 'reference'; _ref: string }[];
}
```

[Order]

```
export interface Order {
  _id: string;
  _type: 'order';
  customer: { _type: 'reference'; _ref: string };
  items: {
    product: { _type: 'reference'; _ref: string };
    quantity: number;
    price: number;
  }[];
  totalAmount: number;
  paymentStatus: 'pending' | 'completed' | 'failed';
  shippingStatus: 'pending' | 'shipped' | 'delivered';
  payment: { _type: 'reference'; _ref: string };
  shipment: { _type: 'reference'; _ref: string };
}
```

[Customer]

```
export interface Customer {
  _id: string;
  _type: 'customer';
  name: string;
  email: string;
  phone?: string;
  address: {
    street: string;
    city: string;
    state: string;
    postalCode: string;
    country: string;
  };
  orders: { _type: 'reference'; _ref: string }[];
}
```

[Payment]

```
export interface Payment {
  _id: string;
  _type: 'payment';
  order: { _type: 'reference'; _ref: string };
  paymentMethod: 'credit_card' | 'paypal' | 'bank_transfer';
  transactionId: string;
  amount: number;
```

```
  status: 'pending' | 'completed' | 'failed';
}
```

[Shippment]

```
export interface Shipment {
  _id: string;
  _type: 'shipment';
  order: { _type: 'reference'; _ref: string };
  trackingNumber: string;
  carrier: string;
  deliveryZone: { _type: 'reference'; _ref: string };
  estimatedDelivery: string;
  status: 'pending' | 'shipped' | 'delivered';
}
```

[Delivery Zone]

```
export interface DeliveryZone {
  _id: string;
  _type: 'deliveryZone';
  name: string;
  regions: string[];
  shippingCost: number;
  estimatedDeliveryTime: string;
  }
```

3. Third-Party APIs:

▪ **Payment Gateways:** Use Stripe and Use-Shopping-Cart for payment gateway.

▪ **Contact Form:** Use an API like FormSpree to handle form submissions and email notifications.

▪ **Shipment Tracking:** Integrate APIs like Shippo and Shipengine to provide real-time tracking updates.

DAY-02:

3. Plan API Requirement

1. Endpoint: /products

Method: Get

Description: fetch all available products

Response example:


```
[
  {
        "product-id": "prod_001",
        "product-name": "The dandy chair",
        "product-price": 199.99,
        "total-stock": 50,
        "image": "https://cdn.sanity.io/images/project-id/dataset/image1.jpg",
        "description": "A stunning Chair with good quality.",
        "reviews": [
        {
        "reviewer": "Alice Smith",
```

```
        "rating": 5,
        "comment": "Absolutely love this",
        "date": "2025-01-01"
        },
        ]
  },
]
```


2.Endpoint Name: /orders

Method: POST

Description: Create a new order in Sanity.

Payload Example:


```
{
  "customer": {
        "name": "John Doe",
        "email": "john.doe@example.com",
        "address": {
        "street": "123 Main St",
        "city": "New York",
        "state": "NY",
        "zip": "10001"
        }
```

},

"products": [

{

"product-id": "prod_001",

"quantity": 2

},

Response Example:

```
{
  "order_id": "order_12345",
  "status": "Order Created",
  "created_at": "2025-01-16T10:30:00Z",
  "total_amount": 499.97
}
```

3.Endpoint Name: /shipment

Method: GET

Description: Track order status via third-party API.

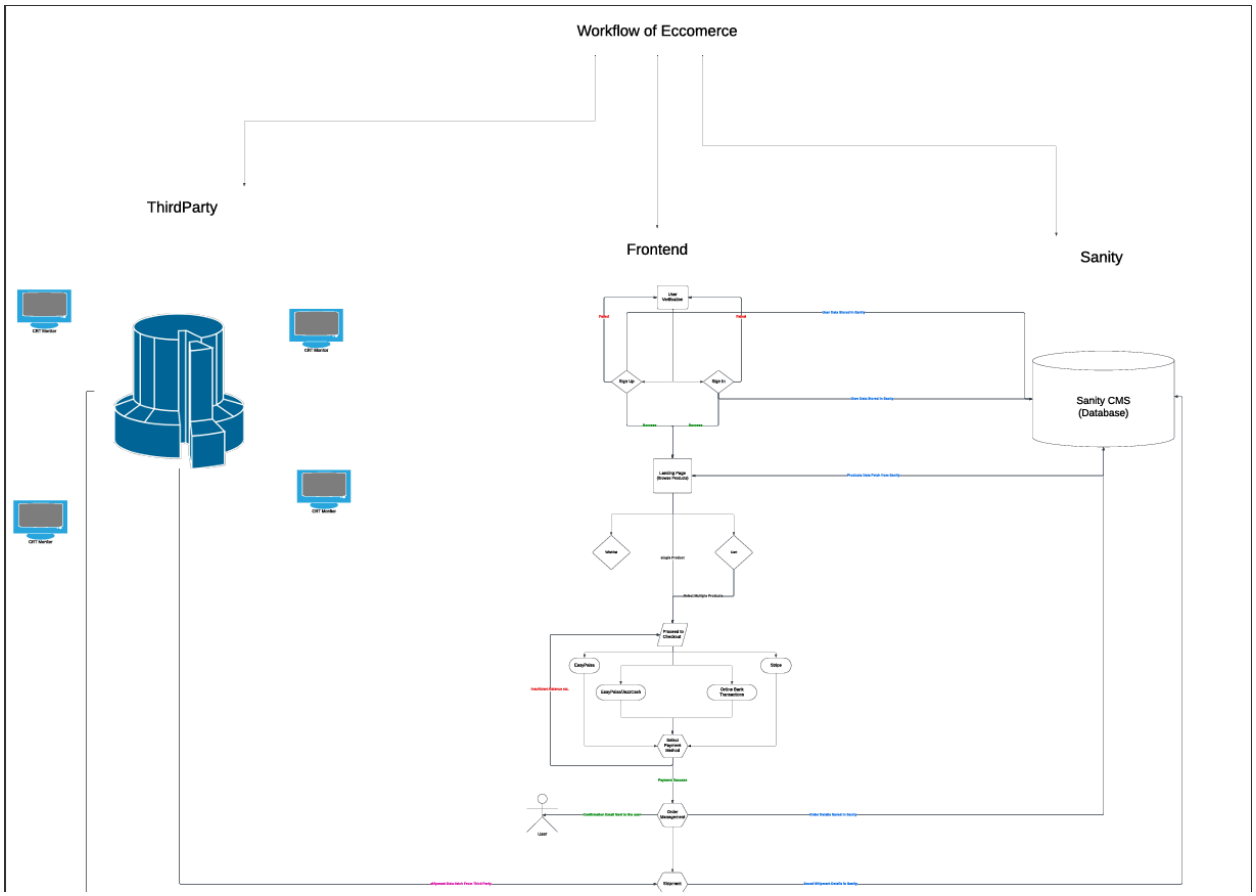
Response Example:

```
{
  "shipment-id": "ship_67890",
  "order-id": "order_12345",
  "status": "In Transit",
  "expected-delivery-date": "2025-01-20T18:00:00Z",
```

"tracking-url": "https://thirdparty.com/track/ship_67890"

}

DAY-03



Day-04

SYSTEM ARCHITECTURE OVERVIEW

COMPONENTS DESCRIPTIONS:

- Frontend (Nextjs. Tailwind CSS):

* Manages user interactions, displays data, and handles routing.
* Communicates with Sanity CMS and APIs for data retrieval and submission.

- SANITY CMS:

* Backend system for managing dynamic content such as products, categories, and inventory.
* Provides a structured schema for storing and retrieving data.

2 KEY WORKFLOWS:

GENERAL WORKFLOWS.

1) User Adds Products to Cart:

* STEP 1: User selects a product and clicks "Add to Cart"
* STEP 2: Frontend sends a POST request to the /cart end point with product details
* STEP 3: Backend updates the cart in the data base and sends a response
* STEP 4: Frontend updates the cart UI

1) User Places AND Orders:

* Step 1: User reviews the cart and clicks checkout

Step 2: Frontend sends a POST request to the /orderss end point with cart details

Date

API's :

Facilitates communication between the frontend and external services.

Examples:

/products for retreiving product listings
/cart for managing cart operations

DATA BASE:

Stores ~~users~~ user-related data (e.g account information, order, history

THIRD PARTI APIs:

Real time inventory updates, payments gateways or delievery status tracking

No.

Step 3 : Backend processes the order updates inventory and returns an order Id

Step 4 User recieves order confirmation

GENERAL ECOMMRCE:

- PRODUCT BROWSING:

* /products. Fetche product listing

- ORDER PLACEMENT WORK FLOW:

* Place order and Payment processed and confirmation sent and inventory updated