Marketplace Technical Foundation - EasyMart

Objective:

Develop an ecommerce platform with a responsive frontend and Sanity CMS backend to manage product data, customer details, and order records.

1. Define Technical Requirements

My marketplace type is **General ECommerce**, where customers can shop for a variety of product categories in one place, ensuring a seamless and diverse shopping experience.

1. **Frontend Requirements:**
   Design a user-friendly and responsive interface for browsing products. Key pages and features include:

   i. **Home Page:**
      - Display featured products, categories, and much more.
      - Include a search bar for quick access.

   ii. **Product Listing Page:**
      - Show a grid or list of products with essential details (image, name, price).
      - Implement sorting and filtering options (e.g., price, category).

   iii. **Product Details Page:**
      - Detailed information about a selected product (description, specifications, price, reviews).
      - Add "Add to Cart" button.

   iv. **Cart Page:**
      - List selected products.
      - Show subtotal and total price.
      - Include a "Proceed to Checkout" button.

   v. **Checkout Page:**
      - Collect user information (name, email, shipping address).
      - Integrate payment gateway for order processing.

   vi. **Order Confirmation Page:**
      - Show order details, estimated delivery time, and order number.

vii. **Responsive Design:**
   - Ensure the design adapts to different screen sizes (desktop, tablet, mobile).
   - Use CSS frameworks (e.g., Bootstrap, Tailwind) or custom CSS for styling.

2. **Sanity CMS as Backend:**

   Sanity CMS will serve as the database for managing ecommerce data. Key tasks include:

i. **Design Schemas in Sanity:**
   Create schemas to align with business requirements. Suggested schemas:
   - **Product Schema:** Fields for name, description, price, images, categories, stock status.
   - **Category Schema:** Fields for name and description.
   - **Order Schema:** Fields for order ID, customer details, products purchased, and order status.
   - **Customer Schema:** Fields for customer name, email, address, and order history.

ii. **Set Up Sanity Studio:**
   - Install and configure Sanity Studio to manage product data.
   - Customize the dashboard for ease of use.

iii. **Connect Sanity to Frontend:**
   - Use Sanity's APIs to fetch data dynamically.
   - Implement CRUD operations for products, categories, and orders.

2. Design System Architecture

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

Component Descriptions

1. **Frontend (Next.js):**
    - The user interface where users browse products, add items to the cart, and place orders.
    - It interacts with the backend (Sanity CMS) to fetch and display product details dynamically.
2. **Sanity CMS:**
    - Acts as the backend for managing product data, customer information, and order records.
    - Stores structured data (e.g., products, orders, customers) and provides APIs for data retrieval and updates.
3. **Product Data API (Sanity CMS API):**
    - Serves as the communication layer between the frontend and Sanity CMS.
    - Handles requests for fetching product listings, categories, and order details.
4. **Third Party API (Shipment Tracking API):**
    - Integrates with logistics services to fetch real-time shipment tracking updates.
    - Provides status updates to users about their order delivery.

5. **Payment Gateway:**
    - Processes user payments securely.
    - Handles sends payment confirmation to the frontend, and records order payment details in Sanity CMS.

Key Workflows:

1. **User Registration**
    - User signs up on the frontend.
    - User data (name, email, password) is sent to Sanity CMS via API.
    - Sanity stores the user data and sends a confirmation to the frontend.

2. **Product Browsing**
    - User visits the product listing page.
    - Frontend sends a request to the Product Data API.
    - Sanity CMS retrieves product data and sends it to the frontend.
    - Products are displayed dynamically, with options to sort and filter.

3. **Order Placement**
    - User adds items to the cart and proceeds to checkout.
    - Frontend sends order details (product IDs, quantity, customer info) to Sanity CMS.
    - Sanity stores the order record and updates inventory status.
    - Payment is processed via the Payment Gateway.
    - Order confirmation is displayed to the user and saved in Sanity.

4. **Shipment Tracking**
    - Sanity CMS records the order tracking ID after payment confirmation.
    - Frontend sends a request to the Third-Party API for shipment updates.
    - Realtime tracking information is fetched and displayed to the user.

3. Plan API Requirements

1. **Endpoint Name:** /products
   - **Method:** GET
   - **Description:** Fetch all product details.
   - **Response Example:**
     ```
     [
       { "id": 1, "name": "Product A", "price": 100 },
       { "id": 2, "name": "Product B", "price": 150 }
     ]
     ```

2. **Endpoint Name:** /order
   - **Method:** POST
   - **Description:** Create a new order with the provided details.
   - **Response Example:**
     ```
     {
     "customerId": 101,
       "products": [
         { "productId": 1, "quantity": 2 },
         { "productId": 2, "quantity": 1 }
       ],
       "totalPrice": 350
     }
     ```

3. **Endpoint Name:** /customer
   - **Method:** GET
   - **Description:** Fetch customer details by ID.
   - **Response Example:**
     ```
     { "id": 101, "name": "arsalan", "email": " arsalan@example.com" }
     ```

4. **Endpoint Name:** /payment
   - **Method:** POST
   - **Description:** Process a payment for an order.
   - **Response Example:**
     ```
     { "paymentId": 98765, "status": "Success", "amount": 350 }
     ```

5. **Endpoint Name:** /shipment
   - **Method:** GET
   - **Description:** Fetch shipment tracking details by order ID.
   - **Response Example:**
     ```
       "orderId": 12345,
       "trackingId": "TRACK123",
       "status": "In Transit",
       "estimatedDelivery": "2025-01-25"
     }
     ```

6. **Endpoint Name:** /zone
   - **Method:** GET
   - **Description:** Fetch available shipping zones or regions.
   - **Response Example:**
     ```
     [
       { "zoneId": 1, "name": "Karachi" },
       { "zoneId": 2, "name": "Lahore" }
     ]
     ```

4. Schema Example

```
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'price', type: 'number', title: 'Price'},
    { name: 'stock', type: 'number', title: 'Stock Level'},
    { name: 'description', type: 'text', title: 'Product Description'},
    { name: 'image', type: 'image', title: 'Product Image', options: { hotspot: true } },
  ],
};
```