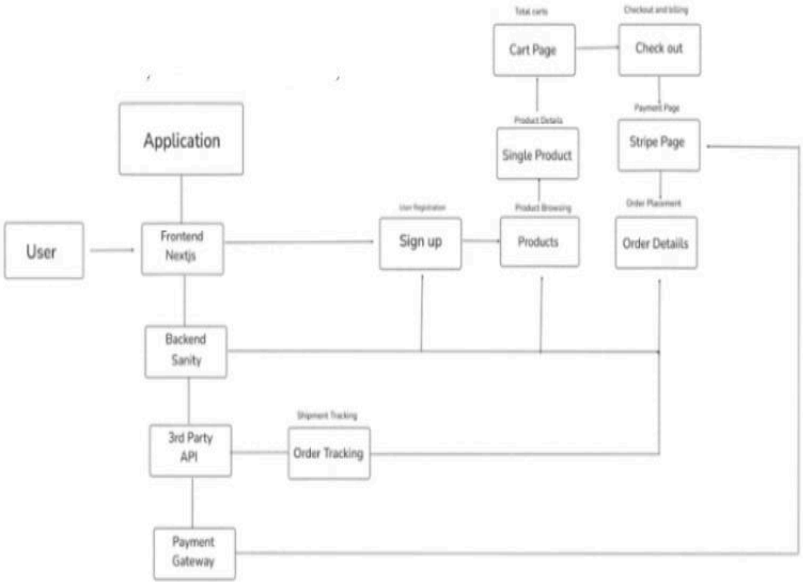
Marketplace Technical Foundation - General Ecommerce

Overview:

This document details the system architecture, key workflows, and API requirements for the Furniture E-Commerce Marketplace. It serves as a foundational guide for the implementation phase.

1. System Architecture Diagram



2.Component Description

Here's a breakdown of the components and their descriptions for your e-commerce application:

Frontend (Next.js)

1. Home Page

Displays featured products, banners, and categories.

Navigation to other sections like Products, Sign Up, and Cart.

2. Single Product Page

Displays detailed information about a specific product.

Includes:

Product name, description, price, and images.

"Add to Cart" and "Buy Now" buttons.

Related products section.

3. Products Page

Lists all products with filters and sorting options (e.g., by price, category).

Pagination for large product lists.

4. Cart Page

Displays products added to the cart with their quantities, prices, and total cost.

Allows users to update quantities or remove items.

Button to proceed to checkout.

5. Checkout Page

Form to collect shipping and billing details.

Summary of the cart and total price.

Integration with Stripe for payment processing.

6. Order Tracking Page

Allows users to track their orders using an order ID.

Displays the current status (e.g., Processing, Shipped, Delivered).

7. Order Details Page

Shows a summary of a specific order.

Includes:

Order ID.

Product details.

Shipping address and order status.

8. Sign Up / Login Page

Sign-up form to register users with email and password.

Login form for returning users.

9. Stripe Payment Page

Secure page for processing payments using the Stripe API.

Displays payment confirmation upon success.

Backend (Sanity CMS)

1. Products Schema

Stores product details such as name, description, price, images, categories, and stock availability.

2. Orders Schema

Tracks order details, including:

Order ID.

Products purchased.

Customer information.

Order status and timestamps.

3. Users Schema

Stores user information like email, password (hashed), and order history.

4. Categories Schema

Manages product categories for filtering.

3rd Party API

API for Traditional and Cultural Dresses:

Fetches product data (e.g., styles, designs, pricing).

Ensures your marketplace has a variety of options for users.

Payment Gateway (Stripe)

Used for secure payment processing.

Supports multiple payment methods (e.g., credit/debit cards, wallets).

Integrates with the Checkout and Stripe Payment Page.

Application Components

1. Header Component

Contains navigation links (Home, Products, Cart, Login/Sign Up).

Responsive design for mobile and desktop views.

2. Footer Component

Displays copyright information, social media links, and contact details.

3. Product Card Component

Reusable component to display individual product details in lists.

4. Cart Item Component

Displays individual items in the cart with quantity controls.

5. Order Summary Component

Shows a summary of the order on the checkout and order details pages.

6. Filter Component

Allows users to filter products by category, price, or rating.

7. Search Bar Component

Enables users to search for products by name or keyword.

8. Pagination Component

Handles navigation between pages of product listings.

9. Modal Component

Used for pop-ups like confirming an action (e.g., removing an item from the cart).

10. Order Tracking Status Component

Visual representation of the order's progress.

—

3.Key Workflows

1. User Registration Workflow

Goal: Allow users to create accounts and manage profiles securely.

Steps:

1. Sign-Up Process:

User fills out a registration form with details like name, email, and password.

Client-side validation ensures required fields are filled and meet criteria (e.g., email format, password strength).

2. Data Storage in Sanity:

After submission, the form data is sent to the backend API.

The backend hashes the password for security and stores the user details in Sanity CMS under a users collection/document.

3. Confirmation Email:

A confirmation email is sent to the user using a service like SendGrid or Nodemailer.

The email contains a verification link or code for account activation.

4. Account Activation:

When the user clicks the link, the backend verifies the token and updates the user status in Sanity (e.g., isVerified: true).

User is redirected to the login page or dashboard after successful activation.

---

2. Product Browsing Workflow

Goal: Provide users with an intuitive way to explore and view product categories and details.

Steps:

1. Category Selection:

Users navigate to the homepage or category menu to choose a product category (e.g., Traditional Dresses).

2. Fetching Data from Sanity:

The frontend sends a request to the Sanity API to fetch products based on the selected category.

Data includes product details such as name, price, images, and descriptions.

3. Frontend Display:

The fetched data is displayed in a grid or list format with filtering and sorting options (e.g., price, popularity).

Clicking on a product redirects the user to a detailed product page with additional information and an "Add to Cart" button.
---

3. Order Placement Workflow

Goal: Enable users to add items to their cart, proceed to checkout, and save order details.

Steps:

1. Adding to Cart:

Users select a product, choose quantity or variations (e.g., size, color), and click "Add to Cart."

The cart is managed on the frontend (e.g., local state or context).

2. Checkout Process:

Users review their cart, provide shipping details, and select a payment method.

The checkout form validates inputs (e.g., address, payment details).

3. Order Storage in Sanity:

Upon successful payment, the order details (user ID, product IDs, quantity, total amount, and shipping info) are sent to the backend.

The backend creates a new orders document in Sanity CMS, storing all order details for future reference.

4. Order Confirmation:

A confirmation message or email is sent to the user with order details and tracking information.

---

4. Shipment Tracking Workflow

Goal: Provide real-time updates on order status using a 3rd-party API.

Steps:

1. Fetching Order Status:

The backend integrates with a shipping API (e.g., Shippo, EasyPost) to track shipments using the order's tracking ID.

The API fetches updates like "Order Shipped," "In Transit," or "Delivered."

2. Updating Order Status in Sanity:

The fetched status is updated in the corresponding orders document in Sanity CMS under a field like shipmentStatus.

3. Frontend Display:

Users log in to their account and navigate to the "My Orders" section.

The frontend fetches order details, including the latest shipment status, from Sanity API and displays it in an intuitive format (e.g., timeline or status bar).

4. Notifications:

Users receive email or in-app notifications for significant updates, like when the order is shipped or delivered.

4.API Endpoints

1. Products

Endpoint Name: /products

Method: GET

Description: Fetch all product details.

Response Example:


```
[
 {
  "id": 1,
  "name": "Modern Sofa",
  "category": "Living Room",
  "price": 25000,
  "description": "A comfortable modern sofa with premium fabric.",
  "image": "sofa.jpg",
  "stock": 15
 },
 {
  "id": 2,
  "name": "Wooden Dining Table",
  "category": "Dining Room",
  "price": 45000,
```

```
   "description": "A six-seater wooden dining table.",
   "image": "dining_table.jpg",
   "stock": 10
 }
]
```


2. Single Product

Endpoint Name: /products/:id

Method: GET

Description: Fetch details of a specific product by ID.

Response Example:


```
{
 "id": 1,
 "name": "Modern Sofa",
 "category": "Living Room",
 "price": 25000,
 "description": "A comfortable modern sofa with premium fabric.",
 "image": "sofa.jpg",
 "stock": 15
}
```


3. Categories

Endpoint Name: /categories

Method: GET

Description: Fetch all product categories.

Response Example:


```
[
 {
   "id": 1,
```

```
  "name": "Living Room"
 },
 {
  "id": 2,
  "name": "Bedroom"
 },
 {
  "id": 3,
  "name": "Dining Room"
 }
]
```


4. Products by Category

Endpoint Name: /categories/:id/products

Method: GET

Description: Fetch all products under a specific category.

Response Example:


```
[
 {
  "id": 3,
  "name": "King Size Bed",
  "category": "Bedroom",
  "price": 55000,
  "description": "A spacious king-size bed with a sturdy frame.",
  "image": "king_bed.jpg",
  "stock": 8
 }
]
```


5. Search Products

Endpoint Name: /products/search

Method: GET

Query Parameters: ?q=<search-term>

Description: Search for products by name or description.

Response Example:

```
[
  {
    "id": 4,
    "name": "Luxury Office Chair",
    "category": "Office",
    "price": 15000,
    "description": "An ergonomic office chair for long working hours.",
    "image": "office_chair.jpg",
    "stock": 20
  }
]
```

6. Add Product (Admin Only)

Endpoint Name: /products

Method: POST

Description: Add a new product to the catalog.

Request Body Example:

```
{
  "name": "Coffee Table",
  "category": "Living Room",
  "price": 12000,
  "description": "A sleek coffee table with a glass top.",
  "image": "coffee_table.jpg",
  "stock": 10
}
```

Response Example:

{
"message": "Product added successfully.",
"productId": 5
}

7. Update Product (Admin Only)

Endpoint Name: /products/:id

Method: PUT

Description: Update details of an existing product.

Request Body Example:

{
"name": "Luxury Sofa",
"price": 28000,
"stock": 12
}

Response Example:

{
"message": "Product updated successfully."
}

8. Delete Product (Admin Only)

Endpoint Name: /products/:id

Method: DELETE

Description: Remove a product from the catalog.

Response Example:

{
"message": "Product deleted successfully."

```
}
```

9. Cart

Endpoint Name: /cart

Method: GET

Description: Fetch items in the user's cart.

Response Example:

```
[
  {
    "productId": 1,
    "name": "Modern Sofa",
    "price": 25000,
    "quantity": 1
  }
]
```

10. Add to Cart

Endpoint Name: /cart

Method: POST

Description: Add an item to the user's cart.

Request Body Example:

```
{
  "productId": 1,
  "quantity": 1
}
```

Response Example:

```
{
```

```
  "message": "Item added to cart."
}
```


11. Checkout

Endpoint Name: /checkout

Method: POST

Description: Process the user's order.

Request Body Example:


```
{
  "cart": [
    {
      "productId": 1,
      "quantity": 1
    }
  ],
  "address": "123 Main Street, Karachi",
  "paymentMethod": "Credit Card"
}
```


Response Example:


```
{
  "message": "Order placed successfully.",
  "orderId": 101
}
```


—

API ENDPOINT CHART

END POINT	METHOD	DESCRIPTION
/api/cars	Get	Get list of availabe cars for rental.
/api/cars/:id	Get	Get details of a specific car.
/api/Bookings	Post	Book a car for rental.
/api/Booking	Get	Get a list of user bookings for rental.
/api/cancel booking	Delete	Cancel a car booking.
/api/Payments	Post	Process payment for booking.

5.Sanity Schema:

```
export default {
  name: 'bar',
  type: 'document',
  title: 'Bar Furniture',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Product Name',
      description: 'Enter the name of the bar furniture item',
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
      options: {
        list: ['Bar', 'Bar Stool', 'Bar Cabinet', 'Bar Cart'], // Add more bar-related categories
if needed
        layout: 'dropdown',
      },
      description: 'Select the category of the bar furniture item',
    },
    {
```

```
    name: 'price',
    type: 'number',
    title: 'Price',
    description: 'Enter the price of the bar furniture item',
  },
  {
    name: 'stock',
    type: 'number',
    title: 'Stock Level',
    description: 'Enter the number of items in stock',
  },
  {
    name: 'dimensions',
    type: 'object',
    title: 'Dimensions',
    fields: [
      { name: 'width', type: 'number', title: 'Width (in cm)' },
      { name: 'height', type: 'number', title: 'Height (in cm)' },
      { name: 'depth', type: 'number', title: 'Depth (in cm)' },
    ],
    description: 'Enter the dimensions of the bar furniture item',
  },
  {
    name: 'material',
    type: 'string',
    title: 'Material',
    description: 'Enter the material of the bar furniture item (e.g., wood, metal)',
  },
  {
    name: 'capacity',
    type: 'number',
    title: 'Capacity (No. of Bottles)',
    description: 'Enter the number of bottles the bar furniture can hold (if applicable)',
  },
  {
    name: 'features',
    type: 'array',
    title: 'Special Features',
    of: [{ type: 'string' }],
    description: 'Add any special features of the bar furniture (e.g., adjustable height,
built-in storage)',
```

```
    },
{
     name: 'image',
     type: 'image',
     title: 'Product Image',
     options: {
       hotspot: true, // Enable image cropping
     },
    },
    {
     name: 'description',
     type: 'text',
     title: 'Description',
     description: 'Enter a detailed description of the bar furniture item',
    },
  ],
};
```

6.Goals and Achievements

1. Technical Blueprint: Defined all foundational components for development.

2. API Documentation: Outlined endpoints for seamless backend/frontend integration.

3. Alignment with Business Goals: Ensured workflows support both customer experience and operational scalability.