

Hackathon Day 4: Dynamic Marketplace Components and Functionalities Documentation

This document presents a comprehensive analysis of the dynamic features of a marketplace project. It emphasizes modularity, reusability, and seamless integration with Sanity CMS, offering detailed insights into each functionality.

Step 1: Functionalities Overview

The project incorporates the following essential features:

1. **Product Listing Page**
2. **Dynamic Routes**
3. **Cart Functionality**
4. **Checkout Process**
5. **Search Products**
6. **Pagination**
7. **Clerk Auth Integration**

Each feature plays a vital role in creating a responsive, scalable, and user-friendly marketplace.

Step 2: Detailed Functionalities

1. Product Listing Page

The product listing page serves as the central interface for showcasing available products. Products are fetched dynamically from Sanity CMS and displayed in an aesthetically pleasing grid or list format. Additionally, a **View Products** button is provided, which leverages dynamic routes to navigate users to detailed product pages seamlessly.

- **Dynamic Data Fetching:** Product information is retrieved through efficient API calls to Sanity CMS.
- **User Interface Options:** Support for both grid and list layouts to enhance user experience.

```

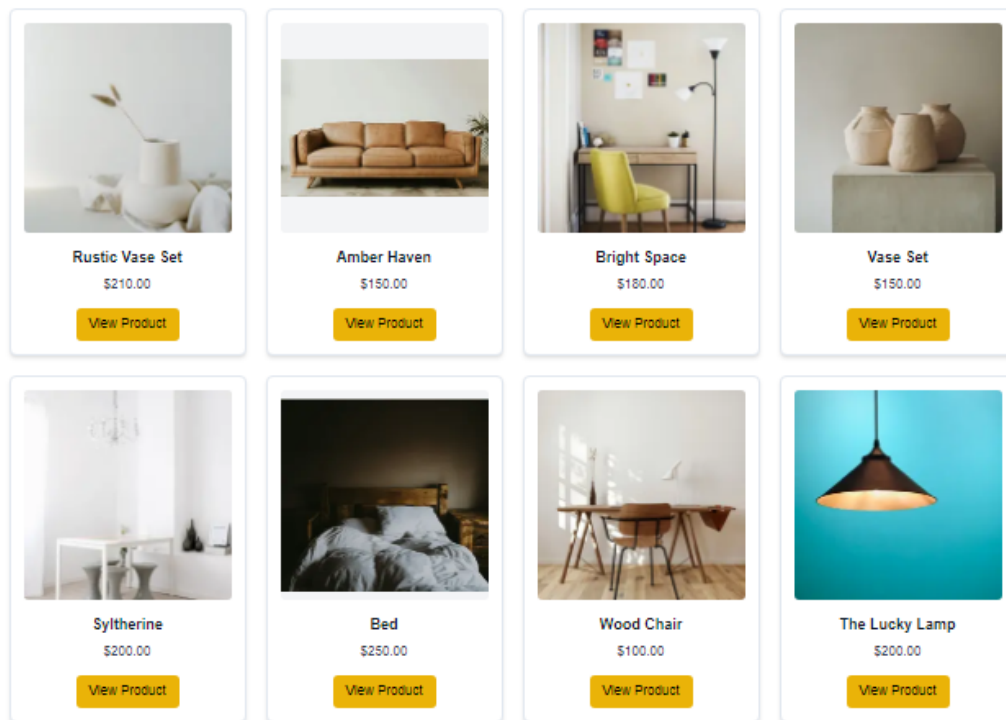
'use client';
import { useEffect, useState } from "react";
import Link from "next/link";
import { sanityFetch } from "@sanity/lib/fetch";
import Image from "next/image";
import SearchAndFilter from "@components/SearchAndFilter";

type Product = {
  _id: string;
  title: string;
  price: number;
  imageUrl: string;
};

async function fetchProducts() {
  const query = `*[_type == "product"]{
    _id,
    title,
    price,
    "imageUrl": productImage.asset->url
  }`;
  const products: Product[] = await sanityFetch({ query });
  return products;
}

```

Our Products List



2. Dynamic Routes

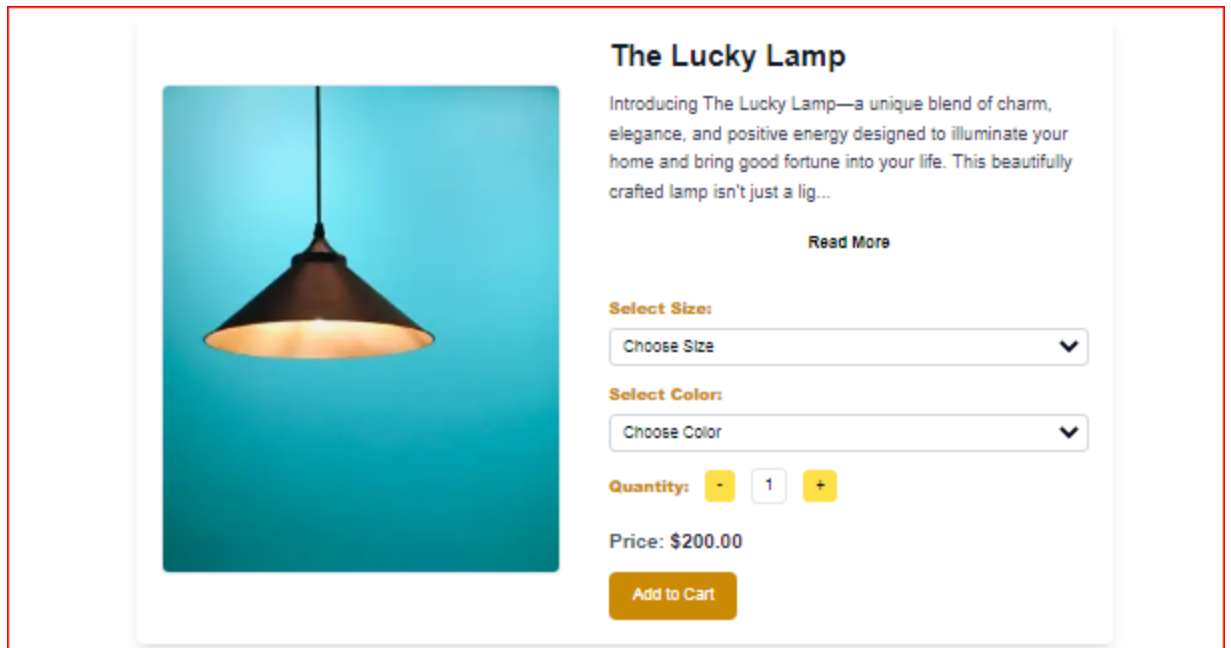
.Product Details: Displays essential product information including:

- **Name:** Title of the product.
- **Description:** Detailed information about product features.
- **Size:** Available size options for the product.
- **Color:** Color variants of the product.
- **Quantity:** Stock availability.
- **Add to Cart:** Option to select and add the product directly to the cart for purchase.

```

c:\> app > productdetails > [id] > 10 page.tsx > ProductDetailsPage > availableColors
39 const [product, setProduct] = useState<Product | null>(null);
40 const [loading, setLoading] = useState(true);
41 const [showFullDescription, setShowFullDescription] = useState(false);
42 const [selectedSize, setSelectedSize] = useState<string | null>(null);
43 const [selectedColor, setSelectedColor] = useState<string | null>(null);
44 const [quantity, setQuantity] = useState<number>(1);
45
46 const availableSizes = ["Small", "Medium", "Large", "Extra Large"];
47 const availableColors = ["Black", "Brown", "Silver", "Blue", "White"];
48
49 useEffect(() => {
50   if (id) {
51     const fetchProductDetails = async () => {
52       const fetchedProduct = await getProductDetails(id as string);
53       setProduct(fetchedProduct);
54       setLoading(false);
55     };
56     fetchProductDetails();
57   }
58 }, [id]);
59
60 const toggleDescription = () => setShowFullDescription(!showFullDescription);
61
62 const handleQuantityChange = (operation: "increment" | "decrement") => {
63   setQuantity((prev) => (operation === "increment" ? prev + 1 : prev > 1 ? prev - 1 : 1));
64 };
65
66 const handleAddToCart = () => {
67   if (!product) return;
68   if (!selectedSize || !selectedColor) {
69     alert("Please select size and color before adding to cart.");
70     return;
71   }
72
73   const newItem: CartItem = {
74     id: product._id,
75     title: product.title,
76     price: product.price,
77     imageUrl: product.imageUrl,
78     size: selectedSize,
79     color: selectedColor,
80     quantity,
81   };

```



3. Cart Functionality

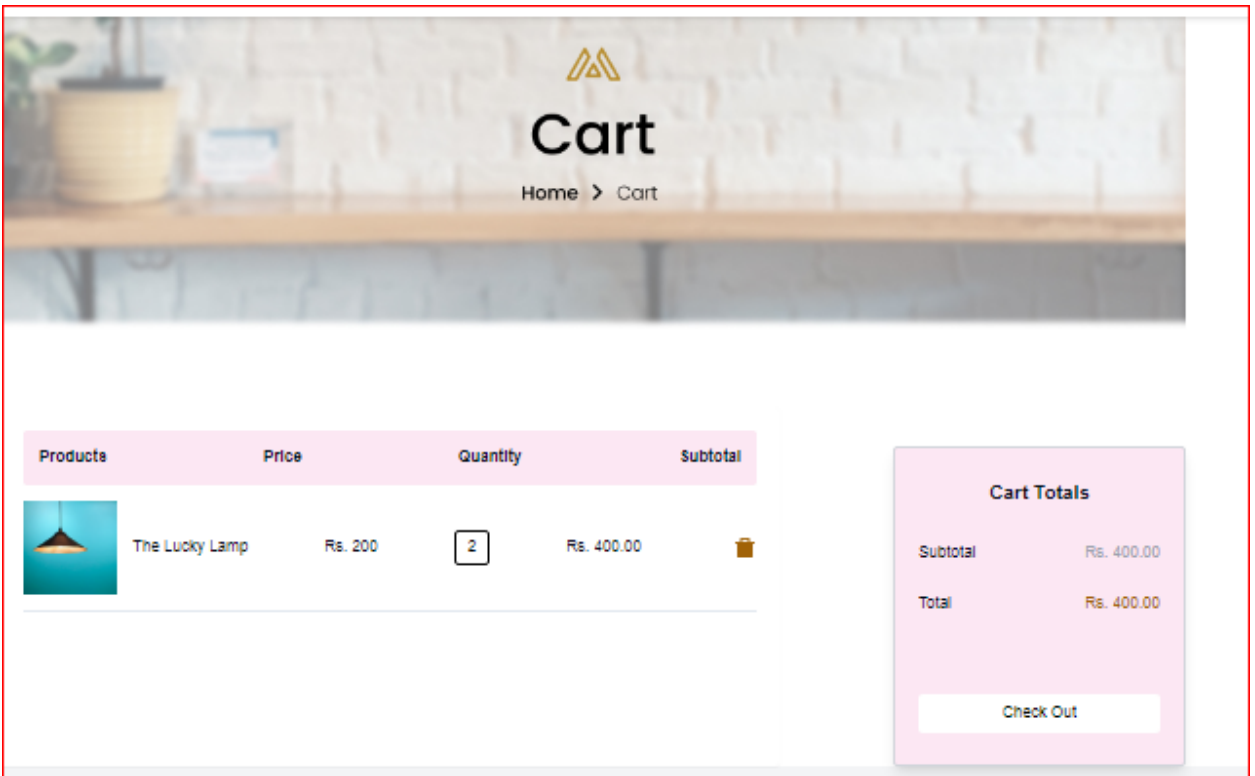
The cart feature allows users to select, review, and modify products before purchasing.

- **Add, Delete, and Subtotal:** Flexible controls for managing cart contents.
- **Persistent Cart:** Cart data is retained across sessions using local storage.

```

1  "use client";
2  import { useCart } from "@/components/CartContext"; // Import CartContext
3  import { RiDeleteBin7Fill } from "react-icons/ri";
4  import Link from "next/link";
5
6  export default function CartPage() {
7    const { cart, removeFromCart } = useCart(); // Accessing cart and removeFromCart method from context
8
9    return (
10     <div className="mx-auto max-w-7xl px-4 sm:px-6 lg:px-8">
11       {/* Cart image */}
12       
17
18       <div className="flex flex-wrap justify-between mt-10 gap-8">
19         {/* Cart products list */}
20         <div className="flex flex-col bg-white px-6 py-6 w-full lg:w-[800px] rounded shadow-sm mt-10">
21           <div className="sm:flex justify-between font-bold mb-4 bg-pink-100 px-4 py-4 rounded grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4">
22             <div>Products</div>
23             <div>Price</div>
24             <div>Quantity</div>
25             <div>Subtotal</div>
26           </div>
27
28           {cart.length === 0 ? (
29             <div className="text-center py-6 text-xl bg-white px-6">Your cart is empty.</div>
30           ) : (
31             cart.map((item) => (
32               <div key={item.id} className="flex flex-wrap sm:flex-nowrap justify-between items-center bg-white mb-4 border-b pb-4">
33                 {/* Product Name and Image */}
34                 <div className="flex flex-col sm:flex-row items-center w-full sm:w-auto mb-4 sm:mb-0">
35                   <img src={item.imageUrl} alt={item.title} className="w-20 h-20 sm:w-24 sm:h-24 mr-4" />
36                   <h3 className="text-sm sm:text-base">{item.title}</h3>
37                 </div>
38
39                 {/* Price */}
40                 <div className="flex flex-col items-start sm:items-center mb-4 sm:mb-0">
41                   <h3 className="text-sm sm:text-base">Rs. {item.price}</h3>
42                 </div>
43
44                 {/* Quantity */}
45                 <div className="flex flex-col items-start sm:items-center mb-4 sm:mb-0">
46                   <span className="px-3 py-1 bg-white text-black rounded border border-black">
47                     {item.quantity}
48                   </span>
49                 </div>
50             )

```




4. Checkout Process

A streamlined and secure checkout process ensures a smooth user journey from cart to payment.

- **User Information Form:** A comprehensive form that collects key details such as:
 - **Name:** Full name of the customer.
 - **Address:** Complete shipping address.
 - **City:** City for product delivery.
 - **Contact Information:** Email and phone number fields.
- **Payment Methods:**
 - **Bank Payment:** Secure bank transfer option.
 - **Cash on Delivery:** Convenient cash payment at the time of delivery.
- **Subtotal Calculation:** Automatic calculation of the total price based on cart contents.
- **Order Review and Confirmation:** A final summary of order details before placing the order.
- **Order Confirmation:** Real-time feedback upon successful purchase.

```
src > app > Checkout > 78 page.tsx > ...
1  'use client';
2
3  import { useEffect, useState } from 'react';
4  import Image from 'next/image';
5  import { useSearchParams } from 'next/navigation';
6  import Shipping from '@components/Shipping';
7
8  interface CartItem {
9    id: string;
10   title: string;
11   quantity: number;
12   price: number;
13 }
14
15 export default function CheckoutPage() {
16   const [cart, setCart] = useState<CartItem[]>([]);
17   const [paymentMethod, setPaymentMethod] = useState<string>('cash'); // Default to cash on delivery
18   const searchParams = useSearchParams();
19
20   useEffect(() => {
21     // Parse cart data from query params
22     const cartParam = searchParams.get('cart');
23     if (cartParam) {
24       const cartData = JSON.parse(cartParam);
25       setCart(cartData);
26     }
27   }, [searchParams]);
28
29   const handleSubmit = (e: React.FormEvent) => {
30     e.preventDefault();
31     if (paymentMethod === 'bank') {
32       alert('Proceeding with Bank Transfer');
33     } else {
34       alert('Proceeding with Cash on Delivery');
35     }
36   };
37 }
```



Checkout

Home > Checkout

Billing details

First Name

Last Name

Company Name (Optional)

Country / Region

Select your country

Billing Address

Town / City

Province

Select your province

ZIP Code

Phone

Email Address

Additional Information

Choose Payment Method

☐ Direct Bank Transfer

☒ Cash On Delivery

Place order

Product

The Lusty Lamp H2

Subtotal

No. 400.00

Estimated

No. 400.00

Total

No. 400.00

☐ Cash On Delivery

You can pay for your order when it arrives at your doorstep.

5. Search Products

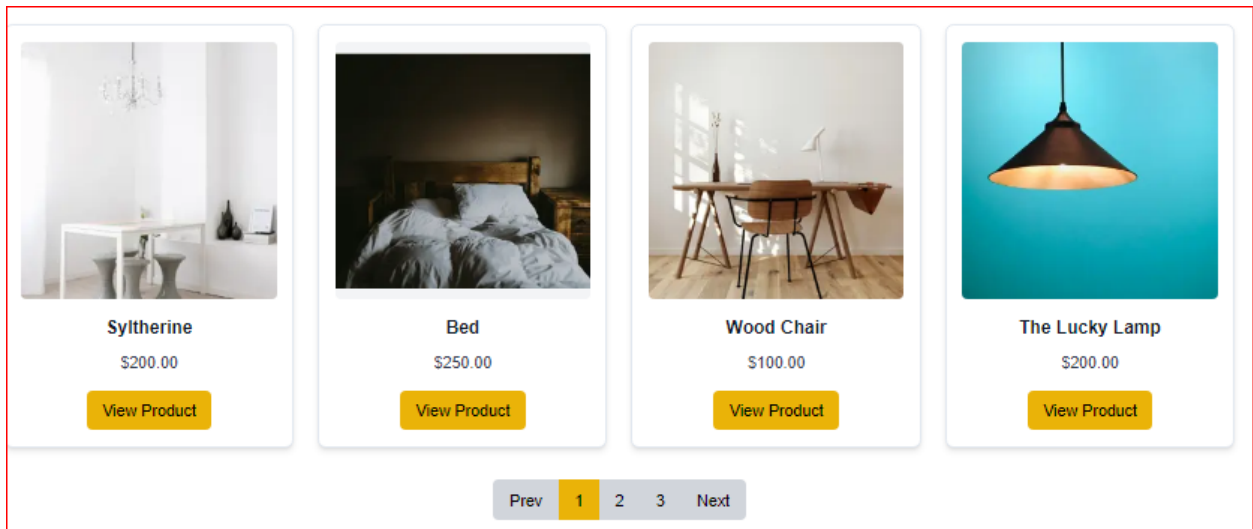
This feature allows users to quickly find specific products by entering keywords.

- **Instant Results:** Products are filtered dynamically as the user types.
- **Search Filters:** Options to refine search results based on categories, price range, and ratings.
- **View All Products:** Clicking the "View All Products" button displays the complete list of products.

6. Pagination

Pagination improves the user experience by breaking product lists into manageable pages.

- **Efficient Navigation:** Users can move between pages seamlessly.
- **Dynamic Page Loading:** Optimized to load only the required data for each page.



```

133
134 /* Pagination Section */
135 <div className="flex justify-center mt-8">
136 <button
137   onClick={handlePrevPage}
138   className="px-4 py-2 bg-gray-300 text-black rounded-l-md hover:bg-gray-400 transition duration-300"
139   disabled={currentPage === 1}
140 >
141   Prev
142 </button>
143
144 /* Show pages 1, 2, 3 dynamically */
145 {totalPages >= 1 && (
146   <button
147     onClick={() => handlePageChange(1)}
148     className={`px-4 py-2 ${currentPage === 1 ? 'bg-yellow-500' : 'bg-gray-300'} text-black hover:bg-gray-400 transition duration-300`}
149     >
150     1
151 </button>
152   )
153
154 {totalPages >= 2 && (
155   <button
156     onClick={() => handlePageChange(2)}
157     className={`px-4 py-2 ${currentPage === 2 ? 'bg-yellow-500' : 'bg-gray-300'} text-black hover:bg-gray-400 transition duration-300`}
158     >
159     2
160 </button>
161   )
162
163 {totalPages >= 3 && (
164   <button
165     onClick={() => handlePageChange(3)}
166     className={`px-4 py-2 ${currentPage === 3 ? 'bg-yellow-500' : 'bg-gray-300'} text-black hover:bg-gray-400 transition duration-300`}
167     >
168     3
169 </button>
170   )
171
172 <button
173   onClick={handleNextPage}
174   className="px-4 py-2 bg-gray-300 text-black rounded-r-md hover:bg-gray-400 transition duration-300"
175   disabled={currentPage === totalPages}
176 >
177   Next
178 </button>
179 </div>
180 </div>
181 </div>
182 );
183 }
184

```

7. Clerk Auth Integration

Clerk authentication ensures secure user access and management.

- **User Sign-In/Sign-Up:** Secure authentication flow with email or social login options.
- **Session Management:** Persistent and secure session handling.
- **Role-Based Access:** Custom roles for admins and users to control access.
- **Dashboard Navigation:** Includes a "Dashboard" button that redirects users to the main dashboard upon clicking.

**Welcome to Furniro
Furniture Shop**

Furniro offers a wide range of high-quality furniture to enhance your home and office space. Discover stylish, durable, and affordable designs tailored to your needs

Login

Email Address
sumbuljawed45@gmail.com

Password

[Forgot Password?](#)

Sign In

Conclusion

This documentation highlights the key functionalities implemented to build a dynamic, responsive, and scalable marketplace. Each feature has been carefully designed for efficiency, user experience, and seamless integration with Sanity CMS. By focusing on modularity and reusability, the project provides a robust foundation for future enhancements and growth.

Prepared by sumbul jawed

Slot: sunday 2 to 5