

# 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.

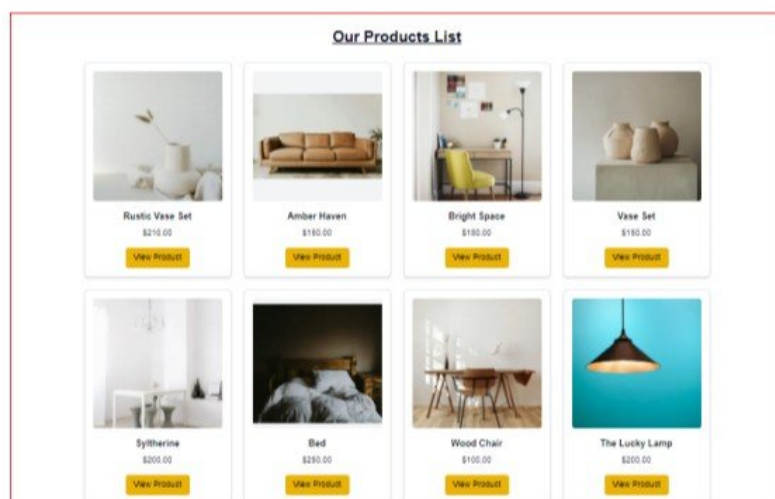
```

'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;
}

```



## 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.

```

1 > app > productDetails / id : 70 pages > 12 ProductDetailsPage > 88 main.jsx
2
3 const [product, setProduct] = useState<Product | null>(null);
4
5 const [loading, setLoading] = useState<boolean>(false);
6
7 const [showFullDescription, setShowFullDescription] = useState<boolean>(false);
8
9 const [selectedSize, setSelectedSize] = useState<string | null>(null);
10
11 const [selectedColor, setSelectedColor] = useState<string | null>(null);
12
13 const [quantity, setQuantity] = useState<number>(1);
14
15
16 const availableSizes = ["small", "medium", "large", "extra large"];
17
18 const availableColors = ["black", "brown", "silver", "blue", "white"];
19
20
21 useEffect(() => {
22   if (id) {
23     const fetchProductDetails = async () => {
24       const fetchedProduct = await getProductDetails(id as string);
25       setProduct(fetchedProduct);
26       setLoading(false);
27     };
28     fetchProductDetails();
29   }
30 }, [id]);
31
32 const toggleDescription = () => setShowFullDescription(!showFullDescription);
33
34 const handleQuantityChange = (operation: "increment" | "decrement") => {
35   setQuantity((prev) => (operation === "increment" ? prev + 1 : prev - 1 > 0 ? prev - 1 : 1));
36 };
37
38 const handleAddToCart = () => {
39   if (!product) return;
40   if (!selectedSize || !selectedColor) {
41     alert("Please select size and color before adding to cart.");
42     return;
43   }
44
45   const newItem: CartItem = {
46     id: product.id,
47     title: product.title,
48     price: product.price,
49     imageUrl: product.imageUrl,
50     size: selectedSize,
51     color: selectedColor,
52     quantity,
53   };
54
55   // Add to cart logic
56 };

```

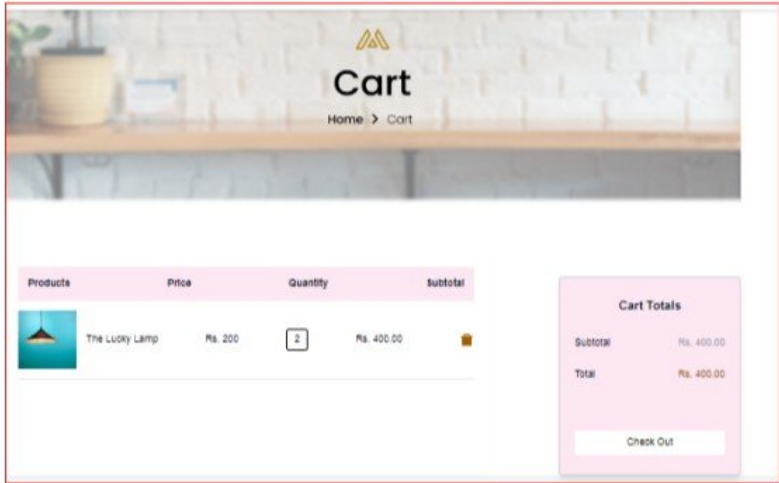


### 3. Cart Functionality

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.



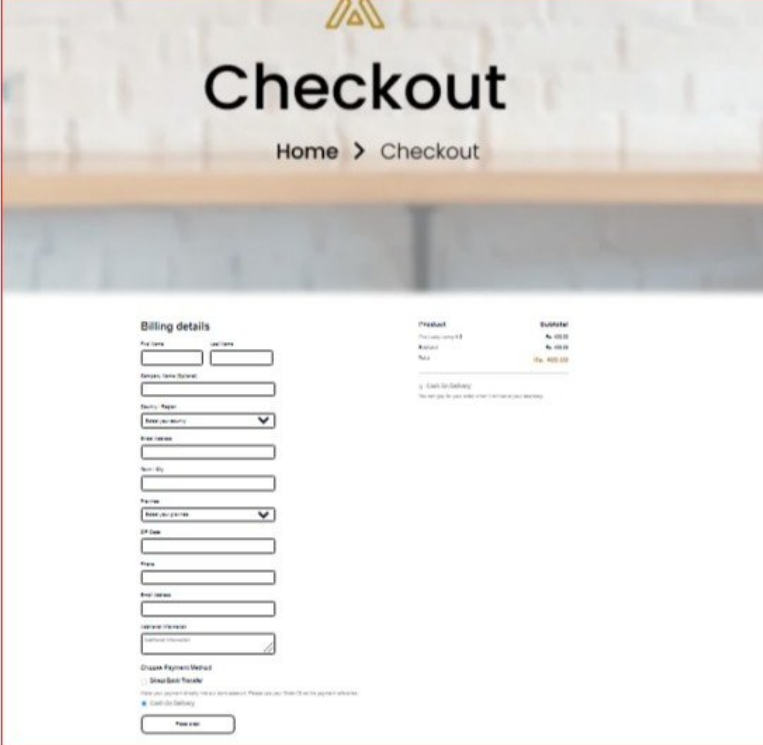
[illegible]

## 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 > 16 pages > ...
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 }
```



The image shows a mobile app checkout screen. At the top, there's a yellow logo and the word "Checkout" in large black font. Below it, a breadcrumb trail reads "Home > Checkout". The main content area is divided into two columns. The left column, titled "Billing details", contains several form fields: "Full Name" and "Last Name" (both with input fields), "Company Name (Optional)" (with an input field), "Country" (a dropdown menu showing "United Kingdom"), "Phone Number" (with an input field), "Email" (with an input field), "City" (with an input field), "Postcode" (with an input field), "CVV" (with an input field), "Card Number" (with an input field), and "Card Expiry" (with an input field). Below these fields is a section for "Quick Payment Method" with a radio button selected for "Credit Card" and a "Pay Now" button. The right column, titled "Product", shows a list of items with their prices and a total. At the bottom right, there's a "Pay Now" button.

## Checkout

Home > Checkout

### Billing details

Full Name

Last Name

Company Name (Optional)

Country

Phone Number

Email

City

Postcode

CVV

Card Number

Card Expiry

Quick Payment Method

☒ Credit Card

☐ Debit Card

☐ Net Banking

☐ Cash on Delivery

### Product

Product	Quantity	Price	Total
Product 1	1	100.00	100.00
Product 2	1	100.00	100.00
<b>Total</b>			<b>200.00</b>

## 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.

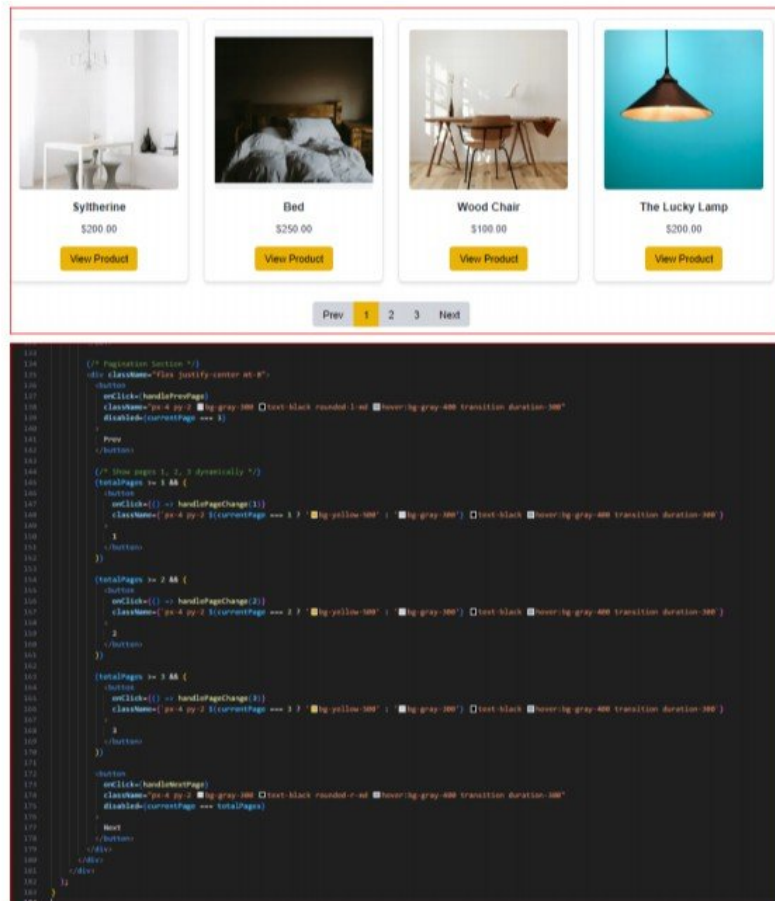
## 6. Pagination

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.







## 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

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.