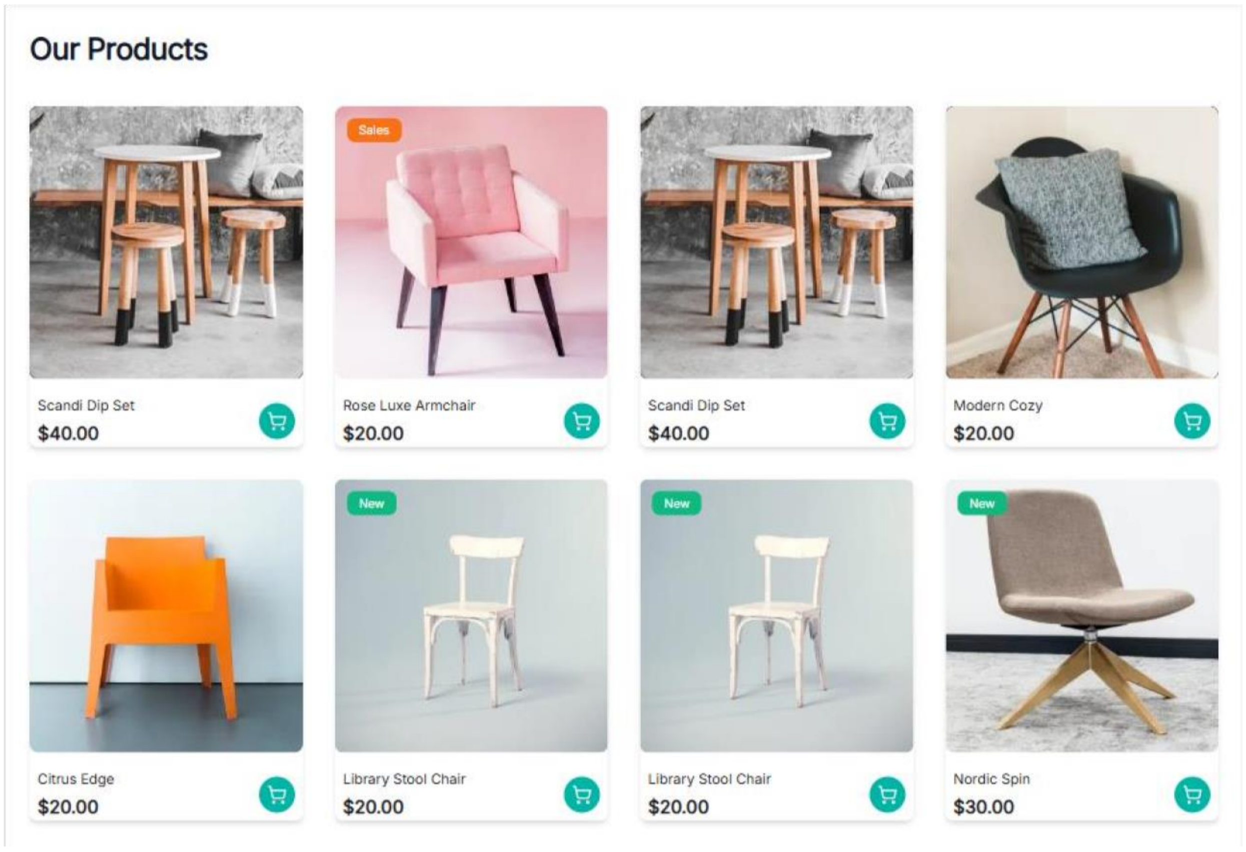
Day 4 Hackathon Report: Dynamic Frontend Components 🚀

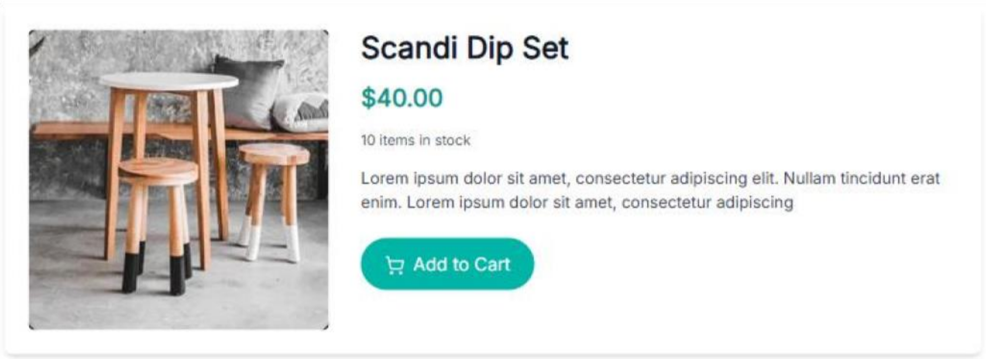
1. Functional Deliverables

Screenshots or Screen Recordings

- **Product Listing Page:**
  - The product listing page dynamically fetches data from **Sanity CMS**. ○ Each product is displayed with its name, image, price, and a brief description.
  -



- **Individual Product Detail Pages:**
  - Implemented dynamic routing to render individual product detail pages based on the product ID.
  - Each page displays detailed information, including the product name, images, price, description, and an "Add to Cart" button.



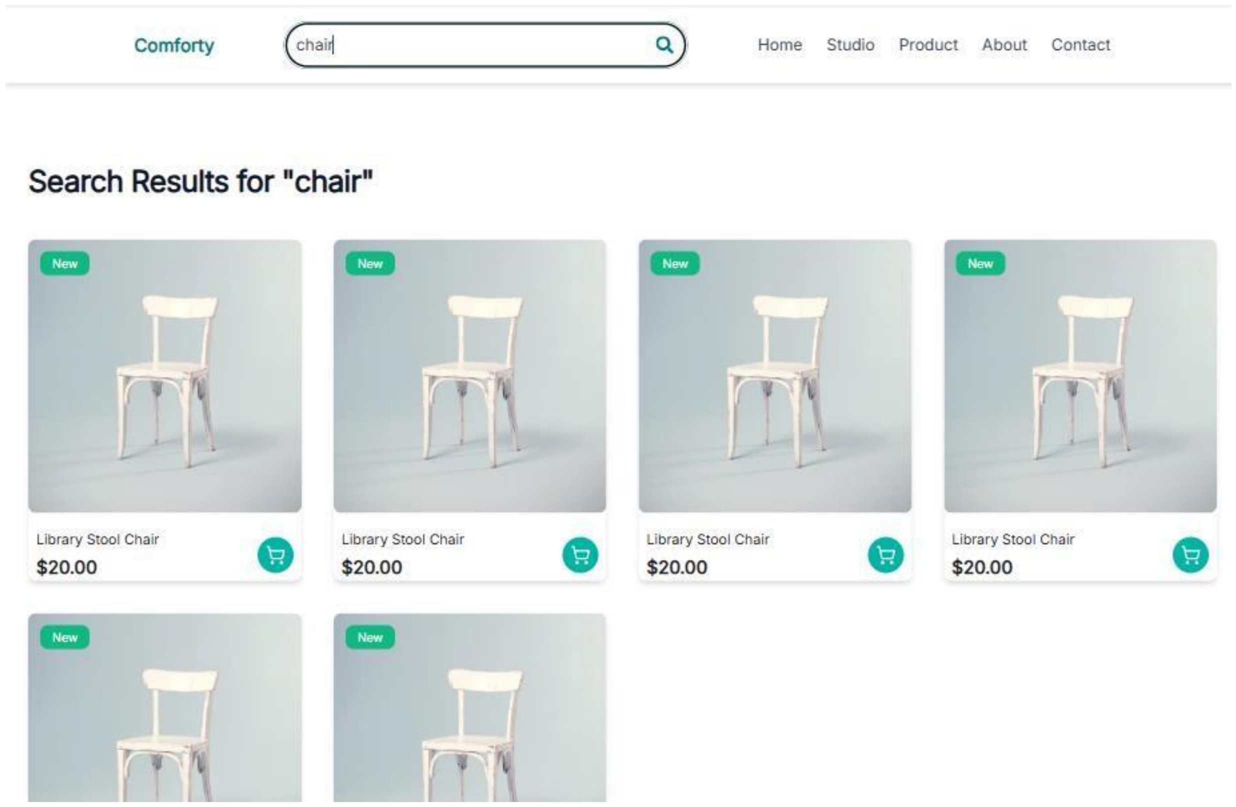
Scandi Dip Set

$40.00

10 items in stock

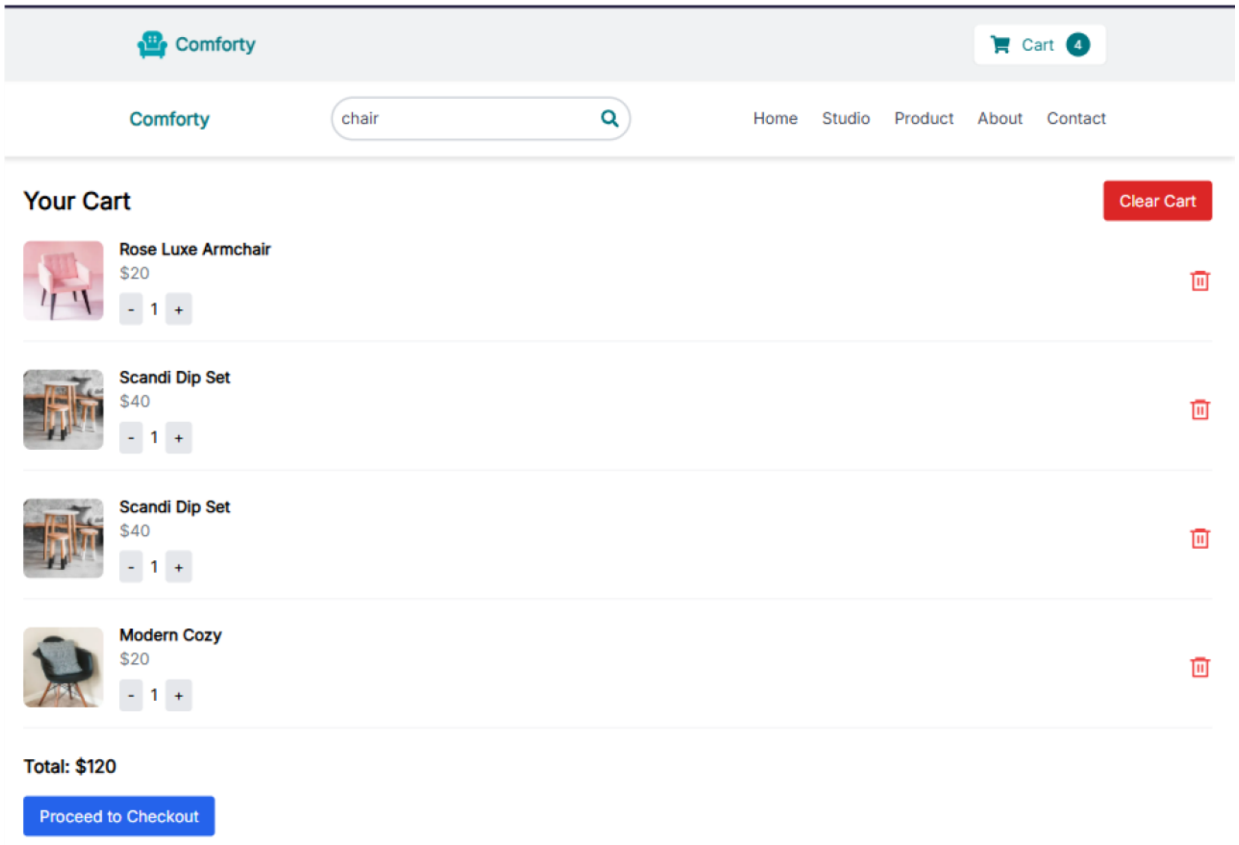
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

- **Category Filters, Search Bar, and Pagination:**
  - Integrated category filters to sort products based on predefined categories.
  - Added a fully functional search bar that dynamically filters the product list as the user types.
  - Pagination implemented for navigating through large datasets, ensuring optimal performance.



Search Results for "chair"

 Add To Cart: ○ Displayed related products on individual product pages based on category. ○ Integrated a "Cart" component showcasing all products that we add in cart.



2. Code Deliverables

Key Component Code Snippets

All ProductCard Component

```
"use client";

import { useEffect, useState } from "react"; import Image
from "next/image"; import { Product } from
"@/types/products"; import { allProducts } from
```

```
"@/src/sanity/lib/queries"; import { client, urlFor } from
"@/src/sanity/lib/client"; import { ShoppingCart } from
"lucide-react"; import Link from "next/link"; import {
addToCart } from "@/src/app/actions/actions";
 export default function AllProduct() {
const [products, setProducts] =
useState<Product[]>([]);
   useEffect(() => {     async function fetchProducts() {       try {
const fetchedProducts: Product[] = await client.fetch(allProducts);
setProducts(fetchedProducts);
      } catch (error) {         console.error("Error
fetching products:", error);
      }     }
fetchProducts();
  }, []);
   const handleAddToCart = (e: React.MouseEvent, product: Product) => {
e.preventDefault();     addToCart(product);

    // Dispatch event to notify the header component
const event = new CustomEvent("cart-updated");
window.dispatchEvent(event);
  };
  return (
    <section className="w-full mx-auto px-6 py-20">
      <header className="mb-10">
        <h1 className="text-3xl font-bold tracking-tight text-gray-900">
          All Products
        </h1>
      </header>

      {products.length > 0 ? (
        <div   className="grid   grid-cols-1   sm:grid-cols-2   md:grid-cols-3
lg:gridcols-4 gap-8">
```

```
          {products.map((prod) => {
const productImage =
              prod.image && Array.isArray(prod.image)
                ? prod.image[0]
                : prod.image;

            return (                <div                    key={prod._id}
className="group relative rounded-lg bg-white shadow-md
hover:shadow-lg transition-shadow duration-300"
            >
                {/* Product Image */}
                <Link href={`/product/${prod.slug.current}`}>
                  <div className="relative overflow-hidden rounded-lg
aspectsquare">
                    {prod.badge && (
                      <span                           className={`absolute left-
3 top-3 px-3 py-1 rounded-lg text-xs ${                            prod.badge
=== "New"
                            ? "bg-emerald-500 text-white"
                            : "bg-orange-500 text-white"
                        }`}
                      >
                        {prod.badge}
                      </span>
                    )}
                    {productImage ? (
                      <Image
src={urlFor(productImage).url()}
alt={prod.title}                           width={400}
height={400}                           className="w-full h-full object-
cover transitiontransform duration-300 group-hover:scale-105"
                      />
                    ) : (
                      <div className="w-full h-full flex items-center
justifycenter bg-gray-200 text-gray-500">
                        No Image
                      </div>
                    )}
                  </div>
                </Link>

                {/* Product Info */}
                <div className="mt-4 flex justify-between items-center px-2">
                  <div>
                    <h3 className="text-sm text-[#1C1B1F] font-medium">
                      {prod.title}
                    </h3>
                    <div className="mt-1 flex items-center gap-2">
                      <span className="text-lg font-semibold text-[#1C1B1F]">
                        ${prod.price.toFixed(2)}
                      </span>
```

```
                 </div>
     </div>
```

```
                  <button                     className="p-2 bg-[#00B5A5] text-
white rounded-full hover:bg-
[#00A294] transition-colors"                        aria-
label="Add to cart"                       onClick={(e) =>
handleAddToCart(e, prod)}
                  >
                    <ShoppingCart className="h-5 w-5" />
                  </button>
                </div>
              </div>
            );
}))}
        </div>
      ) : (
        <div>No products available.</div>
      )}
    </section>
  );
}
```

SearchBar Component:

```
"use client";
 import { useEffect, useState } from "react"; import Image
from "next/image"; import { useSearchParams } from
"next/navigation"; import { Product } from
"@/types/products"; import { client, urlFor } from
"@/src/sanity/lib/client"; import { ShoppingCart } from
```

```
"lucide-react"; import Link from "next/link"; import {
addToCart } from "@/src/app/actions/actions";

export default function SearchResults() {   const
[products, setProducts] = useState<Product[]>([]);   const
searchParams = useSearchParams();
  const query = searchParams?.get("query") || "";
  useEffect(() => {
    async function fetchProducts() {
    if (!query) return;

      const fetchedProducts: Product[] = await client.fetch(
        `*[_type == "products" && title match "${query}*"]{
          _id,
title,
price,
slug,
image,
badge
        }`        );
setProducts(fetchedProducts);
    }
fetchProducts();
  }, [query]);
   const handleAddToCart = (e: React.MouseEvent, product: Product) => {
e.preventDefault();     addToCart(product);

    // Dispatch event to notify the header component
const event = new CustomEvent("cart-updated");
window.dispatchEvent(event);
  };
  return (
    <section className="w-full mx-auto px-6 py-20">
      <header className="mb-10 flex justify-between items-center">
        <h1 className="text-3xl font-bold tracking-tight text-gray-900">
```

```
          Search Results for &quot;{query}&quot;
        </h1>
      </header>

      {products.length > 0 ? (
        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-8">
          {products.map((prod) => {             const
productImage =                  prod.image &&
Array.isArray(prod.image)
                ? prod.image[0]
                : prod.image;
            return (
             <div
             key={prod._id}
                className="group relative rounded-lg bg-white shadow-md
hover:shadow-lg transition-shadow duration-300"
              >
                {/* Product Image */}
                <Link href={`/product/${prod.slug.current}`}>
                  <div className="relative overflow-hidden rounded-lg
aspectsquare">
                    {prod.badge && (
                      <span                         className={`absolute left-
3 top-3 px-3 py-1 rounded-lg text-xs ${                           prod.badge
=== "New"
                            ? "bg-emerald-500 text-white"
                            : "bg-orange-500 text-white"
                        }`}
                      >
                        {prod.badge}
                      </span>
                    )}
                    {productImage ? (
                      <Image
src={urlFor(productImage).url()}
alt={prod.title}                        width={400}
height={400}                        className="w-full h-full object-
cover transitiontransform duration-300 group-hover:scale-105"
                      />
                    ) : (
                      <div className="w-full h-full flex items-center
justifycenter bg-gray-200 text-gray-500">
                        No Image
                      </div>
                    )}
                  </div>
                </Link>

                {/* Product Info */}
                <div className="mt-4 flex justify-between items-center px-2">
                  <div>
                    <h3 className="text-sm text-[#1C1B1F] font-medium">
                      {prod.title}
                    </h3>
                    <div className="mt-1 flex items-center gap-2">
```

```
                      <span className="text-lg font-semibold text-[#1C1B1F]">
                        ${prod.price.toFixed(2)}
                      </span>
                    </div>
                  </div>

                  <button                     className="p-2 bg-[#00B5A5] text-
white rounded-full hover:bg-
[#00A294] transition-colors"                    aria-
label="Add to cart"                     onClick={(e) =>
handleAddToCart(e, prod)}
                  >
                    <ShoppingCart className="h-5 w-5" />
                  </button>
                </div>
              </div>
            );
          })}
        </div>
      ) : (
        <div className="text-center">
          <h2 className="text-xl font-medium text-gray-600">
            No products found for &quot;{query}&quot;
          </h2>
        </div>
      )}
    </section>
  );
}
```

API Integration and Dynamic Routing

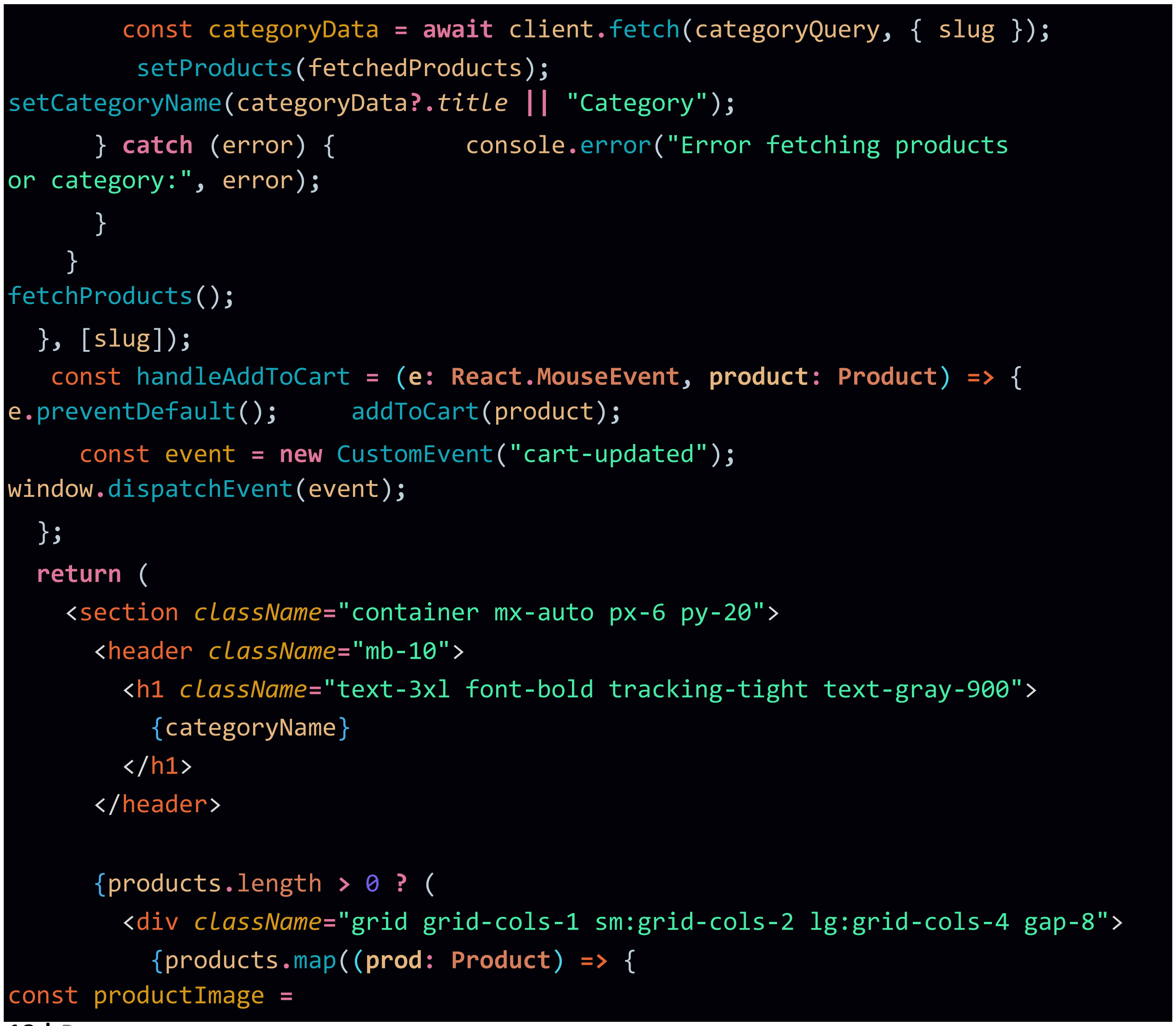
```
"use client";
 import { useEffect, useState } from
"react";
```

```
import { useParams } from "next/navigation"; // ✅ Fix: Use useParams()
import { Product } from "@/types/products"; import { client, urlFor }
from "@/src/sanity/lib/client"; import { groq } from "next-sanity";
import Image from "next/image";
import { ShoppingCart } from "lucide-react"; import
Swal from "sweetalert2"; import { addToCart } from
"@/src/app/actions/actions";
async function getProduct(slug: string): Promise<Product | null> {
  return await client.fetch(
groq`
      *[_type == "products" && slug.current == $slug][0] {
        _id,          title,
description,          price,
priceWithoutDiscount,
badge,          image,
inventory,          slug,
      }
    `,
    { slug }
  );
}  export default function ProductPage() {   const params =
useParams(); // ✅ Fix: useParams() to unwrap params
  const slug = Array.isArray(params?.slug) ? params.slug[0] : params?.slug; //
✅ Ensure slug is a string
   const [product, setProduct] = useState<Product |
null>(null);
   useEffect(() => {     if (!slug) return; // ✅ Prevent
fetching if slug is missing
     async function fetchProduct() {
if (slug) {
        const fetchedProduct = await getProduct(slug);
setProduct(fetchedProduct);
      }
    }
    fetchProduct();
  }, [slug]);
   const handleAddToCart = (e: React.MouseEvent) =>
{
    e.preventDefault();      if (product && product.inventory
> 0) {        addToCart(product);
window.dispatchEvent(new CustomEvent("cart-updated"));
    } else {       Swal.fire({
title: "Out of Stock",
        text: `Sorry, ${product?.title} is out of stock.`,
        icon: "error",
        confirmButtonText: "OK",
      });
    }
  };    if
(!product) {
return (
      <div className="container mx-auto px-6 py-20">
        <p className="text-center text-gray-500">Loading product...</p>
      </div>
    );
  }
  return (
    <div className="container mx-auto px-6 py-20">
      <div className="flex flex-col lg:flex-row items-start bg-white rounded-lg
shadow-md p-6 gap-8">
        <div className="flex-shrink-0 mb-6 lg:mb-0">
          <Image
src={urlFor(product.image).url()}
alt={product.title}             width={300}
height={300}             className="rounded-md
object-contain"
          />
        </div>

        <div className="flex-grow">
          <h1 className="text-3xl font-bold text-gray-900">{product.title}</h1>
          {product.badge && (
            <span
              className={`inline-block px-3 py-1 mt-2 text-sm font-medium rounded
${                   product.badge ===
"New"

                    ? "bg-emerald-500 text-white"
                    : "bg-orange-500 text-white"
              }`}
            >
              {product.badge}
          </span>
          )}

          <div className="mt-4 flex items-center gap-4">
            <span className="text-2xl font-semibold text-teal-600">
              ${product.price.toFixed(2)}
            </span>
            {product.priceWithoutDiscount && (
              <span className="text-lg text-gray-500 line-through">
                ${product.priceWithoutDiscount.toFixed(2)}
              </span>
            )}
          </div>

          <p className="mt-4 text-sm text-gray-600">
            {product.inventory
              ? `${product.inventory} items in stock`
              : "Out of stock"}
          </p>

          <p className="mt-4 text-gray-700">{product.description}</p>
          <button
className={`mt-6 px-6 py-3 ${
product.inventory > 0
                ? "bg-[#00B5A5] hover:bg-[#00A294]"
                : "bg-gray-500 cursor-not-allowed"
            } text-white text-lg font-medium rounded-full transition-colors`}
aria-label="Add to cart"            onClick={handleAddToCart}
disabled={!product.inventory}
          >
            <ShoppingCart className="h-5 w-5 inline-block mr-2" />
            {product.inventory ? "Add to Cart" : "Out of Stock"}
          </button>
        </div>
      </div>
    </div>
```

```
  );
}
```

Categories Slug Code:

```
"use client";  import { useEffect, useState }
from "react"; import Image from "next/image";
import Link from "next/link"; import { useParams } from "next/navigation";
import { client, urlFor } from "@/src/sanity/lib/client"; import { groq }
from "next-sanity"; import { ShoppingCart } from "lucide-react"; import {
addToCart } from "@/src/app/actions/actions"; import { Product } from
"@/types/products"; // Use the shared Product type
 export default function CategoryProducts()
{   const params = useParams();
  const slug = params.slug as string;
   const [products, setProducts] = useState<Product[]>([]);   const
[categoryName, setCategoryName] = useState<string | null>(null);
   useEffect(() => {     async
function fetchProducts() {
const query = groq`
        *[_type == "products" && category->slug.current == $slug] {
          _id,           _type,
title,           price,
badge,           inventory,
priceWithoutDiscount,
slug,           image {
asset-> {               _ref,
url
            }
          }
        }       `;       const
categoryQuery = groq`
        *[_type == "categories" && slug.current == $slug][0] {
          title
 }
      `;
      try {
        const fetchedProducts = await client.fetch(query, { slug });
```



```
        const categoryData = await client.fetch(categoryQuery, { slug });
         setProducts(fetchedProducts);
setCategoryName(categoryData?.title || "Category");
      } catch (error) {         console.error("Error fetching products
or category:", error);
      }
    }
fetchProducts();
  }, [slug]);
   const handleAddToCart = (e: React.MouseEvent, product: Product) => {
e.preventDefault();     addToCart(product);
     const event = new CustomEvent("cart-updated");
window.dispatchEvent(event);
  };
  return (
    <section className="container mx-auto px-6 py-20">
      <header className="mb-10">
        <h1 className="text-3xl font-bold tracking-tight text-gray-900">
          {categoryName}
        </h1>
      </header>

      {products.length > 0 ? (
        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-8">
          {products.map((prod: Product) => {
const productImage =
              prod.image && Array.isArray(prod.image)
                ? prod.image[0]
                : prod.image;
             return (
<div
                key={prod._id}
                className="group relative rounded-lg bg-white shadow-md
hover:shadow-lg transition-shadow duration-300"
              >
                <Link href={`/product/${prod.slug.current}`}>
```

```
                              <div className="relative overflow-hidden rounded-lg aspect-
square">
                    {prod.badge && (
                      <span                        className={`absolute left-
3 top-3 px-3 py-1 rounded-lg text-xs ${                            prod.badge
=== "New"
                              ? "bg-emerald-500 text-white"
                              : "bg-orange-500 text-white"
                        }`}
                      >
                        {prod.badge}
                      </span>
                    )}
                    {productImage ? (
                      <Image
src={urlFor(productImage).url()}
alt={prod.title}                          width={400}
height={400}                          className="w-full h-full object-
cover transitiontransform duration-300 group-hover:scale-105"
                      />
                    ) : (
                      <div className="w-full h-full flex items-center
justifycenter bg-gray-200 text-gray-500">
                        No Image
                      </div>
                    )}
                  </div>
                </Link>

                <div className="mt-4 flex justify-between items-center px-2">
                  <div>
                    <h3 className="text-sm text-[#1C1B1F] font-medium">
                      {prod.title}
                    </h3>
                    <div className="mt-1 flex items-center gap-2">
                      <span className="text-lg font-semibold text-[#1C1B1F]">
                        ${prod.price.toFixed(2)}
                      </span>
                    </div>
                  </div>

                 <button
                    className="p-2 bg-[#00B5A5] text-white rounded-full hover:bg-
[#00A294] transition-colors"                            aria-
label="Add to cart"                             onClick={(e) =>
handleAddToCart(e, prod)}
                  >
                    <ShoppingCart className="h-5 w-5" />
                  </button>
                </div>
              </div>
            );
})}
        </div>
      ) : (
        <div className="text-center text-gray-500 text-lg">
          No products available in this category.
        </div>
)}
    </section>
  );
}
```

Cart Code:

```
"use client";
 import React, { useState, useEffect } from "react";
import Image from "next/image";
import { RiDeleteBin6Line } from "react-icons/ri"; import Swal
from "sweetalert2"; import { getCartItems, removeFromCart,
updateCartQuantity } from
"../actions/actions";
import { Product } from "@/types/products"; import
{ urlFor } from "@/src/sanity/lib/client";

const Cart = () => {   const [cartItems, setCartItems] =
useState<Product[]>([]);

  useEffect(() => {
setCartItems(getCartItems());   },
[]);
  const handleRemove = (id: string) => {
    Swal.fire({       title: "Are you sure?",       text:
"You will not be able to recover this item!",       icon:
"warning",       showCancelButton: true,
confirmButtonColor: "#3085d6",       cancelButtonColor:
"#d33",       confirmButtonText: "Yes, delete it!",
    }).then((result) => {       if (result.isConfirmed) {
removeFromCart(id);         setCartItems(getCartItems()); //
Refresh cart items         const event = new
CustomEvent("cart-updated");
window.dispatchEvent(event); // Trigger header update
        Swal.fire("Removed!", "Item has been removed.", "success");
      }
    });
  };    const handleQuantityChange = (id: string, quantity:
number) => {     updateCartQuantity(id, quantity);
setCartItems(getCartItems());
```

```
  };    const handleIncrement = (id: string) => {     const
product = cartItems.find((item) => item._id === id);     if
(product) {       handleQuantityChange(id, product.inventory +
1);
    }
  };
  const handleDecrement = (id: string) => {
    const product = cartItems.find((item) => item._id === id);
if (product && product.inventory > 1) {
handleQuantityChange(id, product.inventory - 1);
    }
  };    const getTotalPrice = ()
=> {     return
cartItems.reduce(
      (total, item) => total + (item.price ? item.price * item.inventory : 0),
      0
    );
  };
  const handleProceedToCheckout = () => {
const cartSummary = cartItems
      .map(
        (item) =>
          `<div class='flex justify-between mb-2'>
            <span>${item.title} (x${item.inventory})</span>
            <span>$${item.price * item.inventory}</span>
          </div>`
      )
      .join("");
     const total =
getTotalPrice();

    Swal.fire({       title:
"Order Summary",       html: `
        <div class='text-left'>
```

```
          <h3 class='text-lg font-bold mb-3'>Items:</h3>
          ${cartSummary}
          <hr class='my-3' />
          <h4 class='text-lg font-bold'>Total: $${total}</h4>
        </div>       `,
showCancelButton: true,
confirmButtonText: "Place Order",
cancelButtonText: "Cancel",
confirmButtonColor: "#3085d6",
cancelButtonColor: "#d33",
}).then((result) => {       if
(result.isConfirmed) {
Swal.fire({
          title: "Placing Order...",
          html: `<div class='flex justify-center items-center'>
                   <div class='animate-spin rounded-full h-8 w-8 border-t-2
border-b-2 border-blue-500'></div>
                 </div>`,
allowOutsideClick: false,
showConfirmButton: false,
didOpen: () => {
setTimeout(() => {
              Swal.fire("Order Placed!", "Your order has been placed
successfully.", "success");                localStorage.clear();
              setCartItems([]); // Clear cart after checkout
               const event = new CustomEvent("cart-updated");
              window.dispatchEvent(event); // Trigger header update
            }, 2000);
          },
        });
      }
    });
  };

  // Clear Cart function
  const handleClearCart = () => {     Swal.fire({
title: "Are you sure?",       text: "This will remove all
items from your cart!",       icon: "warning",
showCancelButton: true,       confirmButtonColor:
"#3085d6",       cancelButtonColor: "#d33",
      confirmButtonText: "Yes, clear it!",
    }).then((result) => {       if (result.isConfirmed) {
localStorage.removeItem("cart"); // Remove items from local storage
setCartItems([]); // Clear cart state         const event = new
CustomEvent("cart-updated");         window.dispatchEvent(event); //
Trigger header update
        Swal.fire("Cleared!", "All items have been removed from your cart.",
"success");
      }
    });
  };
  return (
    <div className="p-6">
      <div className="flex justify-between items-center mb-4">
        <h2 className="text-2xl font-bold">Your Cart</h2>
        <button
          onClick={handleClearCart}
          className="px-4 py-2 bg-red-600 text-white rounded hover:bg-red-700"
        >
          Clear Cart
        </button>
      </div>
      {cartItems.length > 0 ? (
        <div className="space-y-6">
          {cartItems.map((item) => (
            <div key={item._id} className="flex items-center space-x-4 border-b
pb-4">
              {item.image ? (                <Image
src={urlFor(item.image).url()}                alt={item.image?.alt ||
item.title || "Product image"}                    width={80}
height={80}                     className="rounded-lg"
                />
              ) : (
<Image
src="/placeholder.png"
alt="Placeholder Image"
width={80}                     height={80}
className="rounded-lg"
                />
              )}
              <div className="flex-1">
                <p className="font-semibold">{item.title}</p>
                <p className="text-gray-500">${item.price}</p>
                <div className="flex items-center space-x-2 mt-2">
                  <button                         onClick={() =>
handleDecrement(item._id)}                         className="px-2 py-1 bg-gray-200
rounded hover:bg-gray-300"
                  >
                    -
                  </button>
                  <span>{item.inventory}</span>
                  <button
                    onClick={() => handleIncrement(item._id)}
                    className="px-2 py-1 bg-gray-200 rounded hover:bg-gray-300"
                  >
                    +
                  </button>
                </div>
              </div>
              <RiDeleteBin6Line
size={24}
                className="text-red-500 cursor-pointer"
onClick={() => handleRemove(item._id)}               />
            </div>
```

```
          ))}
          <div className="mt-4">
            <p className="text-lg font-semibold">Total: ${getTotalPrice()}</p>
            <button               onClick={handleProceedToCheckout}
className="mt-4 px-4 py-2 bg-blue-600 text-white rounded hover:bgblue-700"
            >
              Proceed to Checkout
            </button>
          </div>
        </div>
      ) : (
        <div className="flex flex-col items-center justify-center h-64 text-
gray500">
          <p className="mt-4 text-lg font-semibold">Your cart is empty!</p>
          <p className="text-sm">Add some items to get started.</p>
        </div>
      )}
    </div>
  );
};  export default
Cart;
```

Best Practices Followed

1. Ensured reusable components by following the DRY principle.
2. Used TypeScript for type safety and better code maintainability.
3. Followed accessibility guidelines to ensure a user-friendly interface.
4. Validated inputs and API responses to prevent errors and improve security.