

Day 4 - Dynamic Frontend Components - [Comfy]

Roll no : 118068

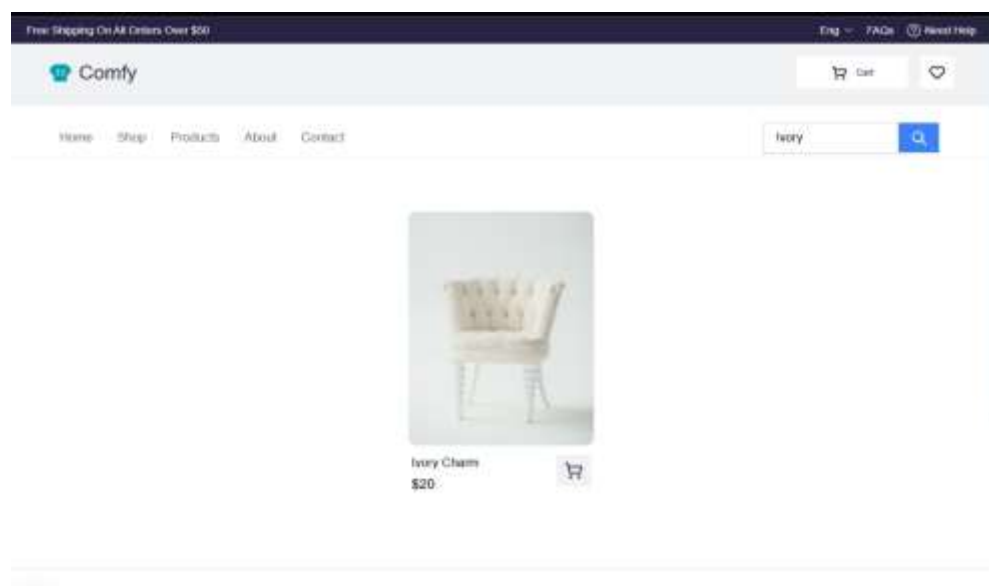
Overview:

This document provides an overview of the general e-commerce website built with dynamic components to enhance user experience and interactivity. The primary goal of this project is to create a modern, feature-rich platform where users can easily browse, search, and purchase products.

Functional Deliverables:

1.Search Page:

- Created a dynamic route: /search/[name].
- Designed a search input form with event handling to capture user input.
- Used a GROQ query to fetch matching products from Sanity CMS, filtering by product name or category.
- Rendered the search results dynamically on the page using reusable components.



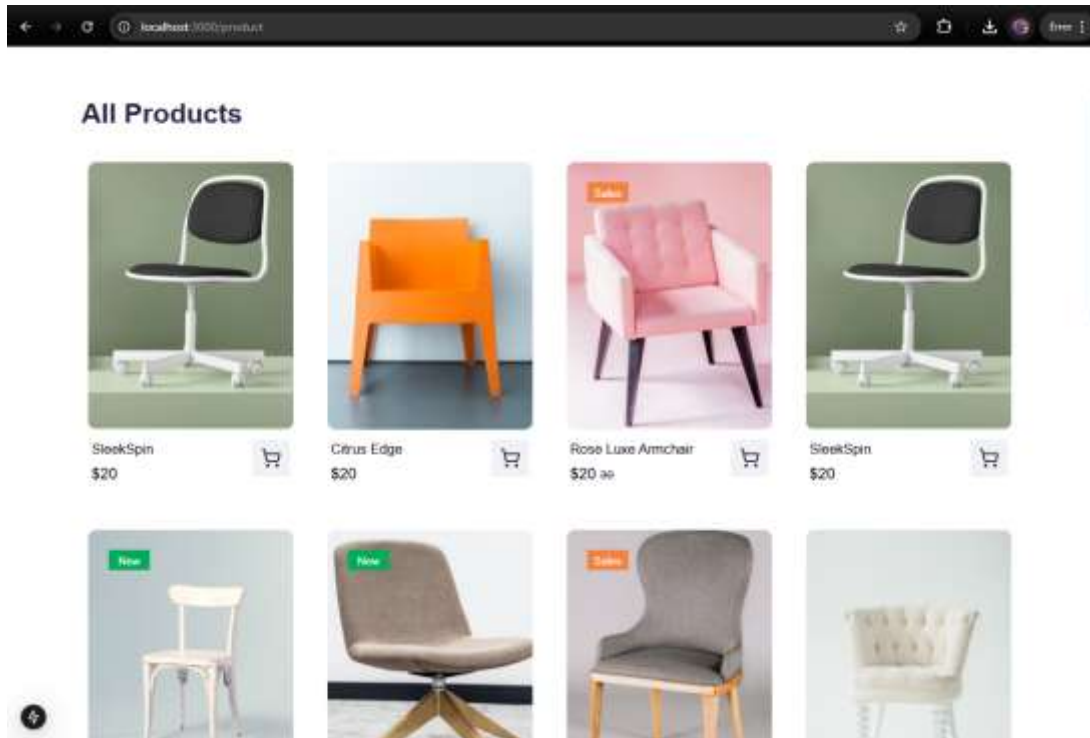
Code snippet:



```
src > app > search > [search] > page.tsx > Searchpage
1  import { fetchSearchProducts } from '@utils/searchProduct';
2  import React from 'react';
3  import { productType } from '@utils/type';
4  import ProductCard from '@components/productcard';
5  import Image from 'next/image';
6
7  async function Searchpage({params}:{params:Promise<{search:string}>}) {
8    const searchparam = await params;
9    const {search}=searchparam;
10
11    const productName = decodeURIComponent(search);
12    const product:productType[] = await fetchSearchProducts(productName);
13
14    if(!product.length){
15      return(
16        <section className="py-14">
17          <div className="max-w-6xl mx-auto px-4 flex justify-center items-center sm:px-6 lg:px-8">
18            <div>
19              <Image
20                src={"/images/No_Product_Found.png"}
21                alt="pic"
22                className="w-80 h-80"
23                width={300}
24                height={300}
25              />
26            </div>
27          </div>
28        </section>
29      )
30    }
31  }
```

2. Product Listing Page;

- Developed a reusable ProductCard component to display product details like name, image, price, and badge.
- Fetched product data from Sanity CMS and mapped over the data to render multiple ProductCard components.
- Styled the product grid to ensure responsiveness across devices.



Code Snippet:

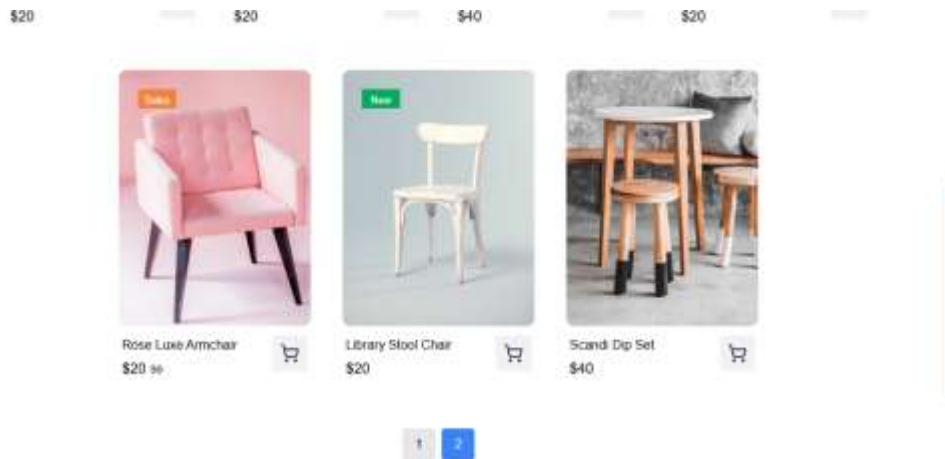
```

55
56
57 export default function AllProducts() {
58   const dispatch = useAppDispatch();
59   const products: ProductType[] = useAppSelector((state) => state.product.products);
60
61   const [currentPage, setCurrentPage] = useState(1);
62   const productsPerPage = 8;
63
64   useEffect(() => {
65     dispatch(fetchProducts());
66   }, [dispatch]);
67
68   const indexOffLastProduct = currentPage * productsPerPage;
69   const indexOffFirstProduct = indexOffLastProduct - productsPerPage;
70   const currentProducts = products.slice(indexOffFirstProduct, indexOffLastProduct);
71   const product = currentProducts.length;
72
73   return (
74     <section className="py-16 md:py-20">
75       <div className="max-w-6xl mx-auto px-4 sm:px-6 md:px-2 lg:px-1">
76         <div className="w-full py-1 flex items-center md:justify-between justify-center mb-5 px-6">
77           <span className="text-3xl md:text-[32px] font-semibold text-color1">
78             All Products
79           </span>
80         </div>
81         <div className="py-4 px-[8px] sm:px-2 md:px-0 w-full flex flex-row flex-wrap gap-x-4 sm:gap-x-10 gap-
82
83         <currentProducts ? (
84           currentProducts.map((e) => {
85             <ProductCard
86               key={e._id}
87               _id={e._id}
88               title={e.title}

```

3.Pagination:

- Calculated the total number of pages based on the product count and items per page.
- Dynamically fetched and rendered products based on the current page.



Code Snippet:

```
File Edit Selection View Go ... Monaco (Administrator)
Welcome | links.tsx | page.tsx | allProducts.tsx | PaginationProps.tsx X
src > components > PaginationProps.tsx (4) default
1 import React from "react";
2
3 interface PaginationProps {
4   total: number;
5   current: number;
6   onPageChange: (page: number) => void;
7 }
8
9 const PaginationProps: React.FC<PaginationProps> = ({ total, current, onPageChange }) => {
10   const pages = Array.from({ length: total }, (_, index) => index + 1);
11
12   return (
13     <div className="flex space-x-2">
14       {pages.map((page) => (
15         <button
16           key={page}
17           onClick={() => onPageChange(page)}
18           className={`px-4 py-2 rounded ${
19             current === page ? "bg-blue-500 text-white" : "bg-gray-200 text-black"
20           }`}
21         >
22           {page}
23         </button>
24       ))}
25     </div>
26   );
27 }
28
29 export default PaginationProps;
```

4.Product Detail Page:

- Set up dynamic routing for product details: /product/[id].
- Captured the id parameter from the route using params.
- Fetched product data from Sanity CMS by matching the id from the URL parameters.
- Displayed detailed information, including product description, price, images, and title.



Citrus Edge

\$20 USD

Stock : 20

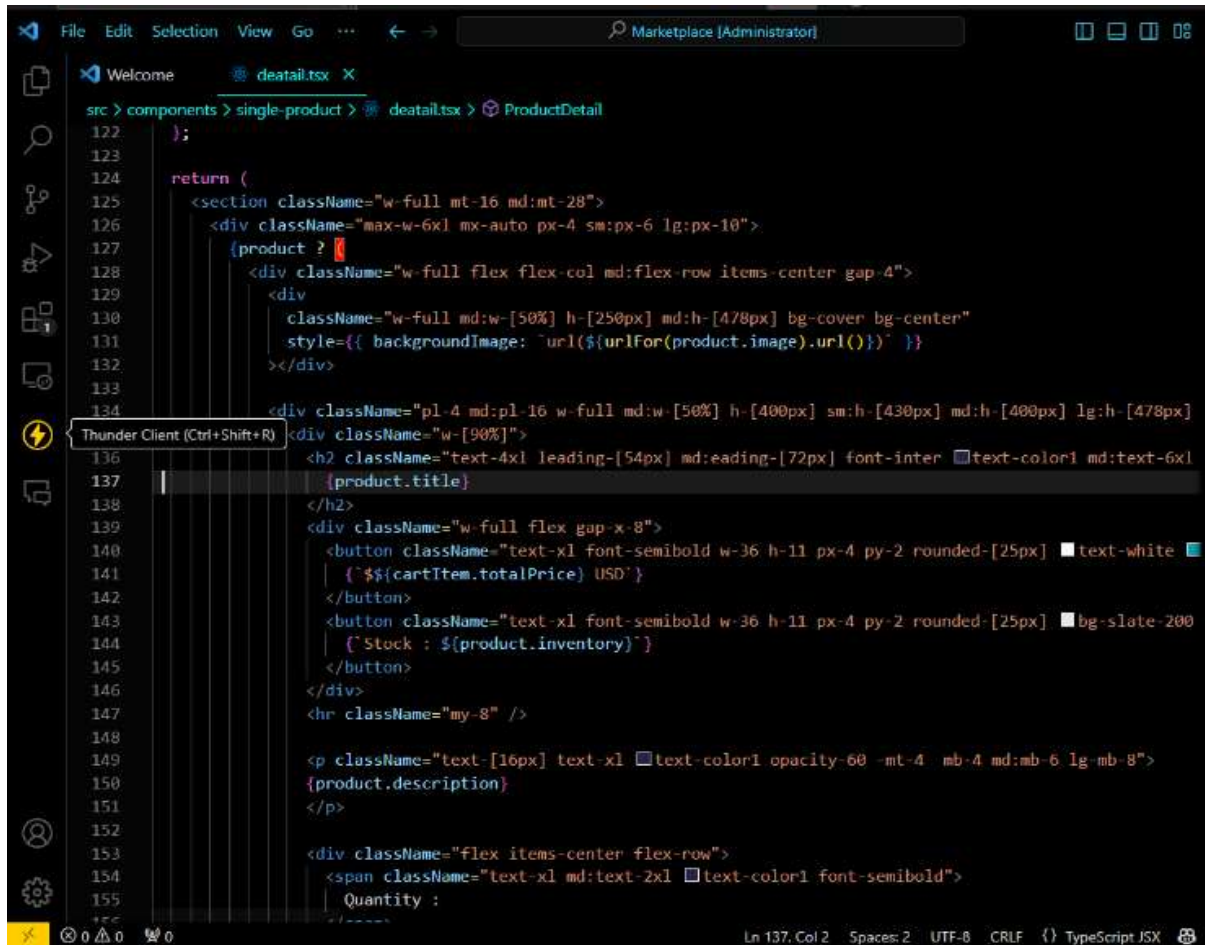
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

Quantity : + 1 -

 Add to cart

 Wishlist

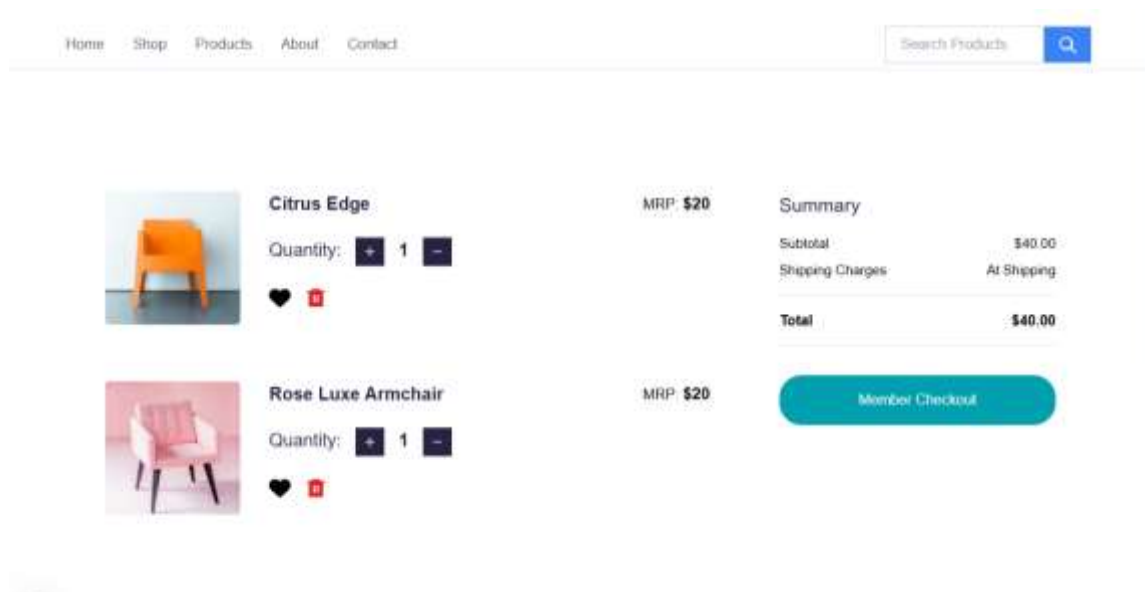
Code Snippet:



```
122  };
123
124  return (
125    <section className="w-full mt-16 md:mt-28">
126      <div className="max-w-6xl mx-auto px-4 sm:px-6 lg:px-10">
127        {product ? (
128          <div className="w-full flex flex-col md:flex-row items-center gap-4">
129            <div
130              className="w-full md:w-[50%] h-[250px] md:h-[478px] bg-cover bg-center"
131              style={{ backgroundImage: `url(${urlFor(product.image).url()})` }}
132            ></div>
133
134            <div className="pl-4 md:pl-16 w-full md:w-[50%] h-[400px] sm:h-[430px] md:h-[400px] lg:h-[478px]
135              <div className="w-[90%]">
136                <h2 className="text-4xl leading-[54px] md:leading-[72px] font-inter text-color1 md:text-6xl">
137                  {product.title}
138                </h2>
139                <div className="w-full flex gap-x-8">
140                  <button className="text-xl font-semibold w-36 h-11 px-4 py-2 rounded-[25px] text-white
141                    ({`${cartItem.totalPrice} USD`)}
142                  </button>
143                  <button className="text-xl font-semibold w-36 h-11 px-4 py-2 rounded-[25px] bg-slate-200
144                    {`Stock : ${product.inventory}`)}
145                  </button>
146                </div>
147                <hr className="my-8" />
148
149                <p className="text-[16px] text-xl text-color1 opacity-60 -mt-4 mb-4 md:mb-6 lg:mb-8">
150                  {product.description}
151                </p>
152
153                <div className="flex items-center flex-row">
154                  <span className="text-xl md:text-2xl text-color1 font-semibold">
155                    Quantity :
```

5.Add to Cart Functionality:

- Integrated Redux Toolkit for state management.
- Created a slice with actions and reducers to manage cart items.
- Implemented an "Add to Cart" button in the product detail page.
- Rendered cart items dynamically in a cart component.



Code Snippet:

```

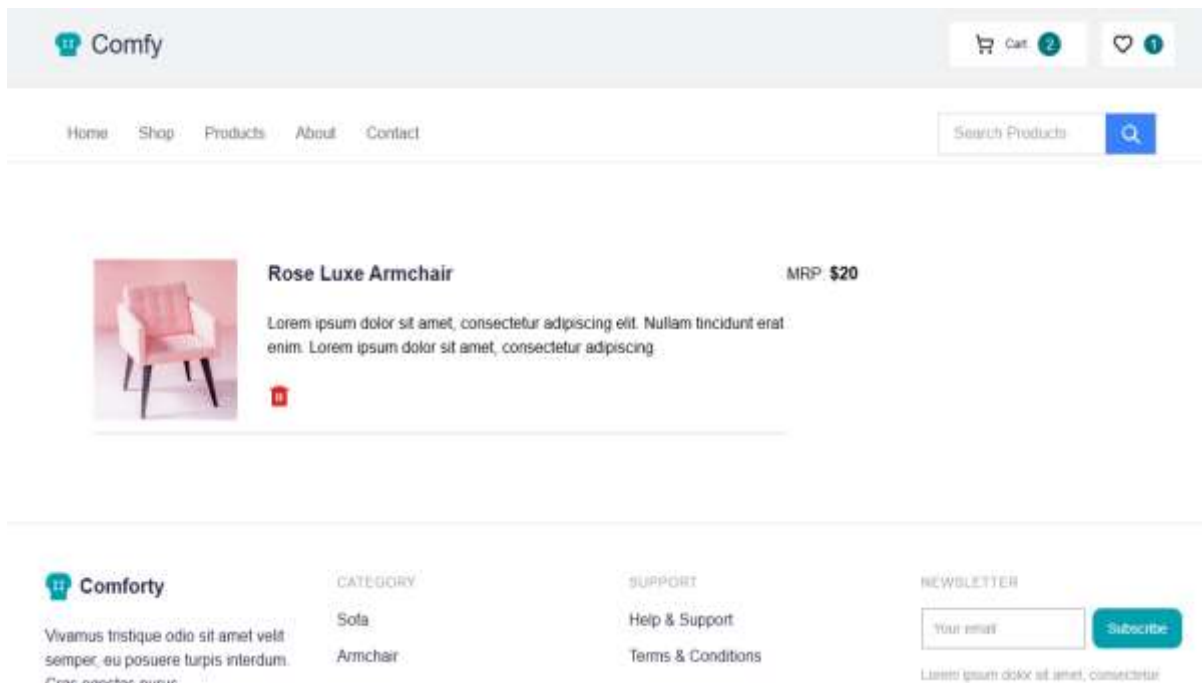
File Edit Selection View Go ... Marketplace (Administrator)
Welcome detail.jsx cart.jsx
src > components > cart.jsx > Cart
1 "use client";
2 import Link from "next/link";
3 import CartCard from "../cartCard";
4 import { useAppSelector } from "@app/store/hooks";
5
6 export default function Cart() {
7   const cart = useAppSelector((state) => state.cart);
8
9   const subtotal = cart.reduce(
10     (total, item) => total + item.price * item.quantity,
11     0
12   );
13
14   const totalPrice = subtotal;
15
16   return (
17     <section className="w-full py-12 mt-10 md:py-24">
18       <div className="max-w-6xl mx-auto px-6 sm:px-6 lg:px-12">
19         <div className="w-full flex flex-col md:flex-row gap-8 justify-between">
20           <div className="w-full lg:w-[70%] flex flex-col space-y-16">
21             {cart.length > 0 ? (
22               cart.map((e) => (
23                 <CartCard
24                   key={e._id}
25                   id={e._id}
26                   name={e.title}
27                   price={e.price * e.quantity}
28                   quantity={e.quantity}
29                   img={e.image}
30                   inventory={e.inventory}
31                 />
32               ))
33             ) : (
34               <div className="text-center text-gray-500 text-2xl">
35                 No items in cart
36               </div>
37             )}
38           <div>
39             <div>Subtotal</div>
40             <div>Shipping</div>
41             <div>Total</div>
42           </div>
43         </div>
44       </div>
45     </section>
46   );
47 }

```

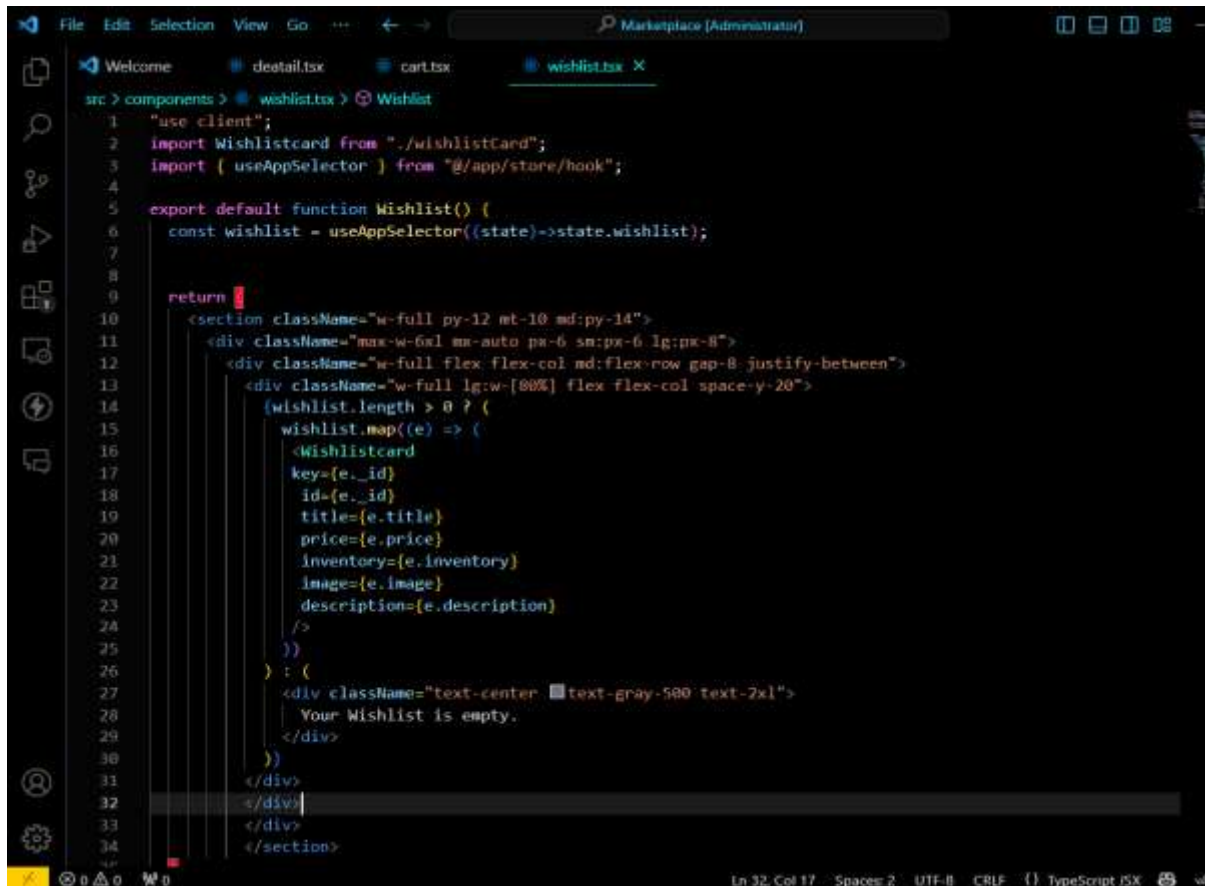
Ln 50, Col 40 Spaces: 2 UTF-8 CRLF TypeScript JSX

6.Wishlist Component:

- Added another slice to the Redux store for managing the wishlist.
- Created actions to add or remove products from the wishlist.
- Implemented a "Wishlist" button in the product detail page.
- Displayed the wishlist items on a dedicated page.



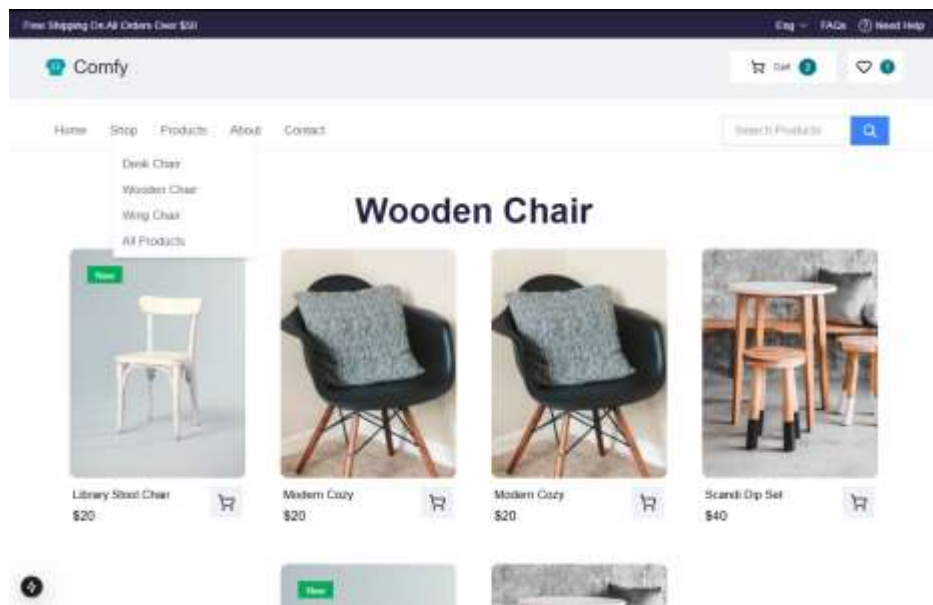
Code Snippet:



```
src > components > wishlist.tsx > Wishlist
1  "use client";
2  import Wishlistcard from "../wishlistcard";
3  import { useAppSelector } from "@app/store/hooks";
4
5  export default function Wishlist() {
6    const wishlist = useAppSelector((state) => state.wishlist);
7
8
9    return (
10     <section className="w-full py-12 mt-10 md:py-14">
11       <div className="max-w-6xl mx-auto px-6 sm:px-6 lg:px-8">
12         <div className="w-full flex flex-col md:flex-row gap-8 justify-between">
13           <div className="w-full lg:w-[80%] flex flex-col space-y-20">
14             {wishlist.length > 0 ? (
15               wishlist.map((e) => (
16                 <Wishlistcard
17                   key={e._id}
18                   id={e._id}
19                   title={e.title}
20                   price={e.price}
21                   inventory={e.inventory}
22                   image={e.image}
23                   description={e.description}
24                 />
25               ))
26             ) : (
27               <div className="text-center text-gray-500 text-2xl">
28                 Your Wishlist is empty.
29               </div>
30             )}
31           </div>
32         </div>
33       </div>
34     </section>
35   );
36 }
```

7.Category Filters:

- Created a dropdown menu for category selection.
- Used dynamic routing to navigate to /category.
- Fetched products from Sanity CMS by matching the selected category slug.
- Updated the product listing page dynamically based on the selected category.



Code snippet:

```

File Edit Selection View Go Marketplace (Administrator)
Welcome | detail.tsx | cart.tsx | wishlist.tsx | page.tsx x
src > app > [category] > page.tsx > Categorypage
1 import { fetchProductCategory } from '@utils/productCategory';
2 import React from 'react';
3 import { productType } from '@utils/type';
4 import ProductCard from '@components/productcard';
5
6
7 async function Categorypage({params}:{params:Promise<category:string>}) {
8   const categoryparam = await params;
9   const [category]=categoryparam;
10  const product:productType[] = await fetchProductCategory(category);
11
12  const categoryName = category.replace("-", " ");
13
14  return
15  < >
16  <section className="py-14">
17    <div className="max-w-6xl mx-auto px-4 sm:px-6 lg:px-8">
18      <div className="w-full py-1 flex items-center justify-center mb-2 md:mb-15">
19        <span className="text-[24px] md:text-5xl font-semibold text-color1 capitalize">
20          {categoryName}
21        </span>
22      </div>
23
24      <div className="py-4 px-[9px] sm:px-2 md:px-8 w-full flex flex-row flex-wrap gap-x-2 sm:gap-x-12 gap-y-4">
25        {product.map((e) => (
26
27          <ProductCard
28            key={e._id}
29            _id={e._id}
30            title={e.title}
31            price={e.price}
32            badge={e.badge || ""}
33            image={e.image}
34            priceWithoutDiscount={e.priceWithoutDiscount}
  
```

Challenges Faced and Solutions Implemented:

1) State Management Complexity:

Managing cart and wishlist states are challenging while developing dynamic components.

Solution: Divided state slices into logical modules (e.g., cart, wishlist) for better maintainability.

2) Category Filter Routing:

Incorrect route handling during category filtering.

Solution: Validated category names before fetching data.

Best Practices Followed:

- Used reusable components like ProductCard to ensure consistency.
- Adopted Redux Toolkit for efficient and scalable state management.
- Build helper functions while fetching data from sanity CMS and used in components.
- Followed best practices for code structure, such as modular files and clear naming conventions.