

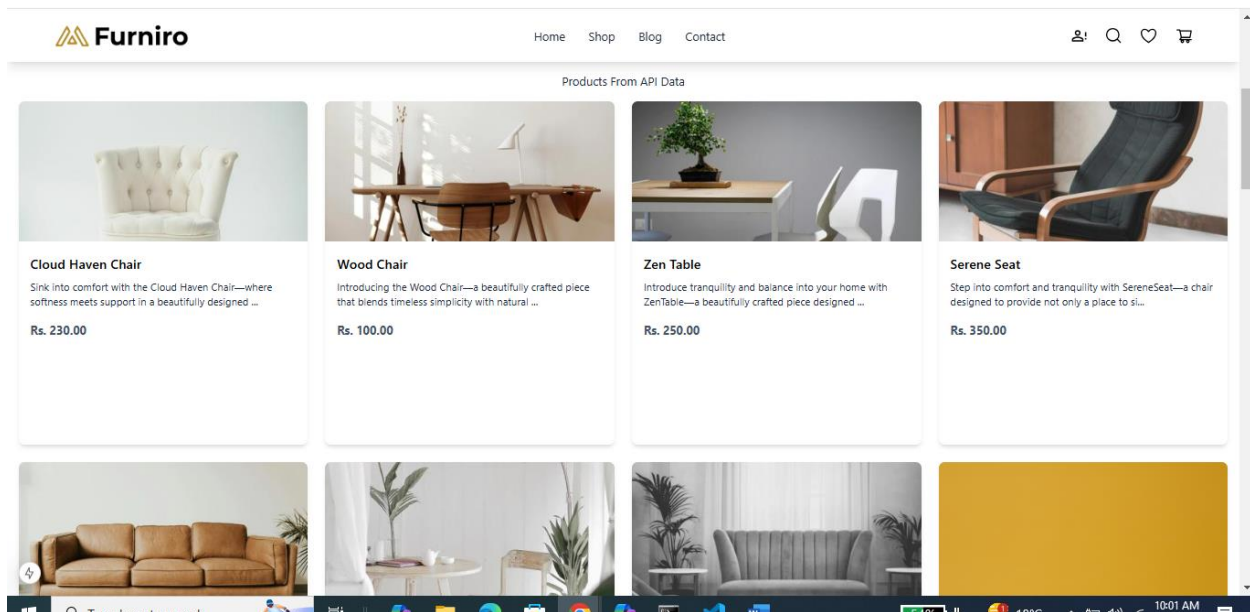
HACKATHON DAY 04:

BUILDING DYNAMIC FRONTEND COMPONENTS FOR OUR


MARKETPLACE:

FUNCTIONAL DELIEVERABLES:





1. Dynamic Data Product Listing:



2. Product Detail Pages Individually:




[Home](#) [Shop](#) [Blog](#) [Contact](#)

Product Details

[Shop](#) > [Product Details](#)




Sunny Chic

Price: \$400

Description: Embrace the warmth of style with SunnyChic—a vibrant and chic contemporary collection designed to bring the cheerful essence of sunshine and chic elegance to your home. Whether you're brightening up a living room, bedroom, or outdoor space, SunnyChic vibrates every corner with a refreshing burst of energy and a touch of sophisticated charm. Inspired by the warmth of sunny days and the laid-back yet stylish vibe of coastal living, SunnyChic features bold colors, light fabrics, and breezy designs that capture the spirit of summer all year round. From sunny yellows and soft neutrals to playful patterns and textures, this collection effortlessly combines comfort with mesmerizing design, creating spaces that feel both lively and inviting. Crafted from high-quality materials, SunnyChic is designed for those who appreciate a modern and cheerful aesthetic without sacrificing comfort. The collection offers a perfect balance of casual elegance and bright appeal, making it an ideal choice for those who love to incorporate light, airy tones and a touch of playfulness into their decor. Key features: Bright, bold colors and breezy designs inspired by sunny, coastal living. High-quality materials that are both durable and comfortable. A versatile collection perfect for living rooms, bedrooms, or outdoor spaces. Combines chic style with a relaxed, welcoming atmosphere ideal for those looking to add a refreshing, cheerful touch to their home. Fill your home with the light and energy of SunnyChic—where sunshine meets style for a lively and sophisticated living experience. Make every day feel like a sunlit escape.

Quantity:

[Add to Cart](#)




Marble Ease


Price: \$419


Description: Introducing MarbleEase—a luxurious collection that brings the timeless elegance of marble into your home with ease and sophistication. Designed for those who appreciate understated beauty and high-end design, MarbleEase combines the natural allure of marble with modern functionality, creating an effortless balance between style and practicality. Each piece in the MarbleEase collection features the exquisite veining and refined textures that make marble a classic choice, while offering lightweight and durable alternatives that ensure easy maintenance and long-lasting appeal. Whether you are adding a statement piece to your living room, kitchen, or office, MarbleEase brings a touch of opulence and tranquility to any space. From sleek tabletops and chic home accessories to sophisticated decor accents, MarbleEase infuses your home with a sense of luxury and simplicity. Its neutral tones and classic design complement a wide variety of interior styles, from modern minimalist to more traditional settings. Perfect for those who value timeless design with a contemporary twist, MarbleEase offers elegance without the hassle. Key features: Timeless marble-inspired design with beautiful veining and texture. High-quality, easy-to-care-for materials that offer durability and longevity. Lightweight and functional, making it ideal for daily use and effortless maintenance. Versatile pieces that complement a wide range of interior styles. Perfect for adding an elegant, luxurious touch to any room. Elevate your home with MarbleEase—where luxury meets ease, and timeless design is made effortlessly modern. Create a space that reflects your love for beauty, simplicity, and sophistication.


Quantity:

[Add to Cart](#)

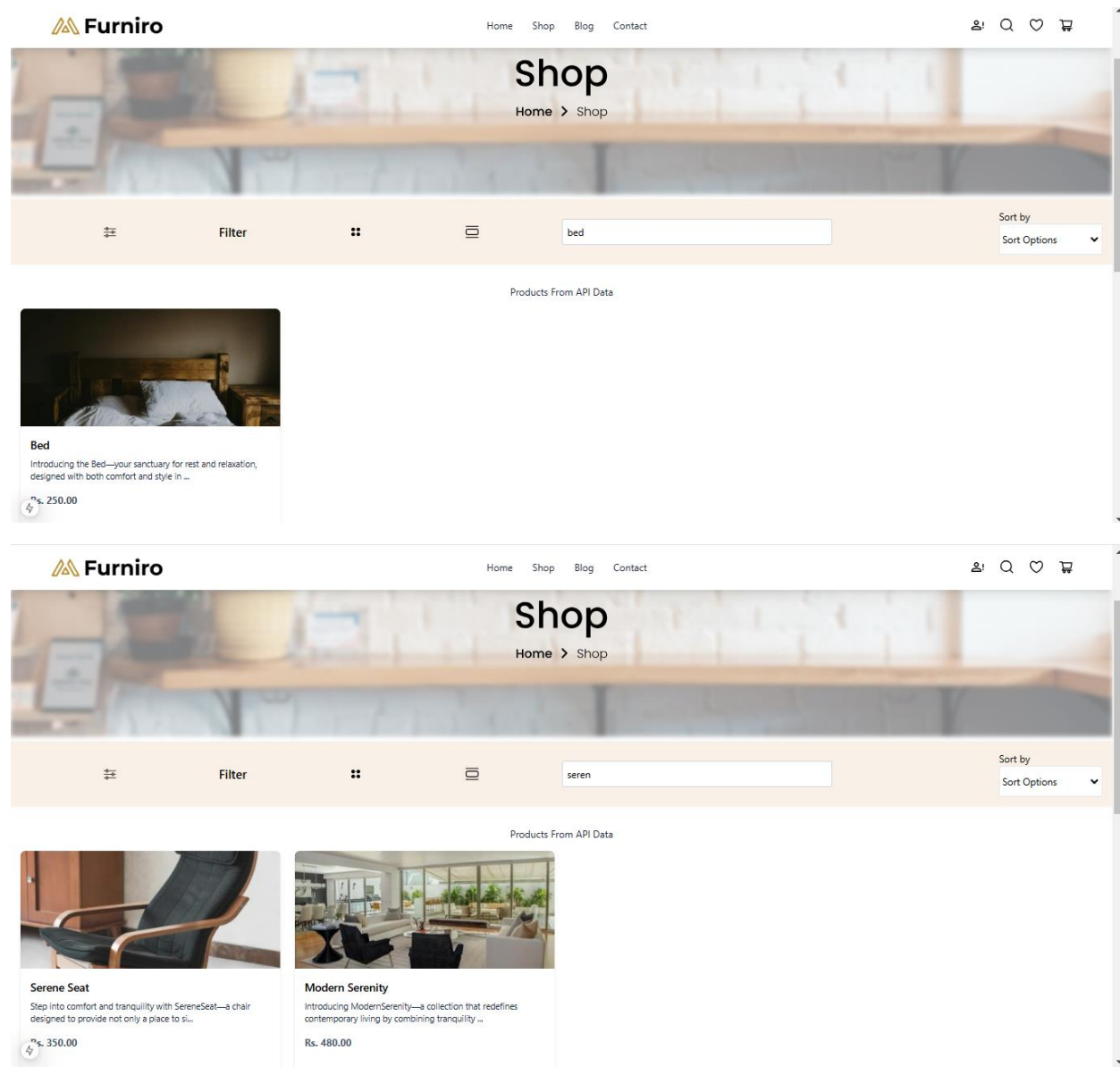
 **High Quality**
crafted from top materials

 **Warranty Protection**
Over 2 years

 **Free Shipping**
Order over 150 \$

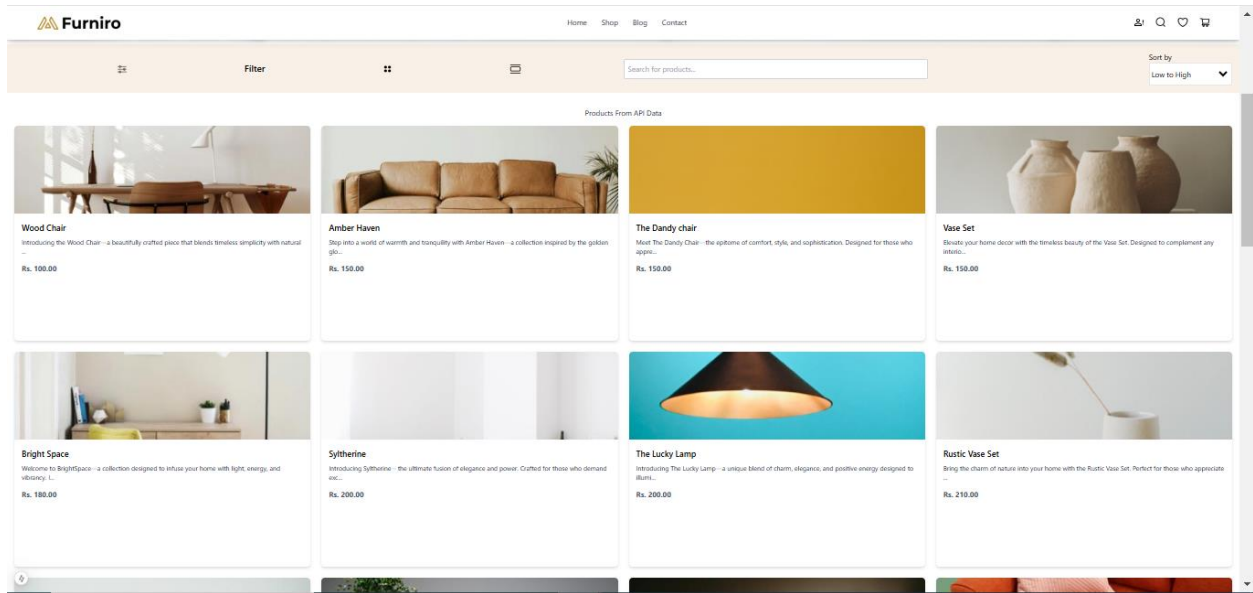
 **24 / 7 Support**
Dedicated support

3. Product Search Bar:




4. Sorting Product Options:

o) Low to High Price Wise



o) A to Z & Z to A



[Home](#) [Shop](#) [Blog](#) [Contact](#)


Filter

Search for products...


Sort by

Sort by A-Z


Products From API Data




Amber Haven
Step into a world of warmth and tranquility with Amber Haven – a collection inspired by the golden glow...
Rs. 150.00




Bed
Introducing the Bed – your sanctuary for rest and relaxation, designed with both comfort and style in mind...
Rs. 250.00




Bold Nest
Welcome to BoldNest – where fearless design meets comfort and creativity. Crafted for those who embrace...
Rs. 260.00




Bright Space
Welcome to BrightSpace – a collection designed to infuse your home with light, energy, and vibrancy...
Rs. 180.00




Cloud Haven Chair
Sink into comfort with the Cloud Haven Chair – where softness meets support in a beautifully designed...
Rs. 230.00




Marble Ease
Introducing MarbleEase – a luxurious collection that brings the timeless elegance of marble into your...
Rs. 419.00



Modern Serenity
Introducing ModernSerenity – a collection that redefines contemporary living by combining...
Rs. 480.00



Nordic Elegance
Elevate your space with Nordic Elegance – a collection that brings the minimalist beauty and understated...
Rs. 280.00



[Home](#) [Shop](#) [Blog](#) [Contact](#)


Filter

Search for products...


Sort by

Sort by Z-A


Products From API Data




Zen Table
Introduce tranquility and balance into your home with Zen Table – a beautifully crafted piece designed...
Rs. 150.00




Wood Chair
Introducing the Wood Chair – a beautifully crafted piece that blends timeless simplicity with natural...
Rs. 100.00




Vase Set
Elevate your home decor with the timeless beauty of the Vase Set. Designed to complement any...
Rs. 150.00




Tropical Vibe
Escape to paradise with TropicalVibe – a collection that brings the vibrant energy of the tropics...
Rs. 550.00




Timeless Elegance
Introducing TimelessElegance – a collection that embodies the perfect fusion of classic beauty and...
Rs. 320.00



Timber Craft
Introducing TimberCraft – a collection that celebrates the timeless beauty of wood craftsmanship...
Rs. 320.00



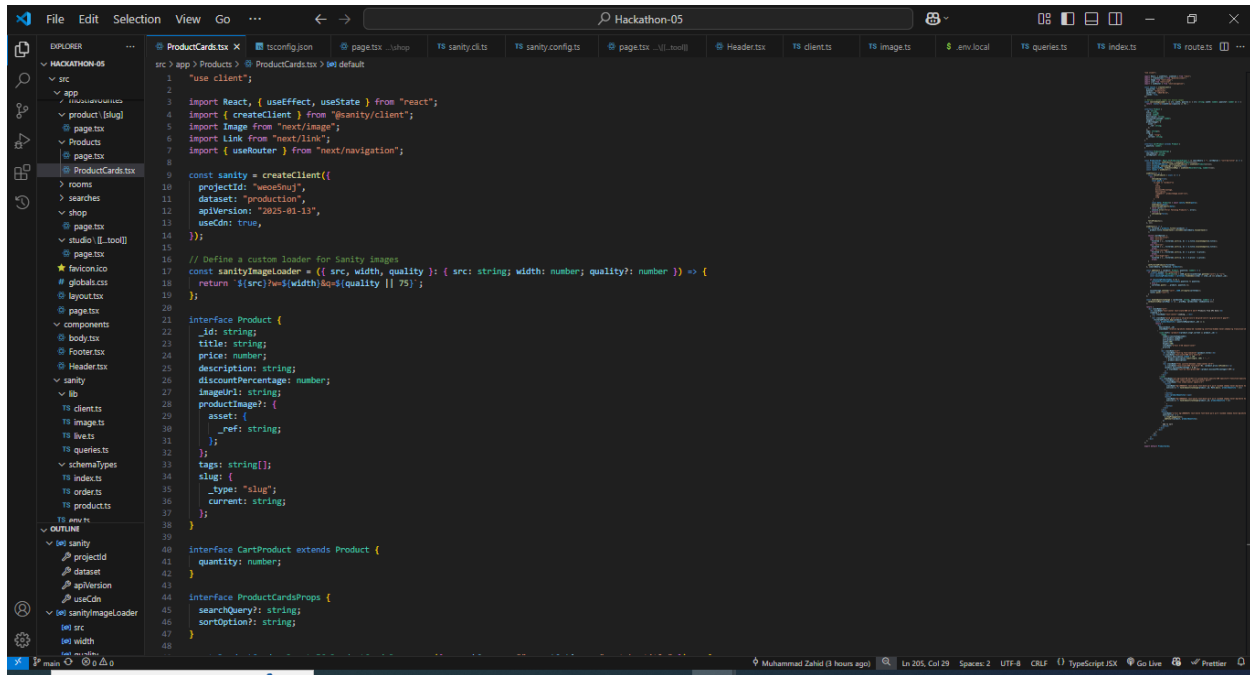
The Lucky Lamp
Introducing The Lucky Lamp – a unique blend of charm, elegance, and positive energy designed to...
Rs. 200.00



The Dandy chair
Meet The Dandy Chair – the epitome of comfort, style, and sophistication. Designed for those who...
Rs. 150.00

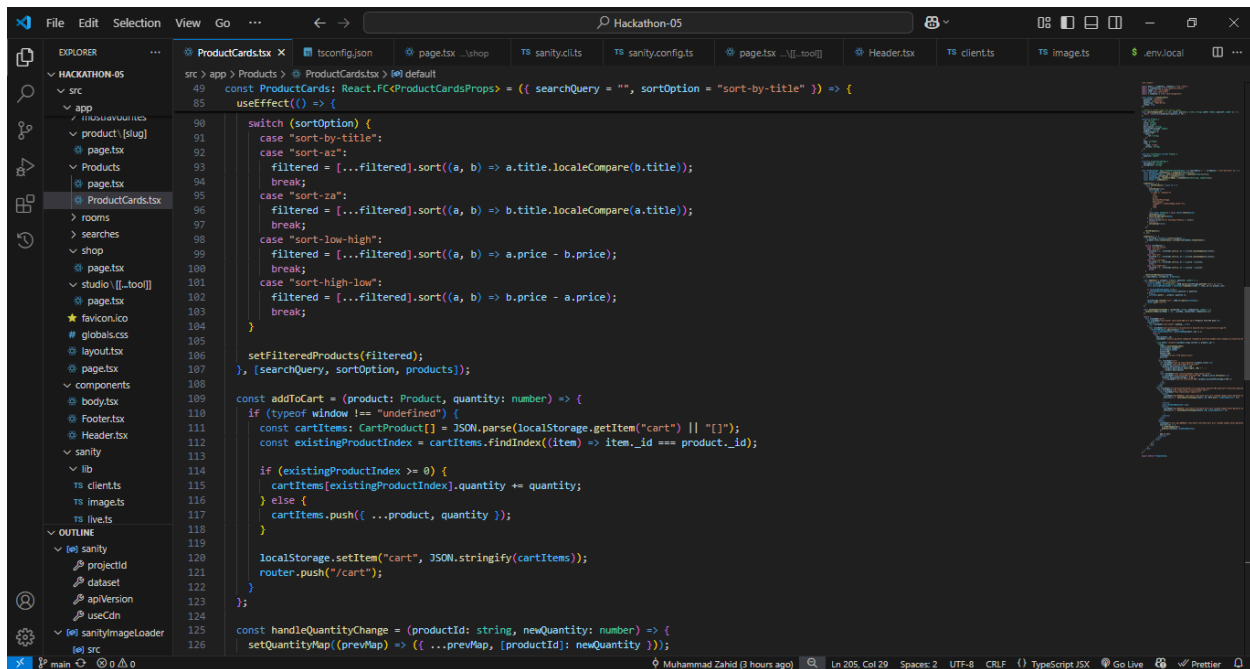
CODE DELIEVERABLES:

1. ProductCard.tsx



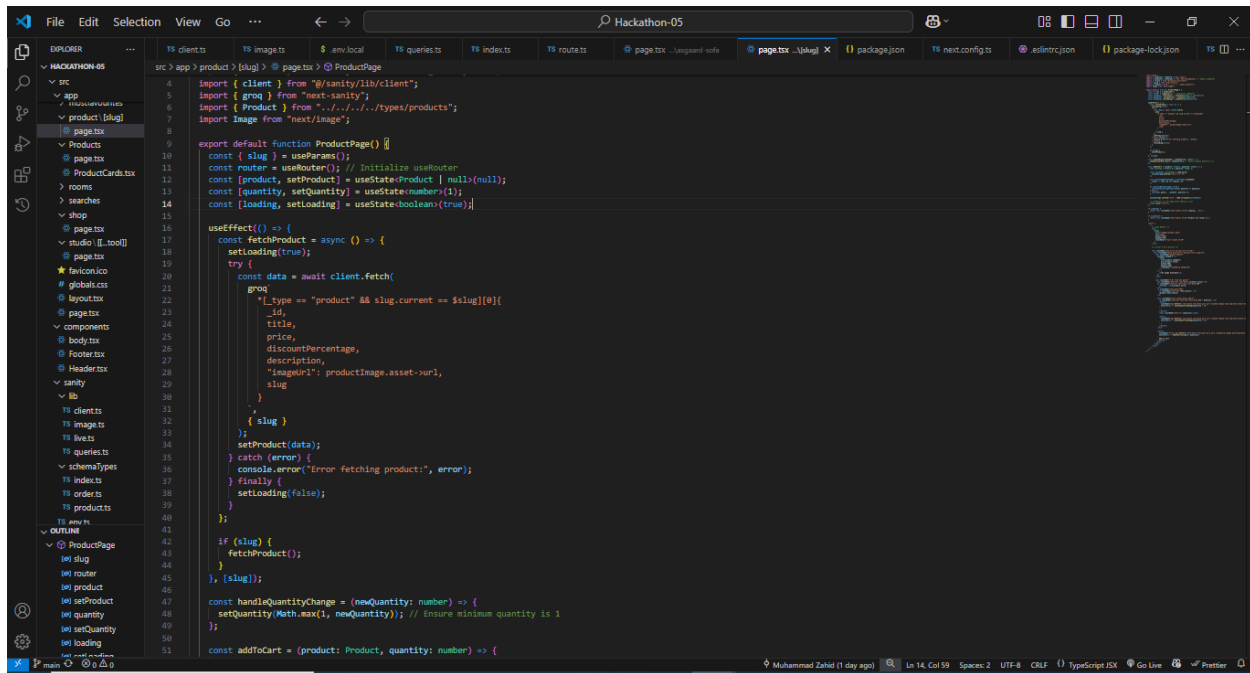
```
src > app > Products > ProductCards.tsx > @ default
1  "use client";
2
3  import React, { useEffect, useState } from "react";
4  import { createClient } from "@sanity/client";
5  import Image from "next/image";
6  import Link from "next/link";
7  import { useRouter } from "next/navigation";
8
9  const sanity = createClient({
10    projectId: "weoe5nu",
11    dataset: "production",
12    apiVersion: "2023-01-13",
13    useCdn: true,
14  });
15
16  // Define a custom loader for sanity images
17  const sanityImageLoader = ({ src, width, quality }: { src: string; width: number; quality?: number }) => {
18    return `${src}?w=${width}&q=${quality || 75}`;
19  };
20
21  interface Product {
22    _id: string;
23    title: string;
24    price: number;
25    description: string;
26    discountPercentage: number;
27    imageUrl: string;
28    productImage: {
29      assets: {
30        _ref: string;
31      };
32    };
33    tags: string[];
34    slug: {
35      _type: "slug";
36      current: string;
37    };
38  };
39
40  interface CartProduct extends Product {
41    quantity: number;
42  };
43
44  interface ProductCardsProps {
45    searchQuery?: string;
46    sortOption?: string;
47  };
48
49  const ProductCards: React.FC<ProductCardsProps> = ({ searchQuery = "", sortOption = "sort-by-title" }) => {
50    const [products, setProducts] = useState<Product[]>([]);
51    const [filteredProducts, setFilteredProducts] = useState<Product[]>([]);
52    const [searchQueryInput, setSearchQueryInput] = useState<string>("");
53    const [sortOption, setSortOption] = useState<string>("sort-by-title");
54    const [isLoading, setIsLoading] = useState<boolean>(true);
55    const [error, setError] = useState<string>("");
56    const router = useRouter();
57
58    useEffect(() => {
59      const fetchProducts = async () => {
60        try {
61          const query = searchQuery ? `*[_type == 'product' & ${searchQuery}]` : `*[_type == 'product']`;
62          const products = await sanity.fetch(query);
63          setProducts(products);
64          setFilteredProducts(products);
65          setIsLoading(false);
66        } catch (error) {
67          setError("Failed to fetch products");
68        }
69      };
70
71      fetchProducts();
72    }, [searchQuery]);
73
74    const handleSearchChange = (e: React.ChangeEvent<HTMLInputElement>) => {
75      setSearchQueryInput(e.target.value);
76    };
77
78    const handleSortChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
79      setSortOption(e.target.value);
80    };
81
82    const filterProducts = () => {
83      let filtered = [...products];
84
85      switch (sortOption) {
86        case "sort-by-title":
87          filtered = [...filtered].sort((a, b) => a.title.localeCompare(b.title));
88          break;
89        case "sort-az":
90          filtered = [...filtered].sort((a, b) => b.title.localeCompare(a.title));
91          break;
92        case "sort-low-high":
93          filtered = [...filtered].sort((a, b) => a.price - b.price);
94          break;
95        case "sort-high-low":
96          filtered = [...filtered].sort((a, b) => b.price - a.price);
97          break;
98      }
99
100      setFilteredProducts(filtered);
101    };
102
103    const addToCart = (product: Product, quantity: number) => {
104      if (typeof window !== "undefined") {
105        const cartItems: CartProduct[] = JSON.parse(localStorage.getItem("cart") || "[]");
106        const existingProductIndex = cartItems.findIndex((item) => item._id === product._id);
107
108        if (existingProductIndex >= 0) {
109          cartItems[existingProductIndex].quantity += quantity;
110        } else {
111          cartItems.push({ ...product, quantity });
112        }
113
114        localStorage.setItem("cart", JSON.stringify(cartItems));
115        router.push("/cart");
116      }
117    };
118
119    const handleQuantityChange = (productId: string, newQuantity: number) => {
120      setQuantityMap((prevMap) => ({ ...prevMap, [productId]: newQuantity }));
121    };
122  };
123
124  return (
125    <div>
126      <div>
127        <input type="text" value={searchQueryInput} />
128        <select value={sortOption}>
129          <option value="sort-by-title">Sort by Title</option>
130          <option value="sort-az">Sort A-Z</option>
131          <option value="sort-low-high">Sort Low to High</option>
132          <option value="sort-high-low">Sort High to Low</option>
133        </select>
134      </div>
135      <div>
136        {isLoading ? <div>Loading...</div> : <div>
137          {error ? <div>{error}</div> : <div>
138            {filteredProducts.map((product) => (
139              <ProductCard
140                key={product._id}
141                product={product}
142                addToCart={addToCart}
143                handleQuantityChange={handleQuantityChange}
144              />
145            ))}
146          </div>
147        </div>
148      </div>
149    </div>
150  );
151
152  export default ProductCards;
```

2. Sort & Search Products:



```
src > app > Products > ProductCards.tsx > @ default
49  const ProductCards: React.FC<ProductCardsProps> = ({ searchQuery = "", sortOption = "sort-by-title" }) => {
50    const [products, setProducts] = useState<Product[]>([]);
51    const [filteredProducts, setFilteredProducts] = useState<Product[]>([]);
52    const [searchQueryInput, setSearchQueryInput] = useState<string>("");
53    const [sortOption, setSortOption] = useState<string>("sort-by-title");
54    const [isLoading, setIsLoading] = useState<boolean>(true);
55    const [error, setError] = useState<string>("");
56    const router = useRouter();
57
58    useEffect(() => {
59      const fetchProducts = async () => {
60        try {
61          const query = searchQuery ? `*[_type == 'product' & ${searchQuery}]` : `*[_type == 'product']`;
62          const products = await sanity.fetch(query);
63          setProducts(products);
64          setFilteredProducts(products);
65          setIsLoading(false);
66        } catch (error) {
67          setError("Failed to fetch products");
68        }
69      };
70
71      fetchProducts();
72    }, [searchQuery]);
73
74    const handleSearchChange = (e: React.ChangeEvent<HTMLInputElement>) => {
75      setSearchQueryInput(e.target.value);
76    };
77
78    const handleSortChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
79      setSortOption(e.target.value);
80    };
81
82    const filterProducts = () => {
83      let filtered = [...products];
84
85      switch (sortOption) {
86        case "sort-by-title":
87          filtered = [...filtered].sort((a, b) => a.title.localeCompare(b.title));
88          break;
89        case "sort-az":
90          filtered = [...filtered].sort((a, b) => b.title.localeCompare(a.title));
91          break;
92        case "sort-low-high":
93          filtered = [...filtered].sort((a, b) => a.price - b.price);
94          break;
95        case "sort-high-low":
96          filtered = [...filtered].sort((a, b) => b.price - a.price);
97          break;
98      }
99
100      setFilteredProducts(filtered);
101    };
102
103    const addToCart = (product: Product, quantity: number) => {
104      if (typeof window !== "undefined") {
105        const cartItems: CartProduct[] = JSON.parse(localStorage.getItem("cart") || "[]");
106        const existingProductIndex = cartItems.findIndex((item) => item._id === product._id);
107
108        if (existingProductIndex >= 0) {
109          cartItems[existingProductIndex].quantity += quantity;
110        } else {
111          cartItems.push({ ...product, quantity });
112        }
113
114        localStorage.setItem("cart", JSON.stringify(cartItems));
115        router.push("/cart");
116      }
117    };
118
119    const handleQuantityChange = (productId: string, newQuantity: number) => {
120      setQuantityMap((prevMap) => ({ ...prevMap, [productId]: newQuantity }));
121    };
122  };
123
124  return (
125    <div>
126      <div>
127        <input type="text" value={searchQueryInput} />
128        <select value={sortOption}>
129          <option value="sort-by-title">Sort by Title</option>
130          <option value="sort-az">Sort A-Z</option>
131          <option value="sort-low-high">Sort Low to High</option>
132          <option value="sort-high-low">Sort High to Low</option>
133        </select>
134      </div>
135      <div>
136        {isLoading ? <div>Loading...</div> : <div>
137          {error ? <div>{error}</div> : <div>
138            {filteredProducts.map((product) => (
139              <ProductCard
140                key={product._id}
141                product={product}
142                addToCart={addToCart}
143                handleQuantityChange={handleQuantityChange}
144              />
145            ))}
146          </div>
147        </div>
148      </div>
149    </div>
150  );
151
152  export default ProductCards;
```

3. Product[slug] Fetching Products:



Steps Taken to Build and Integrate Components

1. Setting Up Sanity Client in a Next.js App

- Install the required dependency:
- Configure the **Sanity client** in a separate file (sanityClient.ts):

2. Fetching Products from Sanity

- Query the product data using **GROQ (Graph-Relational Object Queries)**:
- The query fetches all products with **ID, name, price, description, and image URL**.

3. Writing Fetched Data into a File

- **Convert** the fetched JSON data into a string and **write it into a file** (products.json):
- This script ensures the **product data is stored locally** for backup or further processing.

Challenges Faced and Solutions Implemented

1. Handling Large Data Fetching

- **Challenge:** If there are many products, fetching them all at once may **slow down performance**.
- **Solution:** Implement **pagination** using Sanity's limit and start parameter

typescript

CopyEdit

```
const query = `*_type == "product" | order(_createdAt desc) [0...50]`;
```

This fetches only **50 products at a time**, reducing load time.

2. Image URL Resolution

- **Challenge:** Sanity stores image references, not direct URLs.
- **Solution:** Used asset->url in the query to resolve the **actual image URL**:

groq

CopyEdit

```
"imageUrl": images[0].asset->url
```

3. Handling API Rate Limits

- **Challenge:** Too many requests in a short time may **trigger rate limits**.
- **Solution:** Used setTimeout and **batched requests** if needed:

4. File Writing Errors

- **Challenge:** The file might **fail to save** if the script doesn't have write permissions.
- **Solution:** Used a **try-catch block** to handle errors:

Conclusion

This implementation ensures **efficient fetching** of product data from Sanity and **stores it in a file**. The process optimizes performance, handles API constraints, and improves maintainability by structuring the code with error handling.

“HACKATHON DAY 04 ENDS HERE THANK U”