

Create Dynamic Components and Dynamic Shop Pages and Dynamic Api routes and integrate with sanity then pass the data through api to pages and show on frontend.

Components Created

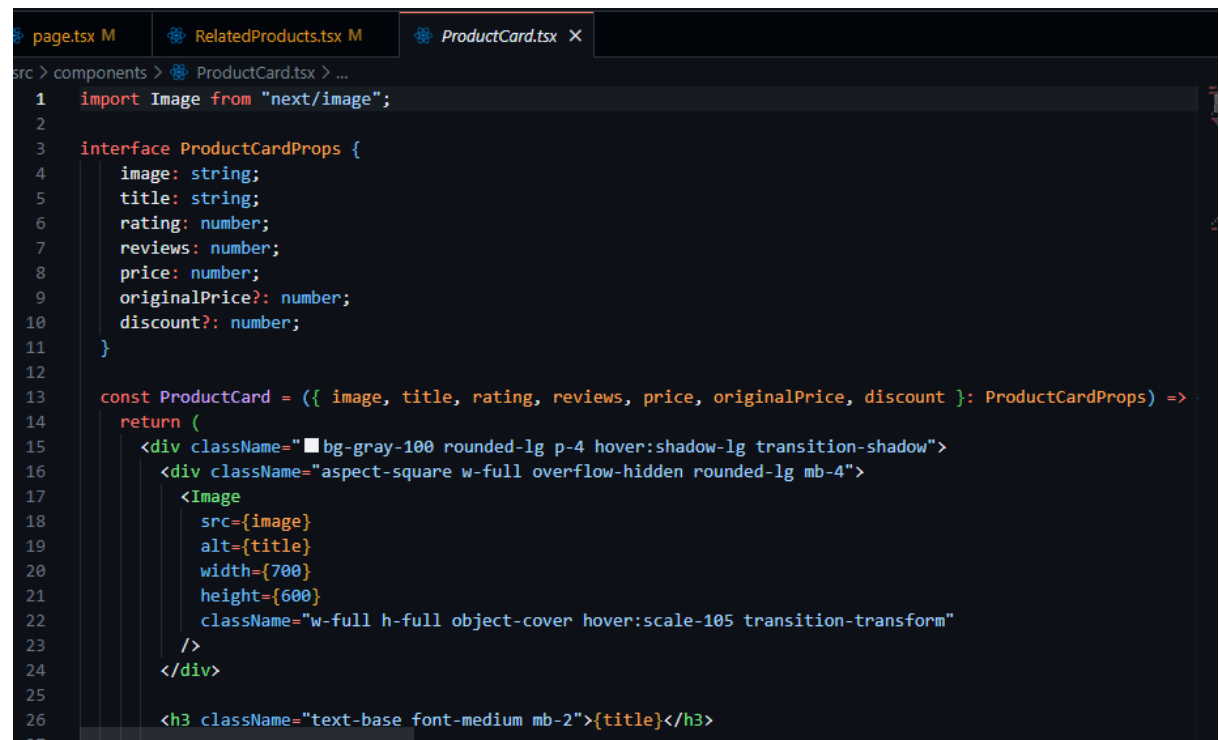
1. Product Card
2. Filter Bar
3. Pagination
4. Related Products
5. Product Reviews

Dynamic Shop pages

Dynamic Api Routes

And other related pages

Product Card



```
1 import Image from "next/image";
2
3 interface ProductCardProps {
4   image: string;
5   title: string;
6   rating: number;
7   reviews: number;
8   price: number;
9   originalPrice?: number;
10  discount?: number;
11 }
12
13 const ProductCard = ({ image, title, rating, reviews, price, originalPrice, discount }: ProductCardProps) =>
14   return (
15     <div className="bg-gray-100 rounded-lg p-4 hover:shadow-lg transition-shadow">
16       <div className="aspect-square w-full overflow-hidden rounded-lg mb-4">
17         <Image
18           src={image}
19           alt={title}
20           width={700}
21           height={600}
22           className="w-full h-full object-cover hover:scale-105 transition-transform"
23         />
24       </div>
25       <h3 className="text-base font-medium mb-2">{title}</h3>
26     </div>
27   )
```

Product Listing

```
page.tsx M X
src > app > shop > page.tsx > CategoryPage
1  "use client";
2
3  import React, { useState, useEffect } from "react";
4  import FilterBar from "@components/FilterBar";
5  import Pagination from "@components/Pagination";
6  import ProductCard from "@components/ProductCard";
7  import Link from "next/link"; // Import Link from Next.js
8
9  const CategoryPage = () => {
10   const [products, setProducts] = useState<any[]>([]);
11   const [filteredProducts, setFilteredProducts] = useState<any[]>([]);
12   const [categories] = useState(["T-Shirts", "Hoodies", "Jeans", "Shirts"]);
13   const [colors] = useState(["red", "blue", "green", "black", "white"]);
14   const [sizes] = useState(["S", "M", "L", "XL"]);
15   const [priceRange, setPriceRange] = useState<[number, number]>([0, 1000]);
16   const [selectedCategory, setSelectedCategory] = useState("");
17   const [selectedColor, setSelectedColor] = useState("");
18   const [selectedSize, setSelectedSize] = useState("");
19   const [selectedPrice, setSelectedPrice] = useState(1000);
20   const [currentPage, setCurrentPage] = useState(1);
21   const productsPerPage = 12;
22
23   // used a useEffect to fetch data from sanity through api route.
24   useEffect(() => {
25     const fetchProducts = async () => {
26       const res = await fetch("/api/products");
27     }
28   });
```

```
page.tsx M X
src > app > shop > page.tsx > CategoryPage
9  const CategoryPage = () => {
23   // used a useEffect to fetch data from sanity through api route.
24   useEffect(() => {
25     const fetchProducts = async () => {
26       const res = await fetch("/api/products");
27       const data = await res.json();
28       setProducts(data);
29       setFilteredProducts(data);
30     };
31     fetchProducts();
32   }, []);
33
34   const applyFilter = () => {
35     const filtered = products.filter(
36       (product) =>
37         (!selectedCategory || product.category === selectedCategory) &&
38         (!selectedColor || product.colors.includes(selectedColor)) &&
39         (!selectedSize || product.sizes.includes(selectedSize)) &&
40         product.price <= selectedPrice
41     );
42     setFilteredProducts(filtered);
43     setCurrentPage(1);
44   };
45
46   const totalPages = Math.ceil(filteredProducts.length / productsPerPage);
47   const currentProducts = filteredProducts.slice(
48     (currentPage - 1) * productsPerPage,
49     (currentPage) * productsPerPage
50   );
```

```
page.tsx M X
src > app > shop > page.tsx > CategoryPage
9  const CategoryPage = () => {
52  return (
53    <div className="container mx-auto flex flex-col lg:flex-row gap-6 px-4 py-8">
54      <aside className="w-full lg:w-1/4">
55        <FilterBar
56          categories={categories}
57          colors={colors}
58          sizes={sizes}
59          priceRange={priceRange}
60          selectedCategory={selectedCategory}
61          setSelectedCategory={setSelectedCategory}
62          selectedColor={selectedColor}
63          setSelectedColor={setSelectedColor}
64          selectedSize={selectedSize}
65          setSelectedSize={setSelectedSize}
66          selectedPrice={selectedPrice}
67          setSelectedPrice={setSelectedPrice}
68          applyFilter={applyFilter}
69        />
70      </aside>
71      <section className="w-full lg:w-3/4">
72        <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
73          {currentProducts.map((product) => (
74            <Link key={product._id} href={`~/shop/${product._id}`}>
75              <ProductCard
76                image={product.imageUrl}
77                title={product.name}
78                price={product.price}
79                description={product.description}
80              />
81            </Link>
82          ))}
83        </div>
84      </section>
85    </div>
86  )
87}
```

Product Detail

```
page.tsx M productdetail.tsx X
src > app > shop > [id] > productdetail.tsx > ...
1  "use client";
2
3  import React, { useEffect, useState } from "react";
4  import { useParams } from "next/navigation";
5  import Image from "next/image";
6  import { urlFor } from "@sanity/lib/image";
7
8  const ProductDetailPage: React.FC = () => {
9    const { id } = useParams();
10    const [product, setProduct] = useState<any>(null);
11    const [loading, setLoading] = useState<boolean>(true);
12    const [error, setError] = useState<string>("");
13
14    useEffect(() => {
15      const fetchProduct = async () => {
16        try {
17          const res = await fetch(`/api/products/${id}`);
18          if (!res.ok) throw new Error("Failed to fetch product details");
19          const data = await res.json();
20          setProduct(data);
21        } catch (err: any) {
22          setError(err.message);
23        } finally {
24          setLoading(false);
25        }
26      };
27    });
28  }
```

Filter bar

```
src > components > FilterBar.tsx > ...
1  "use client";
2  import React from "react";
3
4  interface FilterBarProps {
5    categories: string[];
6    colors: string[];
7    sizes: string[];
8    priceRange: [number, number];
9    selectedCategory: string;
10   setSelectedCategory: (category: string) => void;
11   selectedColor: string;
12   setSelectedColor: (color: string) => void;
13   selectedSize: string;
14   setSelectedSize: (size: string) => void;
15   selectedPrice: number;
16   setSelectedPrice: (price: number) => void;
17   applyFilter: () => void;
18 }
19
20 const FilterBar: React.FC<FilterBarProps> = ({
21   categories,
22   colors,
23   sizes,
24   priceRange,
25   selectedCategory,
26   setSelectedCategory,
```

```
src > components > FilterBar.tsx > ...
20   const FilterBar: React.FC<FilterBarProps> = ({
21     applyFilter,
22   }) => (
23     <div className="bg-white p-4 border rounded-lg shadow-md">
24       <h2 className="text-lg font-semibold mb-4">Filters</h2>
25       <div className="mb-4">
26         <h3 className="font-semibold mb-2">Category</h3>
27         {categories.map((category) => (
28           <div
29             key={category}
30             className={`cursor-pointer py-1 ${
31               selectedCategory === category ? "font-bold" : ""
32             }`}
33             onClick={() => setSelectedCategory(category)}
34           >
35             {category}
36           </div>
37         ))}
38       </div>
39       <div className="mb-4">
40         <h3 className="font-semibold mb-2">Price</h3>
41         <input
42           type="range"
43           min={priceRange[0]}
44           max={priceRange[1]}
45           value={selectedPrice}
46           onChange={(e) => setSelectedPrice(Number(e.target.value))}
47         />
48       </div>
49     </div>
50   );
51 }
```

Pagination

```
page.tsx M  FilterBar.tsx M  Pagination.tsx M X
src > components > Pagination.tsx > [e] Pagination
1  "use client";
2  import React from "react";
3
4  interface PaginationProps {
5    currentPage: number;
6    totalPages: number;
7    onPageChange: (page: number) => void;
8  }
9
10 const Pagination: React.FC<PaginationProps> = ({
11   currentPage,
12   totalPages,
13   onPageChange,
14 }) => {
15   <div className="mt-6 flex justify-center items-center gap-2">
16     <button
17       className="px-3 py-1 border rounded"
18       onClick={() => onPageChange(currentPage - 1)}
19       disabled={currentPage === 1}
20     >
21       Previous
22     </button>
23     {[...Array(totalPages)].map((_, index) => (
24       <button
25         key={index}
26         className={`px-3 py-1 border rounded ${
27           currentPage === index + 1 ? "bg-black text-white" : "bg-gray-100"
28         }}
29         onClick={() => onPageChange(index + 1)}
30       >
31         {index + 1}
32       </button>
```

Reviews

```
src > components > ProductReviews.tsx > ...
1  "use client";
2
3  import React from "react";
4
5  const ProductReviews: React.FC = () => {
6    <div className="bg-white p-4 rounded-lg shadow-md">
7      <h2 className="text-lg font-bold mb-4">Customer Reviews</h2>
8      <div className="grid grid-cols-1 gap-6">
9        {Array(3).fill("").map((_, index) => (
10          <div key={index} className="p-4 border rounded shadow-sm">
11            <h3 className="font-medium">Reviewer Name</h3>
12            <p className="text-gray-600">Amazing product, loved it!</p>
13            <div className="flex gap-1 mt-2">
14              {[...Array(5)].map((_, i) => (
15                <span key={i} className="text-yellow-400">★</span>
16              ))}
17            </div>
18          </div>
19        ))}
20      </div>
21    </div>
22  );
23
24  export default ProductReviews;
25
```

Related Products

```
page.tsx M  FilterBar.tsx M  Pagination.tsx M  RelatedProducts.tsx M X
src > components > RelatedProducts.tsx > ...
1  "use client";
2
3  import React, { useState, useEffect } from "react";
4  import Link from "next/link";
5  import Image from "next/image";
6
7  export const RelatedProducts = () => {
8    const [products, setProducts] = useState<any[]>([]);
9    const [error, setError] = useState<string>("");
10
11    useEffect(() => {
12      async function fetchProducts() {
13        try {
14          const response = await fetch("/api/products");
15          if (!response.ok) throw new Error("Failed to fetch products");
16          const data = await response.json();
17
18          // Shuffle the products array randomly
19          const shuffledProducts = data.sort(() => Math.random() - 0.5);
20
21          // Select only the first 4 products
22          setProducts(shuffledProducts.slice(0, 4));
23        } catch (err: any) {
24          setError(err.message || "An error occurred");
25        }
26      }
27      fetchProducts();
28    }, []);
29
30    if (error) {
31      return <div className="text-center text-red-500">{error}</div>;
32    }
33  }
```

```
page.tsx M  FilterBar.tsx M  Pagination.tsx M  RelatedProducts.tsx M X
src > components > RelatedProducts.tsx > ...
7  export const RelatedProducts = () => {
34    return (
35      <section className="max-w-7xl mx-auto px-4 py-12">
36        <h2 className="text-3xl font-bold text-center mb-8">You Might Also Like</h2>
37
38        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-6">
39          {products.map((product) => (
40            <Link key={product.id} href={` /shop/${product.id}`}>
41              <div className="border rounded-lg p-4 shadow hover:shadow-lg transition h-[480px]">
42                <Image
43                  src={product.imageUrl}
44                  alt={product.name}
45                  width={192}
46                  height={192}
47                  className="w-full h-72 object-cover rounded"
48                />
49                <h2 className="mt-4 text-lg font-semibold">{product.name}</h2>
50                <div className="flex items-center mt-2">
51                  <span className="text-yellow-500">
52                    {"★".repeat(Math.floor(product.rating || 0))}
53                    {"☆".repeat(5 - Math.floor(product.rating || 0))}
54                  </span>
55                  <span className="ml-2 text-sm text-gray-500">{product.rating || 0}/5</span>
56                </div>
57                <div className="mt-4">
58                  <span className="text-lg font-bold">${product.price}</span>
59                  {product.originalPrice && (
60                    <span className="ml-2 text-sm line-through text-gray-500">
61                      ${product.originalPrice}
62                    </span>
63                  )}
64                </div>
65              </div>
66            </Link>
67          )}
68        </div>
69      </section>
70    );
71  }
```

API integration and dynamic routing

```
TS route.ts M X
src > app > api > products > TS route.ts > ...
1 import { client } from "@sanity/lib/client";
2 import { NextResponse } from "next/server";
3
4 export async function GET() {
5   try {
6     const data = await client.fetch(`
7       *[_type == "products"] {
8         _id,
9         name,
10        price,
11        description,
12        imageslist,
13        "imageUrl": image.asset->url,
14        category,
15        discountPercent,
16        isNew,
17        colors[],
18        sizes[]
19      }
20    `);
21
22    if (!data) {
23      return NextResponse.json({ error: "No products found" }, { status: 404 });
24    }
25
26    return NextResponse.json(data);
27  } catch (error) {
28    console.error("Error fetching products:", error);
29    return NextResponse.json({ error: "Failed to fetch products" }, { status: 500 });
30  }
31 }
32
```

Dynamic Route Integration

```
TS route.ts X
src > app > api > products > [id] > TS route.ts > ...
1 import { client } from "@sanity/lib/client";
2 import { NextResponse } from "next/server";
3
4 export async function GET(req: Request, { params }: { params: { id: string } }) {
5   const { id } = params;
6
7   try {
8     const query = `
9       *[_type == "products" && _id == ${id}][0] {
10         _id,
11         name,
12         image,
13         imageslist,
14         price,
15         description,
16         "imageUrl": image.asset->url,
17         category,
18         discountPercent,
19         isNew,
20         colors,
21         sizes
22       }
23     `;
24
25     const product = await client.fetch(query, { id });
26
27     if (!product) {
28       return NextResponse.json({ error: "Product not found" }, { status: 404 });
29     }
30
31     return NextResponse.json(product);
32   } catch (error) {
33
34   }
35 }
```

