Comforty E-Commerce Website Documentation

Day 4: Building Dynamic Frontend Components
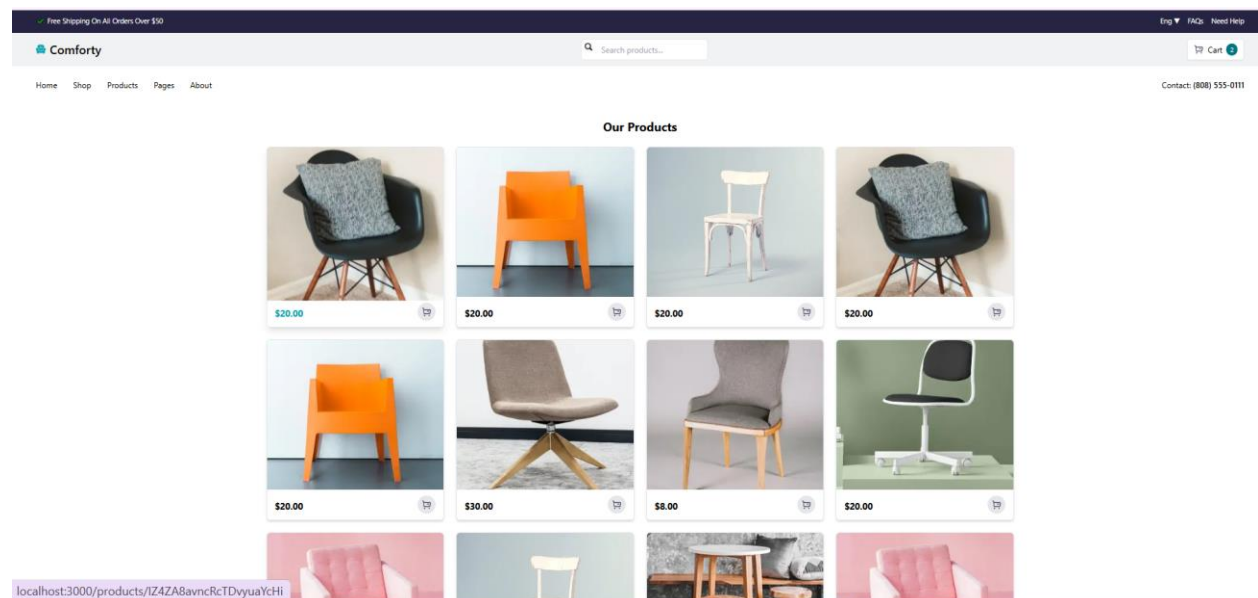
Muskaan Fayyaz

## Roll no: 00216348

1. Functional Deliverables

1.1 Product Listing Page

A dynamic product listing page where products are fetched from an API and displayed as cards with titles, prices, images, and options to add to the cart or wishlist.
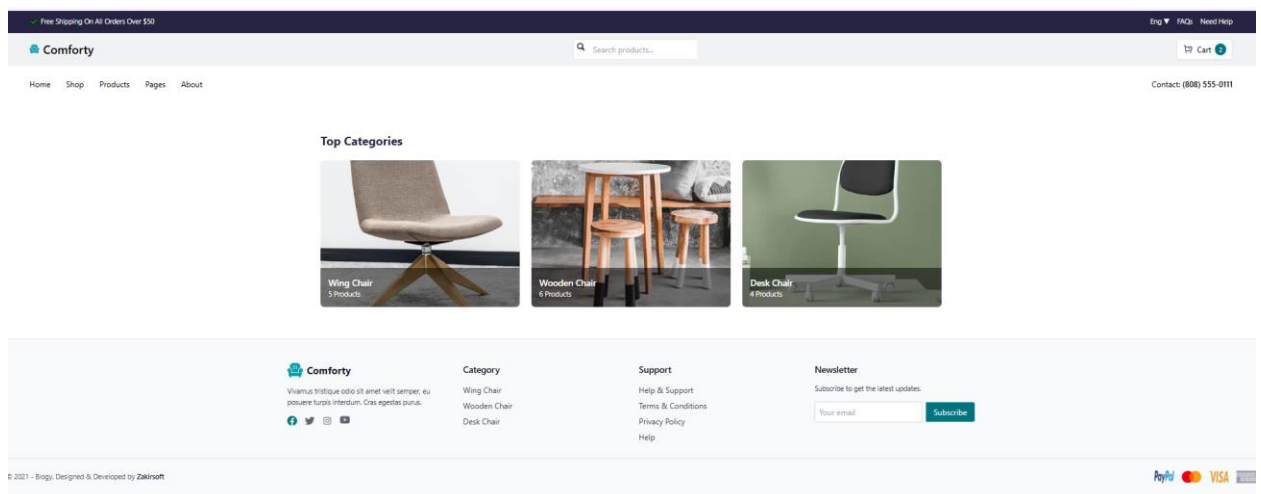

1.2 Product Detail Page

Individual product detail pages that render the full product information when the user clicks on a product card. Data is fetched dynamically based on the selected product ID via routing.
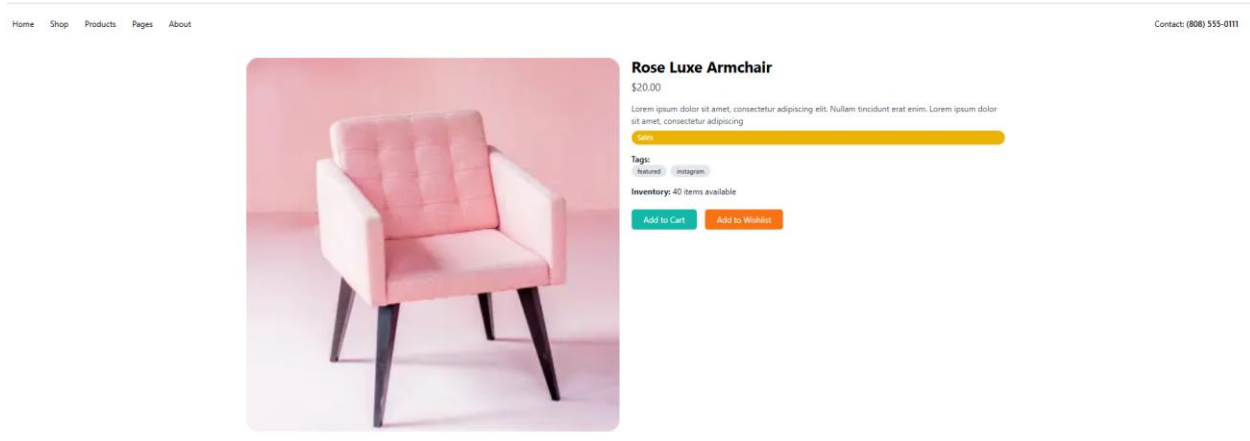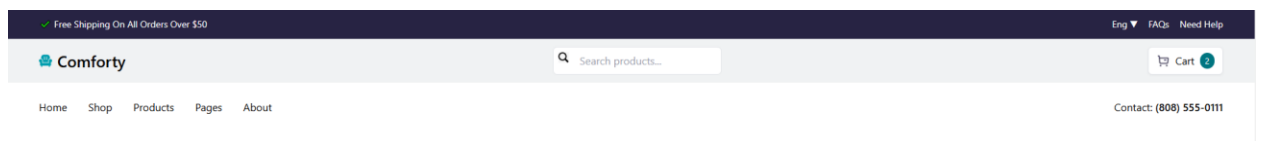
1.3 Category Filters and Search Bar

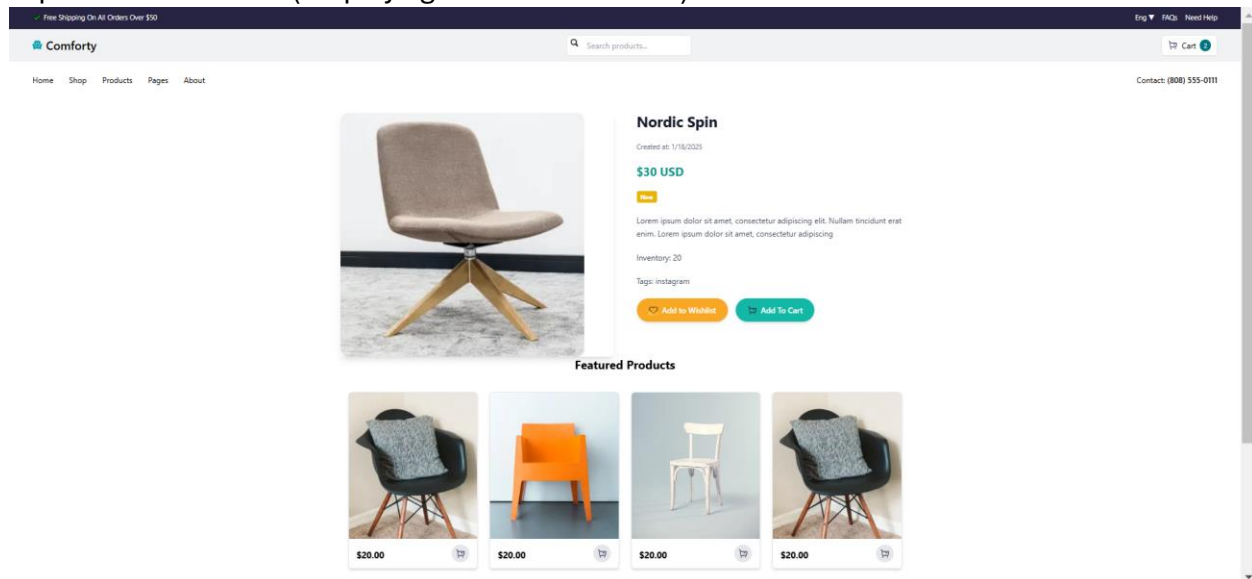- **Category Filters**: Users can filter products by categories, dynamically changing the products displayed.

- **Search Bar**: Allows users to search for products by name or tags.

1.4 Related Products

Dynamically fetched related products that are shown on the product detail page, helping users explore similar items. (Displaying Featured Products)

1.6 Cart Page

The cart allows users to add products, view item quantity, and total price. Users can also remove items or adjust quantities.
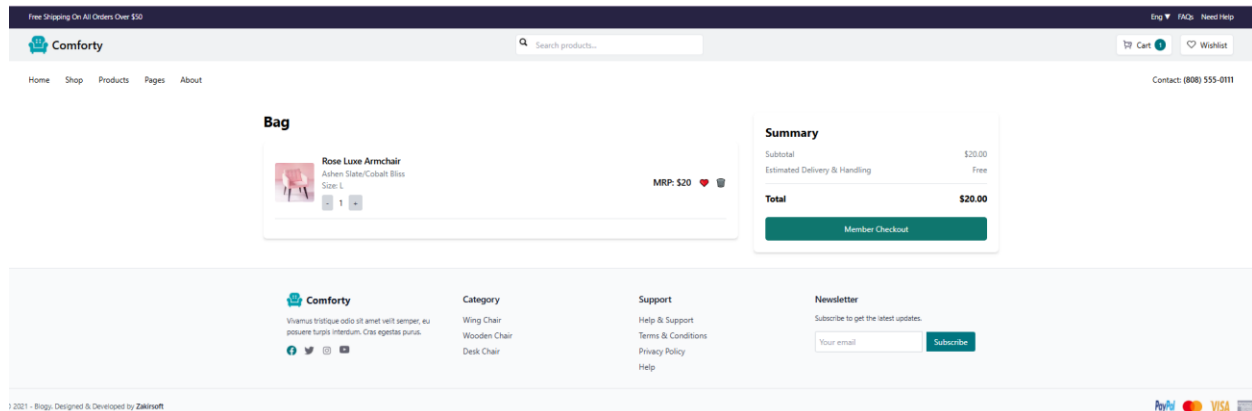
1.7 FAQ and Help Center

An FAQ section that answers common questions, along with a help center for customer support.

- Show the FAQ page with questions and answers.

- Display the contact form or help center features.
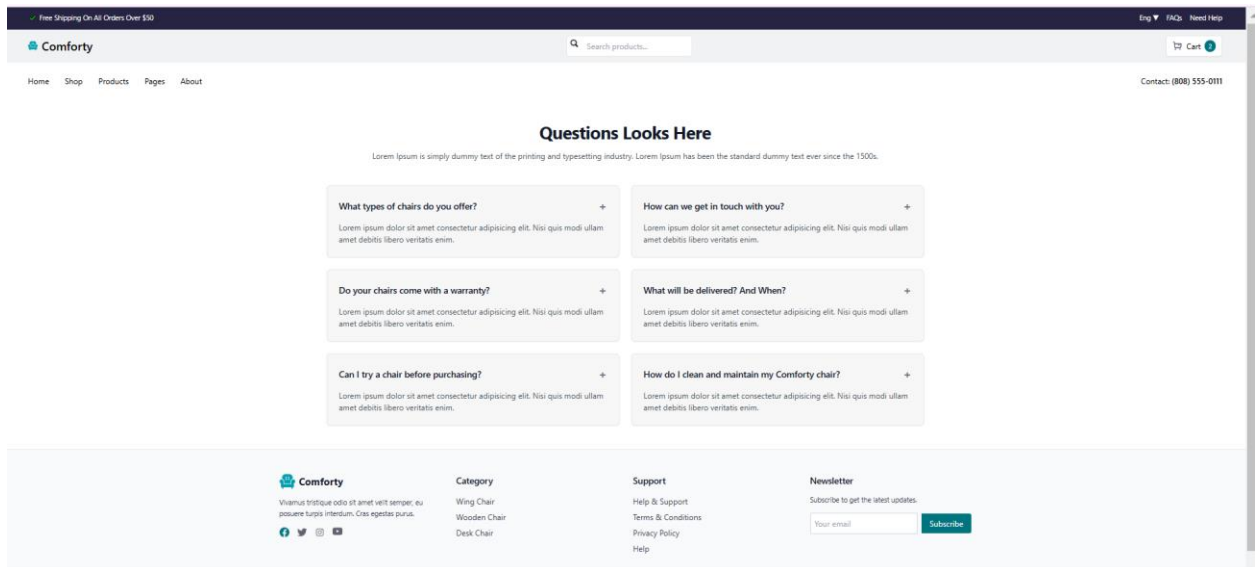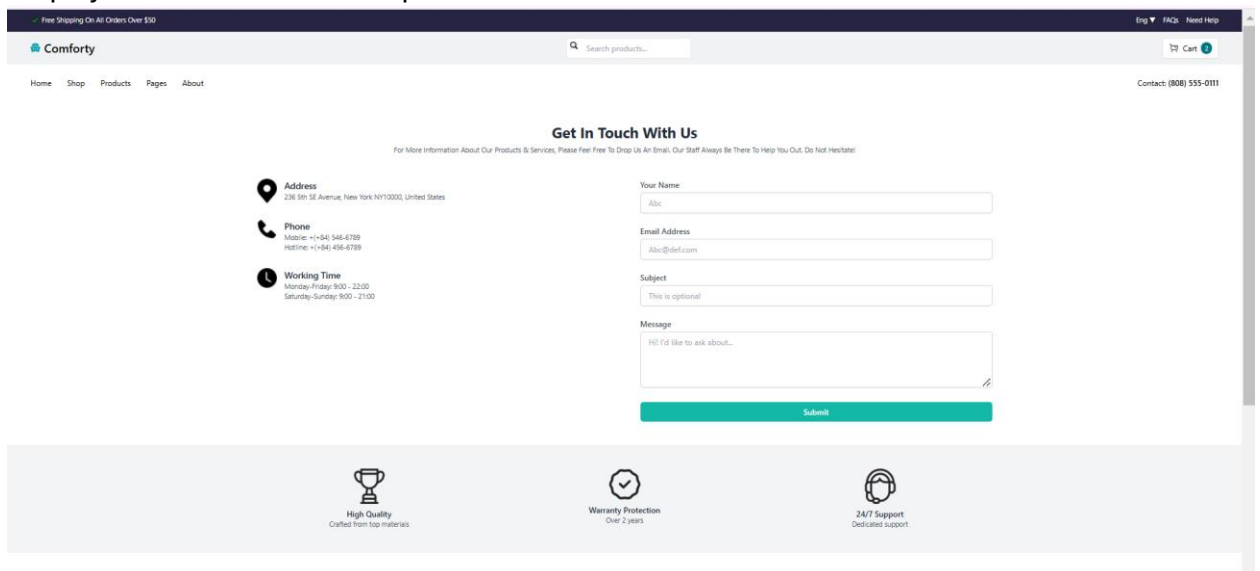

1.8 Wishlist Page

Comforty

Search products...

Cart 1   ♡ Wishlist

Home   Shop   Products   Pages   About

Contact: (808) 555-0111

Wishlist

Rose Luxe Armchair
Price: $20

Add to Cart 🗑

Comforty

Vivamus tristique odio sit amet velit semper, eu
posuere turpis interdum. Cras egestas purus.

Category

Wing Chair

Wooden Chair

Desk Chair

Support

Help & Support

Terms & Conditions

Privacy Policy

Help

Newsletter

Subscribe to get the latest updates.
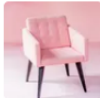
Your email   Subscribe

1.9 Checkout Page

Checkout

Cart Summary

Rose Luxe Armchair
Quantity: 1
Price: $20
Total: $20.00

Total Price   **$20.00**

Billing Details

Full Name	Email Address
Phone Number	Company (optional)
Street Address	City
State	ZIP Code
Country	Order Notes (optional)

Payment Method

Cash on Delivery

Place Order

2.0 AboutUs Page

About Us - Comforty

At Comforty, we believe that the right chair can transform your space and elevate your comfort. Specializing in ergonomic design, premium materials, and modern aesthetics, we craft chairs that seamlessly blend style with functionality.

View collection

What Makes Our Brand Different

Next day as standard

Order before 3pm and get your order the next day as standard.

Made by true artisans

Handmade crafted goods made with real passion and craftsmanship.

Unbeatable prices

For our materials and quality, you won't find better prices anywhere.

Recycled packaging

We use 100% recycled to ensure our footprint is more manageable.

Our Popular Products

$20.00

$20.00

$20.00

2. Code Deliverables

Here are code snippets for some of the key components:

2.1 ProductCard Component

The ProductCard component displays an individual product's image, title, and price.

```tsx
"use client";

import React from "react";
import Image from "next/image";
import Link from "next/link";

interface ProductCardProps {
  id: string;
  image: string;
  title: string;
  price: string;
  oldPrice?: string;
  isNew?: boolean;
  isOnSale?: boolean;
  alt: string; // Add alt prop here
}

const ProductCard: React.FC<ProductCardProps> = ({
  id,
  image,
  title,
  price,
  oldPrice,
  isNew,
  isOnSale,
  alt, // Destructure alt here
}) => {
  return (
    <div className="relative group rounded-lg overflow-hidden border border-gray-200 shadow-md hover:shadow-lg transition">
      {/* Label for New or Sale */}
      {(isNew || isOnSale) && (
        <div
          className={`absolute top-2 left-2 text-white text-xs px-2 py-1 rounded-md font-medium ${
            isNew ? "bg-[#1DBF73]" : "bg-[#F97316]"
          }`}
        >
          {isNew ? "New" : "Sale"}
        </div>
      )}

      {/* Image Section */}
      <Link href={`/products/${id}`}>
        <Image
          src={image}
          alt={alt} // Use the alt prop here
          width={300}
          height={300}
          className="object-cover w-full h-[250px] sm:h-[300px] transition-transform group-hover:scale-105"
        />
      </Link>

      {/* Product Info Section */}
      <div className="flex justify-between items-center px-4 py-3 bg-white">
        <div>
          <p className="text-sm text-gray-700 group-hover:text-[#029FAE] transition cursor-pointer">
            {title}
          </p>
          <p className="text-lg font-bold mt-1 group-hover:text-[#029FAE] transition cursor-pointer">
            {price}
            {oldPrice && (
              <span className="line-through text-gray-400 ml-2 text-sm">
                {oldPrice}
              </span>
            )}
          </p>
        </div>
        <Link href="/cart">
          <div className="hover:bg-[#029FAE] bg-gray-200 p-2 rounded-full transition">
            <Image
              src="/cart.png"
              alt="Cart Icon"
              width={20}
              height={20}
              className="text-white"
            />
          </div>
        </Link>
      </div>
    </div>
  );
};

export default ProductCard;
```

2.2 ProductList Component

The ProductList component renders a list of products dynamically fetched from an API.

```
1   "use client";
2
3   import React, { useEffect, useState } from "react";
4   import ProductCard from "../../../components/productcard";
5   import NewsletterSubscription from "../../../components/newslettersubscription";
6   import { client } from "@/sanity/lib/client";
7
8   interface Product {
9     _id: string;
10    name: string;
11    price: number;
12    oldPrice?: number;
13    isNew?: boolean;
14    isOnSale?: boolean;
15    image_url: string;
16  }
17
18  const ProductsPage = () => {
19    const [products, setProducts] = useState<Product[]>([]);
20    const [loading, setLoading] = useState<boolean>(true); // Loading state
21
22    useEffect(() => {
23      const fetchProducts = async () => {
24        const query = `*[_type == "products"]{
25          _id,
26          name,
27          price,
28          oldPrice,
29          isNew,
30          isOnSale,
31          "image_url": image.asset->url
32        }`;
33        const fetchedProducts = await client.fetch<Product[]>(query);
34        setProducts(fetchedProducts);
35        setLoading(false); // Set loading to false after fetching
36      };
37
38      fetchProducts();
39    }, []);
40
41    return (
42      <div className="container mx-auto px-4 py-8">
43        <h2 className="text-2xl font-bold mb-6 text-center">Our Products</h2>
44        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-4 sm:gap-6">
45          {loading ? (
46            // Skeleton loader (use Tailwind CSS for skeleton effect)
47            Array.from({ length: 4 }).map((_, index) => (
48              <div
49                key={index}
50                className="p-4 bg-gray-300 animate-pulse rounded-lg"
51              >
52                <div className="h-40 bg-gray-200 mb-4"></div>
53                <div className="h-4 bg-gray-200 mb-2"></div>
54                <div className="h-4 bg-gray-200"></div>
55              </div>
56            ))
57          ) : products.length > 0 ? (
58            products.map((product) => (
59              <ProductCard
60                key={product._id}
61                id={product._id}
62                image={product.image_url || "/placeholder.png"} // Default to placeholder
63                alt={product.name} // Add alt attribute for accessibility
64                title={product.name}
65                price={`$${product.price.toFixed(2)}`}
66                oldPrice={
67                  product.oldPrice ? `$${product.oldPrice.toFixed(2)}` : undefined
68                }
69                isNew={product.isNew}
70                isOnSale={product.isOnSale}
71              />
72            ))
73          ) : (
74            <p className="text-center col-span-full">No products available.</p>
75          )}
76        </div>
77        {/* Add Newsletter Subscription Section */}
78        <NewsletterSubscription />
79      </div>
80    );
81  };
82
83  export default ProductsPage;
84
```

2.3 SearchBar Component

The SearchBar component allows users to search products by name.

```
1   // Fetch products from Sanity
2     useEffect(() => {
3       const fetchProducts = async () => {
4         const query = `*[_type == "products"]{ _id, title, tags }`;
5         const result = await client.fetch<Product[]>(query);
6         setProducts(result);
7       };
8
9       fetchProducts();
10    }, []);
11
12    // Search filter function
13    const handleSearch = (query: string) => {
14      setSearchQuery(query);
15      const lowerCaseQuery = query.toLowerCase();
16      const filtered = products.filter(
17        (product) =>
18          product.title.toLowerCase().includes(lowerCaseQuery) ||
19          product.tags.some((tag) => tag.toLowerCase().includes(lowerCaseQuery))
20      );
21      setFilteredProducts(filtered);
22    };
```

2.4 API Integration for Product Data

Fetching data from an API to display products dynamically.

```
1   useEffect(() => {
2       const fetchProducts = async () => {
3         const query = `*[_type == "products"]{
4           _id,
5           name,
6           price,
7           oldPrice,
8           isNew,
9           isOnSale,
10          "image_url": image.asset->url
11        }`;
12        const fetchedProducts = await client.fetch<Product[]>(query);
13        setProducts(fetchedProducts);
14        setLoading(false); // Set loading to false after fetching
15      };
16
17      fetchProducts();
18    }, []);
```

2.5 Dynamic Routing for Product Detail Page

Using Next.js's useRouter to dynamically load product details based on the product ID from the URL.

```
1   "use client";
2
3   import React, { useEffect, useState } from "react";
4   import ProductCard from "../../../components/productcard";
5   import NewsletterSubscription from "../../../components/newslettersubscription";
6   import { client } from "@/sanity/lib/client";
7
8   interface Product {
9     _id: string;
10    name: string;
11    price: number;
12    oldPrice?: number;
13    isNew?: boolean;
14    isOnSale?: boolean;
15    image_url: string;
16  }
17
18  const ProductsPage = () => {
19    const [products, setProducts] = useState<Product[]>([]);
20    const [loading, setLoading] = useState<boolean>(true); // Loading state
21
22    useEffect(() => {
23      const fetchProducts = async () => {
24        const query = `*[_type == "products"]{
25          _id,
26          name,
27          price,
28          oldPrice,
29          isNew,
30          isOnSale,
31          "image_url": image.asset->url
32        }`;
33        const fetchedProducts = await client.fetch<Product[]>(query);
34        setProducts(fetchedProducts);
35        setLoading(false); // Set loading to false after fetching
36      };
37
38      fetchProducts();
39    }, []);
40
41    return (
42      <div className="container mx-auto px-4 py-8">
43        <h2 className="text-2xl font-bold mb-6 text-center">Our Products</h2>
44        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-4 sm:gap-6">
45          {loading ? (
46            // Skeleton loader (use Tailwind CSS for skeleton effect)
47            Array.from({ length: 4 }).map((_, index) => (
48              <div
49                key={index}
50                className="p-4 bg-gray-300 animate-pulse rounded-lg"
51              >
52                <div className="h-40 bg-gray-200 mb-4"></div>
53                <div className="h-4 bg-gray-200 mb-2"></div>
54                <div className="h-4 bg-gray-200"></div>
55              </div>
56            ))
57          ) : products.length > 0 ? (
58            products.map((product) => (
59              <ProductCard
60                key={product._id}
61                id={product._id}
62                image={product.image_url || "/placeholder.png"} // Default to placeholder
63                alt={product.name} // Add alt attribute for accessibility
64                title={product.name}
65                price={`$${product.price.toFixed(2)}`}
66                oldPrice={
67                  product.oldPrice ? `$${product.oldPrice.toFixed(2)}` : undefined
68                }
69                isNew={product.isNew}
70                isOnSale={product.isOnSale}
71              />
72            ))
73          ) : (
74            <p className="text-center col-span-full">No products available.</p>
75          )}
76        </div>
77        {/* Add Newsletter Subscription Section */}
78        <NewsletterSubscription />
79      </div>
80    );
81  };
82
83  export default ProductsPage;
84
```

3. Documentation

3.1 Challenges Faced

- **Dynamic Routing**: Ensuring the correct data loads for product details based on the dynamic URL parameters was initially tricky. We solved it by using Next.js's `useRouter` hook.
- **Handling Large Product Data**: Displaying large datasets in a user-friendly manner required implementing pagination and optimizing the API response.
- **State Management**: Managing the state for the wishlist and cart was challenging, so we used React Context and `localStorage` for persistence.

5. Conclusion

The dynamic frontend components have been successfully implemented for the Comforty e-commerce website. The features include product listing, search functionality, dynamic product detail pages, related products, a cart, a wishlist, and social media sharing.

This documentation ensures all components and their functionalities are well understood and easy to navigate through, both in the codebase and in the user interface.

You can use this structure to organize your documentation and share it in PDF or Markdown format. Let me know if you'd like further details or assistance!