Comforty E-Commerce Website Documentation

Day 4: Building Dynamic Frontend Components

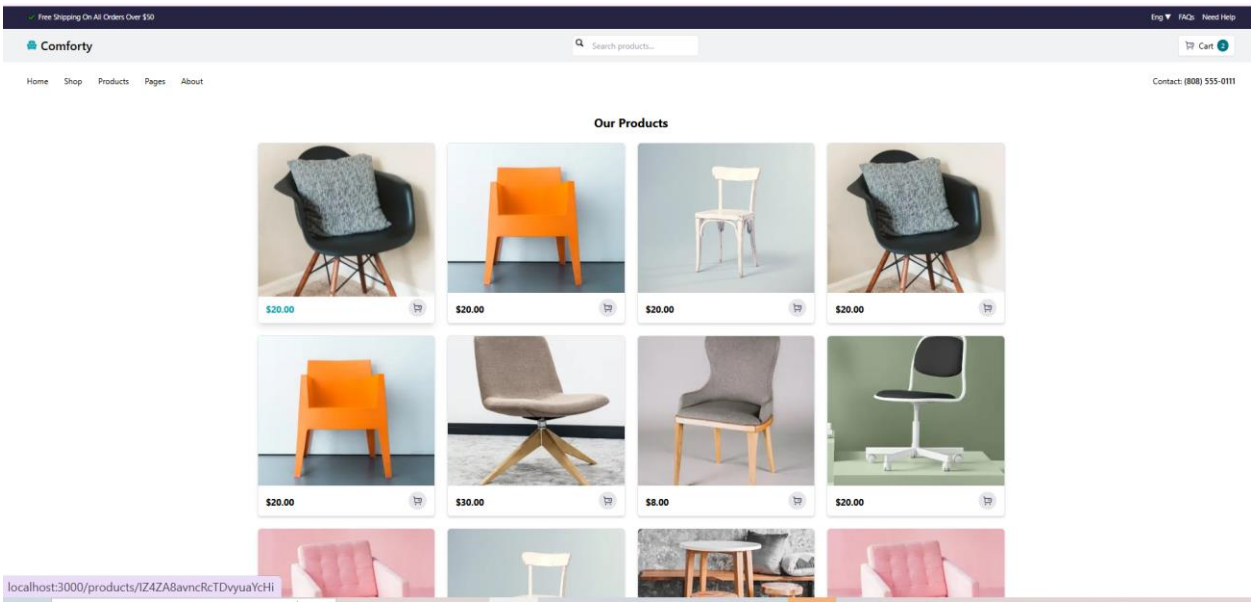
Muskaan Fayyaz

Roll no: 00216348

1. Functional Deliverables

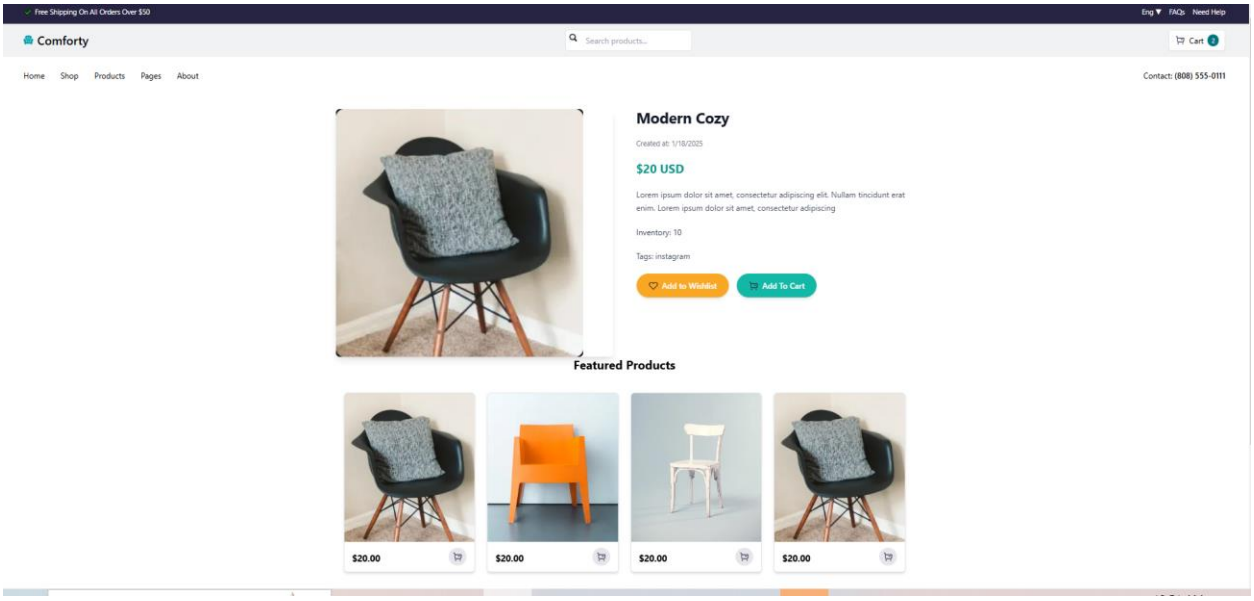
1.1 Product Listing Page

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.



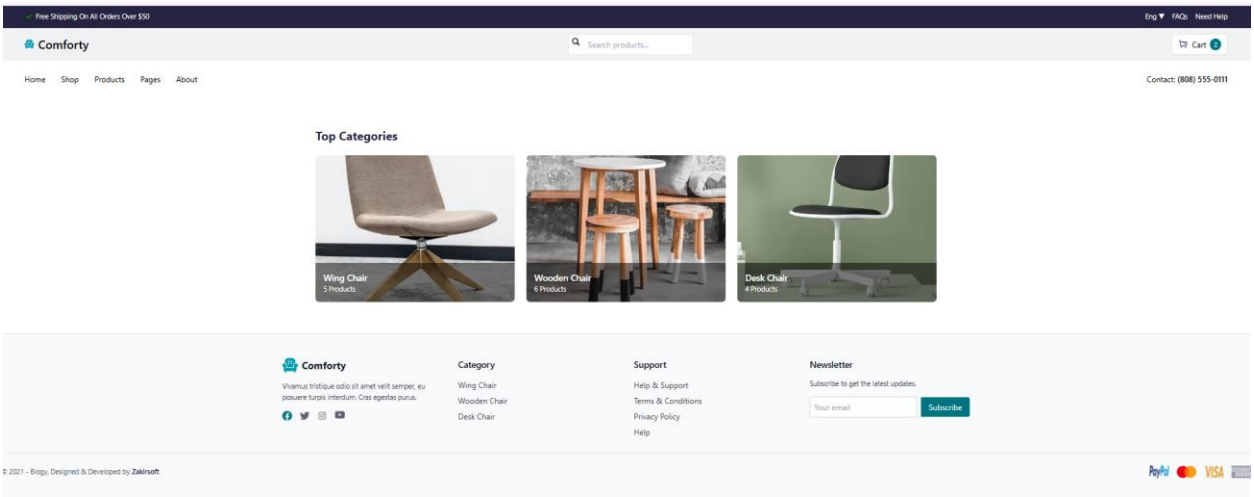
1.2 Product Detail Page

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.

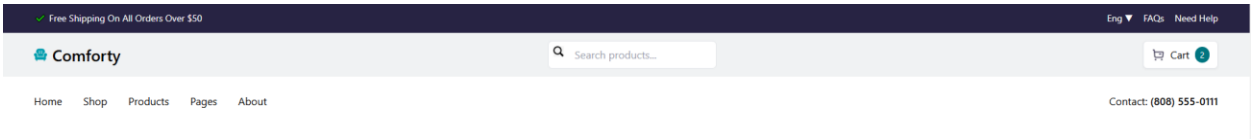


1.3 Category Filters and Search Bar

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

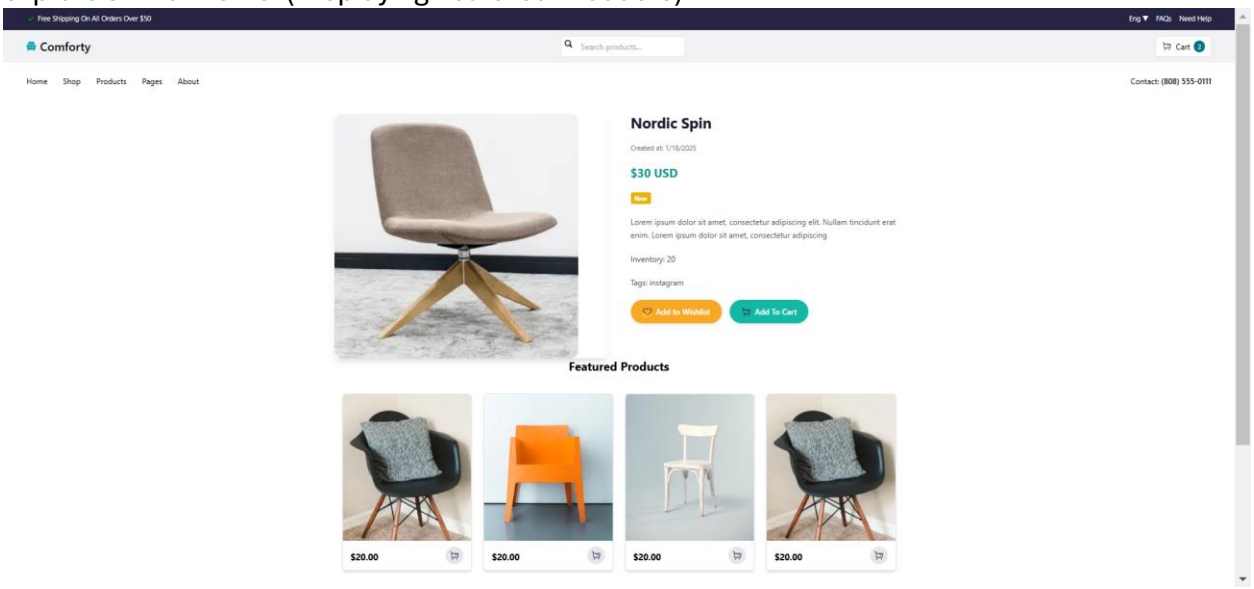


- **Search Bar**: Allows users to search for products by name or tags.



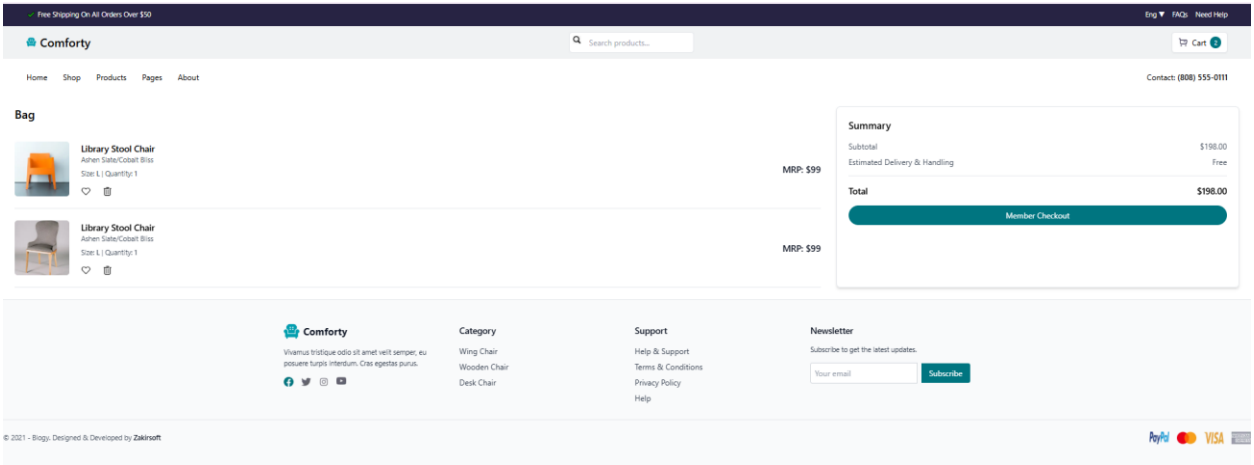
1.4 Related Products

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



1.6 Cart Component

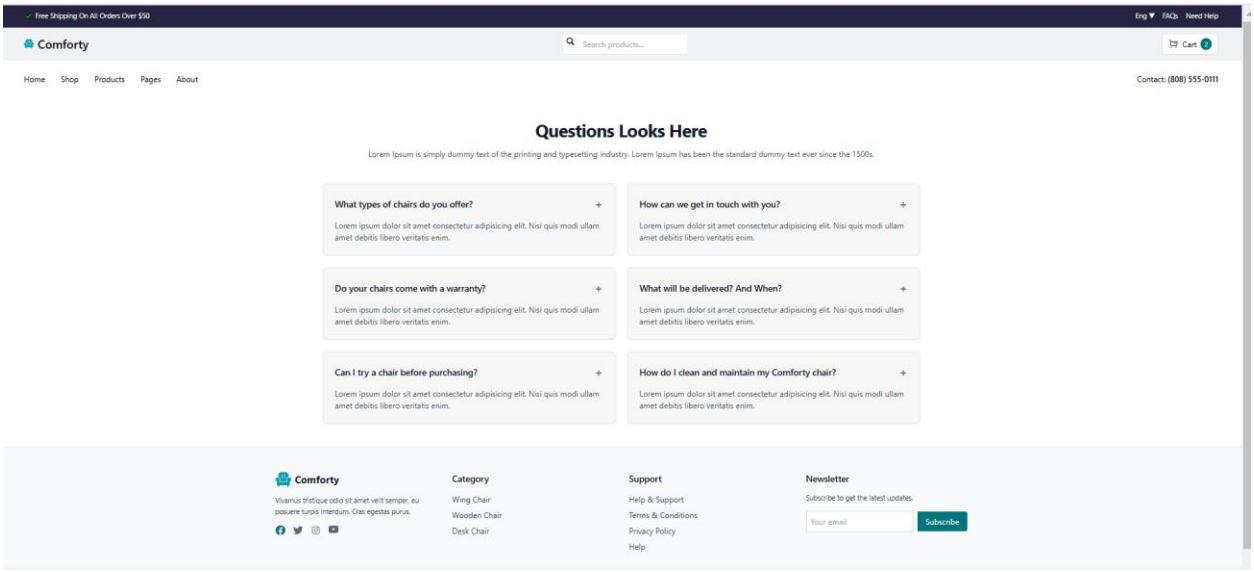
The cart allows users to add products, view item quantity, and total price. Users can also remove items or adjust quantities.



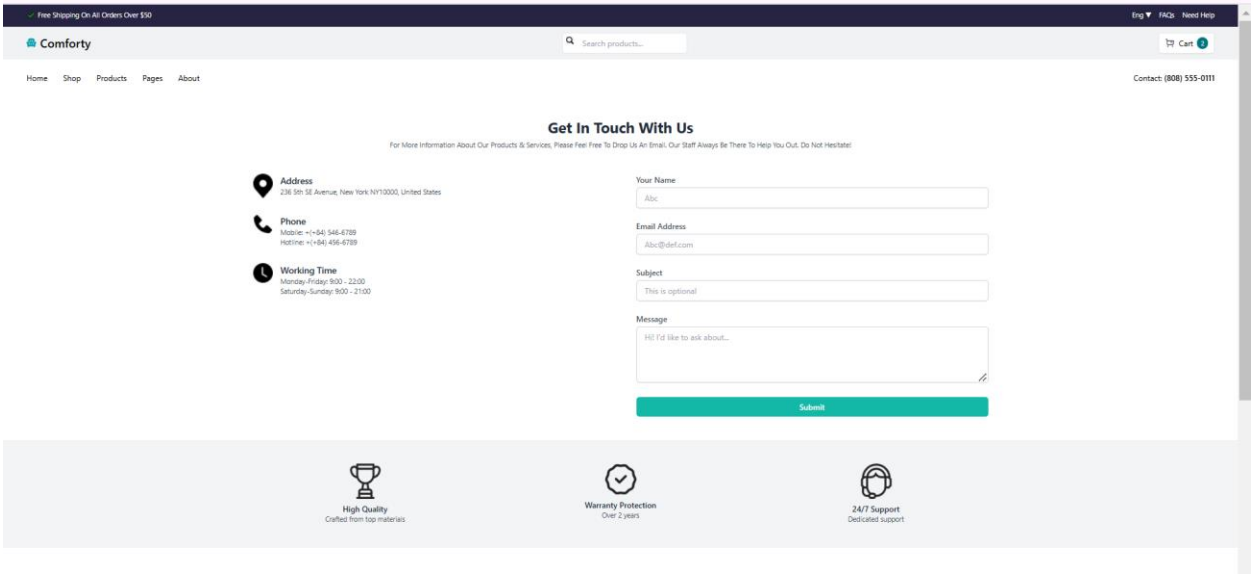
1.7 FAQ and Help Center

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.



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.

```tsx
"use client";

import React, { useEffect, useState } from "react";
import ProductCard from "../../../components/productcard";
import NewsletterSubscription from "../../../components/newslettersubscription";
import { client } from "@/sanity/lib/client";

interface Product {
  _id: string;
  name: string;
  price: number;
  oldPrice?: number;
  isNew?: boolean;
  isOnSale?: boolean;
  image_url: string;
}

const ProductsPage = () => {
  const [products, setProducts] = useState<Product[]>([]);
  const [loading, setLoading] = useState<boolean>(true); // Loading state

  useEffect(() => {
    const fetchProducts = async () => {
      const query = `*[_type == "products"]{
        _id,
        name,
        price,
        oldPrice,
        isNew,
        isOnSale,
        "image_url": image.asset->url
      }`;
      const fetchedProducts = await client.fetch<Product[]>(query);
      setProducts(fetchedProducts);
      setLoading(false); // Set loading to false after fetching
    };

    fetchProducts();
  }, []);

  return (
    <div className="container mx-auto px-4 py-8">
      <h2 className="text-2xl font-bold mb-6 text-center">Our Products</h2>
      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-4 sm:gap-6">
        {loading ? (
          // Skeleton loader (use Tailwind CSS for skeleton effect)
          Array.from({ length: 4 }).map((_, index) => (
            <div
              key={index}
              className="p-4 bg-gray-300 animate-pulse rounded-lg"
            >
              <div className="h-40 bg-gray-200 mb-4"></div>
              <div className="h-4 bg-gray-200 mb-2"></div>
              <div className="h-4 bg-gray-200"></div>
            </div>
          ))
        ) : products.length > 0 ? (
          products.map((product) => (
            <ProductCard
              key={product._id}
              id={product._id}
              image={product.image_url || "/placeholder.png"} // Default to placeholder
              alt={product.name} // Add alt attribute for accessibility
              title={product.name}
              price={`$${product.price.toFixed(2)}`}
              oldPrice={
                product.oldPrice ? `$${product.oldPrice.toFixed(2)}` : undefined
              }
              isNew={product.isNew}
              isOnSale={product.isOnSale}
            />
          ))
        ) : (
          <p className="text-center col-span-full">No products available.</p>
        )}
      </div>
      {/* Add Newsletter Subscription Section */}
      <NewsletterSubscription />
    </div>
  );
};

export default ProductsPage;
```

2.3 SearchBar Component

The SearchBar component allows users to search products by name.

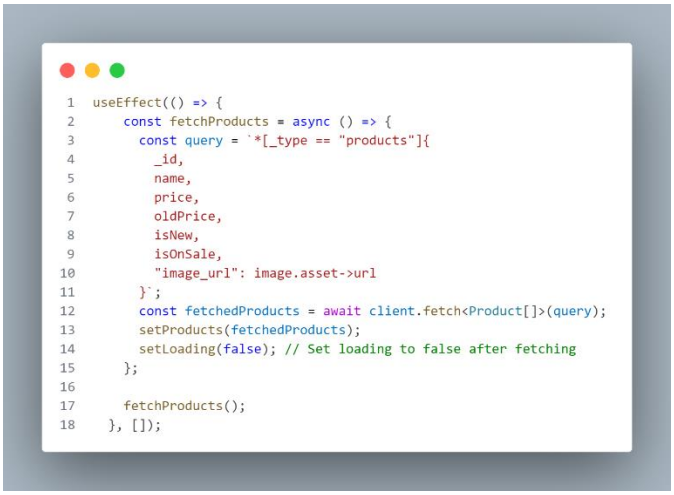
```tsx
// Fetch products from Sanity
  useEffect(() => {
    const fetchProducts = async () => {
      const query = `*[_type == "products"]{ _id, title, tags }`;
      const result = await client.fetch<Product[]>(query);
      setProducts(result);
    };

    fetchProducts();
  }, []);

  // Search filter function
  const handleSearch = (query: string) => {
    setSearchQuery(query);
    const lowerCaseQuery = query.toLowerCase();
    const filtered = products.filter(
      (product) =>
        product.title.toLowerCase().includes(lowerCaseQuery) ||
        product.tags.some((tag) => tag.toLowerCase().includes(lowerCaseQuery))
    );
    setFilteredProducts(filtered);
  };
```

2.4 API Integration for Product Data

Fetching data from an API to display products dynamically.



```tsx
useEffect(() => {
    const fetchProducts = async () => {
      const query = `*[_type == "products"]{
        _id,
        name,
        price,
        oldPrice,
        isNew,
        isOnSale,
        "image_url": image.asset->url
      }`;
      const fetchedProducts = await client.fetch<Product[]>(query);
      setProducts(fetchedProducts);
      setLoading(false); // Set loading to false after fetching
    };

    fetchProducts();
  }, []);
```

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.

```tsx
"use client";

import React, { useEffect, useState } from "react";
import ProductCard from "../../../components/productcard";
import NewsletterSubscription from "../../../components/newslettersubscription";
import { client } from "@/sanity/lib/client";

interface Product {
  _id: string;
  name: string;
  price: number;
  oldPrice?: number;
  isNew?: boolean;
  isOnSale?: boolean;
  image_url: string;
}

const ProductsPage = () => {
  const [products, setProducts] = useState<Product[]>([]);
  const [loading, setLoading] = useState<boolean>(true); // Loading state

  useEffect(() => {
    const fetchProducts = async () => {
      const query = `*[_type == "products"]{
        _id,
        name,
        price,
        oldPrice,
        isNew,
        isOnSale,
        "image_url": image.asset->url
      }`;
      const fetchedProducts = await client.fetch<Product[]>(query);
      setProducts(fetchedProducts);
      setLoading(false); // Set loading to false after fetching
    };

    fetchProducts();
  }, []);

  return (
    <div className="container mx-auto px-4 py-8">
      <h2 className="text-2xl font-bold mb-6 text-center">Our Products</h2>
      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-4 sm:gap-6">
        {loading ? (
          // Skeleton loader (use Tailwind CSS for skeleton effect)
          Array.from({ length: 4 }).map((_, index) => (
            <div
              key={index}
              className="p-4 bg-gray-300 animate-pulse rounded-lg"
            >
              <div className="h-40 bg-gray-200 mb-4"></div>
              <div className="h-4 bg-gray-200 mb-2"></div>
              <div className="h-4 bg-gray-200"></div>
            </div>
          ))
        ) : products.length > 0 ? (
          products.map((product) => (
            <ProductCard
              key={product._id}
              id={product._id}
              image={product.image_url || "/placeholder.png"} // Default to placeholder
              alt={product.name} // Add alt attribute for accessibility
              title={product.name}
              price={`$${product.price.toFixed(2)}`}
              oldPrice={
                product.oldPrice ? `$${product.oldPrice.toFixed(2)}` : undefined
              }
              isNew={product.isNew}
              isOnSale={product.isOnSale}
            />
          ))
        ) : (
          <p className="text-center col-span-full">No products available.</p>
        )}
      </div>
      {/* Add Newsletter Subscription Section */}
      <NewsletterSubscription />
    </div>
  );
};

export default ProductsPage;

```

---

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!