

Day 4 - Building Dynamic Frontend Components for Your Marketplace

OVER VIEW

On Day 4 is all about how to design and develop dynamic frontend components for displaying marketplace data fetched from Sanity CMS or APIs. The focus will be on creating modular, reusable components that ensure scalability and responsiveness. Practical techniques for integrating real-world data into web applications will be emphasized. This day will equip students with essential skills for building dynamic, data-driven user interfaces.

Built in the Following Key Components;

- 1- Product Listing Component:
- 2- Product Detail Component:
- 3- Cart Component:
- 4- Wishlist Component:
- 5- Footer and Header Components
- 6- Category Component:
- 7- Checkout Flow Component:

Product Listing Component:

The Product Listing Component integrates data from a provided API into Sanity, dynamically fetching product details for the production page. It features a reusable ProductInCard component for individual product display and an AllProductsPage to show the full catalog. This setup ensures a smooth, responsive, and up-to-date product listing experience.

```
src
├── app
│   ├── components
│   │   ├── editorspick
│   │   ├── Footer
│   │   ├── Furniture
│   │   │   ├── page.tsx
│   │   │   ├── Futured
│   │   │   ├── Hero
│   │   ├── Navbar
│   │   │   ├── page.tsx
│   │   ├── ProductCard
│   │   │   ├── page.tsx
│   │   │   ├── products-Card
│   │   │   ├── SearchBar
│   │   ├── Topheader
│   │   ├── Universe
│   │   ├── Wishlist
│   │   ├── WishlistPopup
│   │   ├── fonts
│   │   └── product
│   │       ├── [id]
│   │       ├── page.tsx
│   │       └── page.tsx
│   └── studio
├── ...
└── ...
```

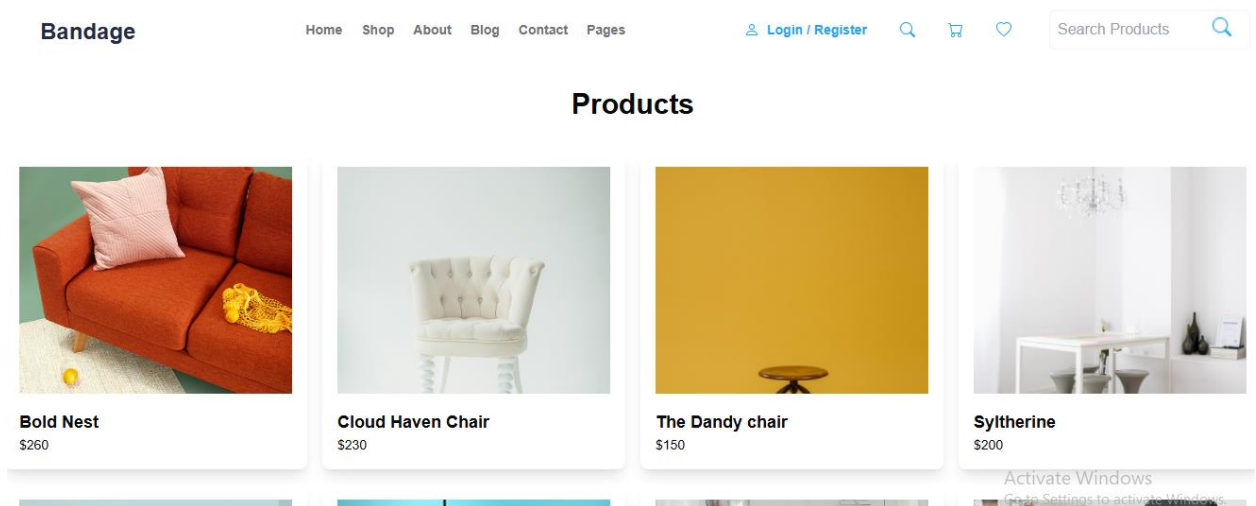
```
import Navbar from "../components/Navbar/page";
import Footer from "../components/Footer/page";
import ProductCard from "../components/ProductCard/page";

type Product = {
  _id: string;
  title: string;
  description: string;
  price: number;
  productImage: {
    asset: {
      url: string;
    };
  };
  tags: string[]; // Available sizes/colors
  stockStatus: "In Stock" | "Out of Stock";
};

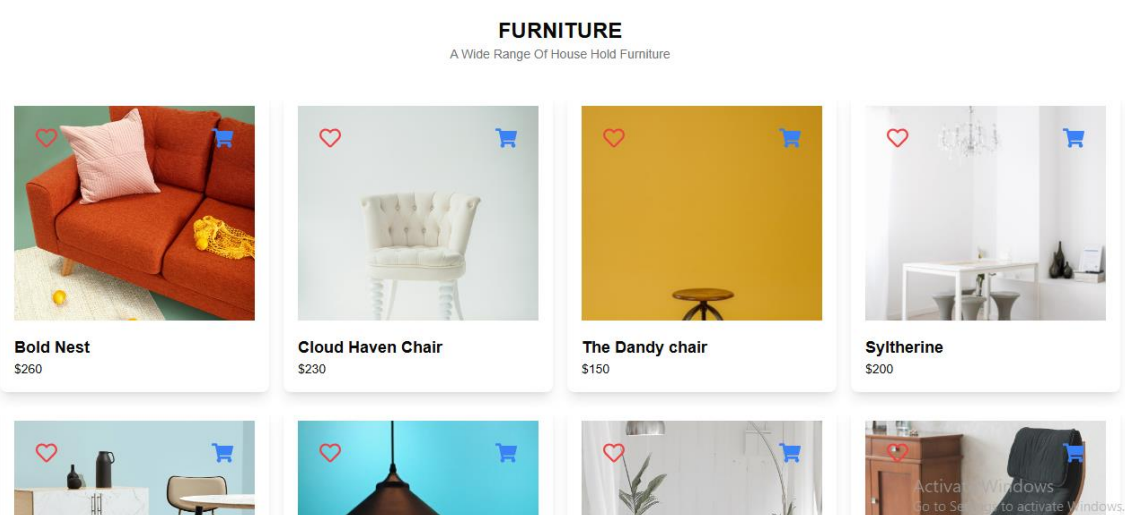
export default async function Home() {
  const products: Product[] = await sanityFetch({ query: allproducts });

  return (
    <div>
      <Topheader />
      <Navbar />
      <h1 className="text-center text-3xl font-bold my-8">Products</h1>
      <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4">
        {products.map((product) => (
          <ProductCard key={product._id} product={product} />
        ))}
      </div>
    </div>
  );
}
```

VS code of all products fetched in to the shop page and components

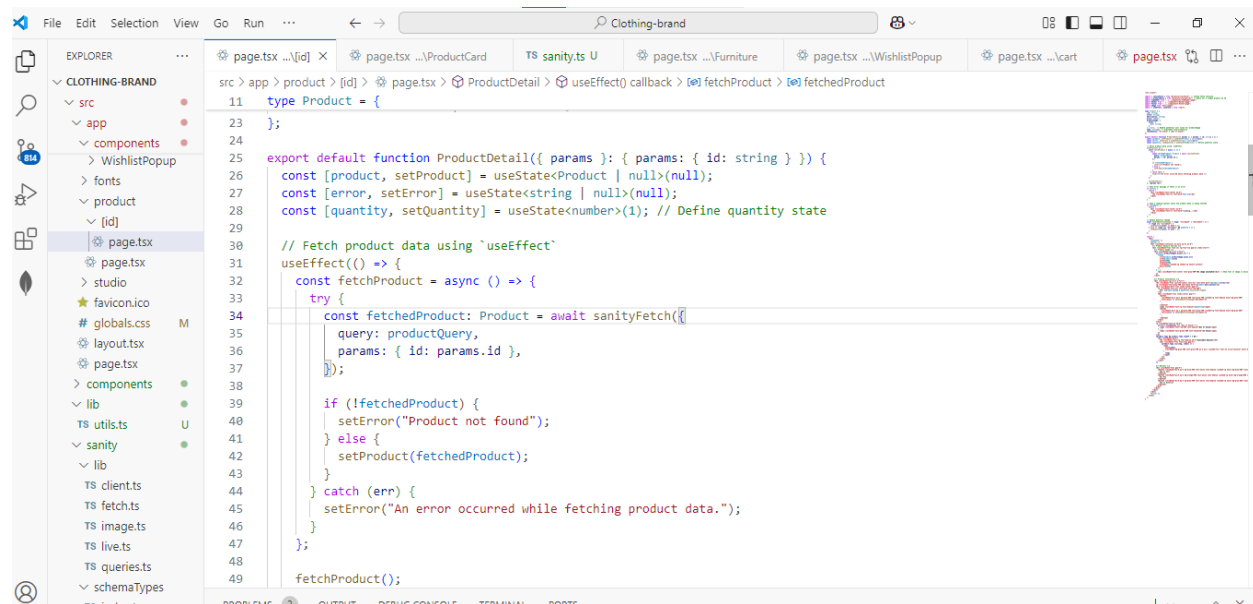


Selected Components render on main page

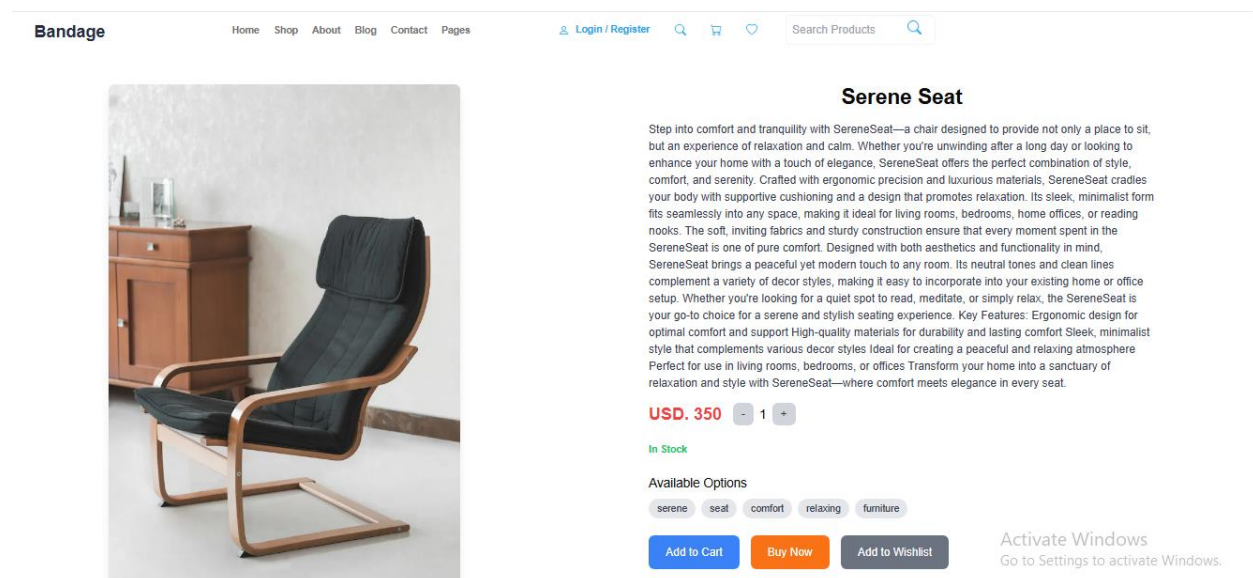


Product Detail Component:

The Product Detail Component is built with dynamic routing, allowing users to click on a product image from the homepage or the all-products page and be redirected to a detailed product view. This component displays key product information, including the product name, description, price, stock availability, and interactive buttons for adding to the cart, making a purchase, or adding to the wishlist. The dynamic routing ensures a seamless transition and personalized experience for users. The component is designed to provide a comprehensive and user-friendly interface for product interaction. This approach enhances the overall shopping experience by delivering all essential details in one place.



```
src > app > product > [id] > page.tsx > ProductDetail > useEffect() callback > fetchProduct > fetchedProduct
11 type Product = {
23   };
24
25 export default function ProductDetail({ params }: { params: { id: string } }) {
26   const [product, setProduct] = useState<Product | null>(null);
27   const [error, setError] = useState<string | null>(null);
28   const [quantity, setQuantity] = useState<number>(1); // Define quantity state
29
30   // Fetch product data using 'useEffect'
31   useEffect(() => {
32     const fetchProduct = async () => {
33       try {
34         const fetchedProduct: Product = await sanityFetch({
35           query: productQuery,
36           params: { id: params.id },
37         });
38
39         if (!fetchedProduct) {
40           setError("Product not found");
41         } else {
42           setProduct(fetchedProduct);
43         }
44       } catch (err) {
45         setError("An error occurred while fetching product data.");
46       }
47     };
48     fetchProduct();
49   });
```



Cart Component:

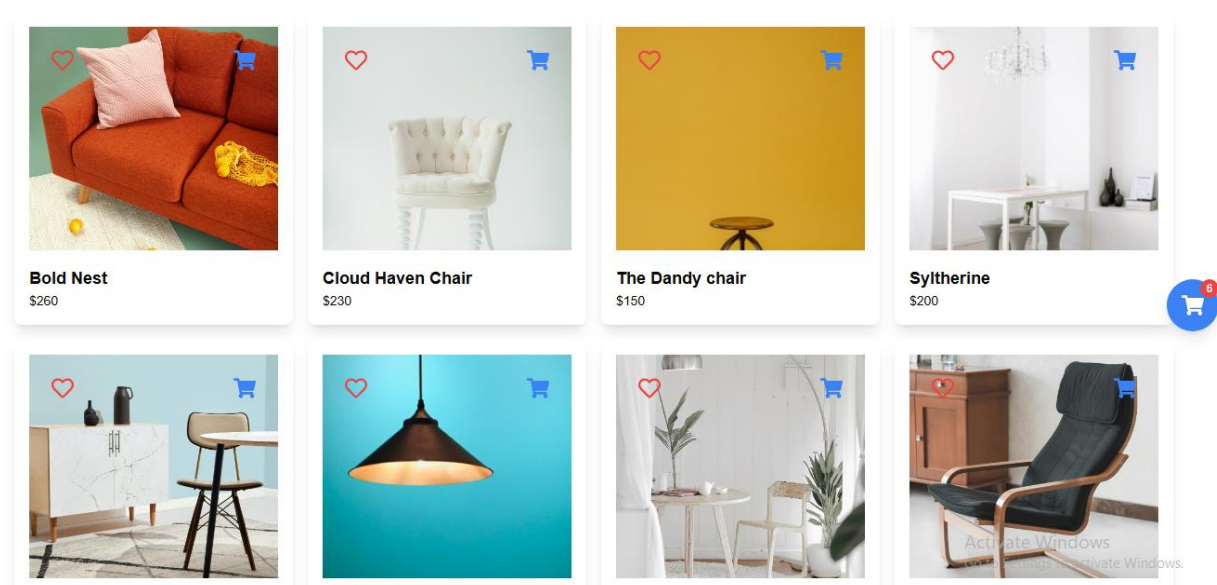
The Cart Component allows users to easily add, view, and manage products in their cart. The Cart icon is displayed dynamically, updating with the number of items added. When an item is added, a pop-up shows the quantity, providing instant feedback. Clicking the Cart icon opens a detailed cart view, where users can adjust quantities, remove items, and see total prices. The component integrates seamlessly with product pages and supports a smooth shopping experience. It uses global state management through the CartContext for consistent cart data across the app. The cart can be toggled open and closed, allowing quick access.

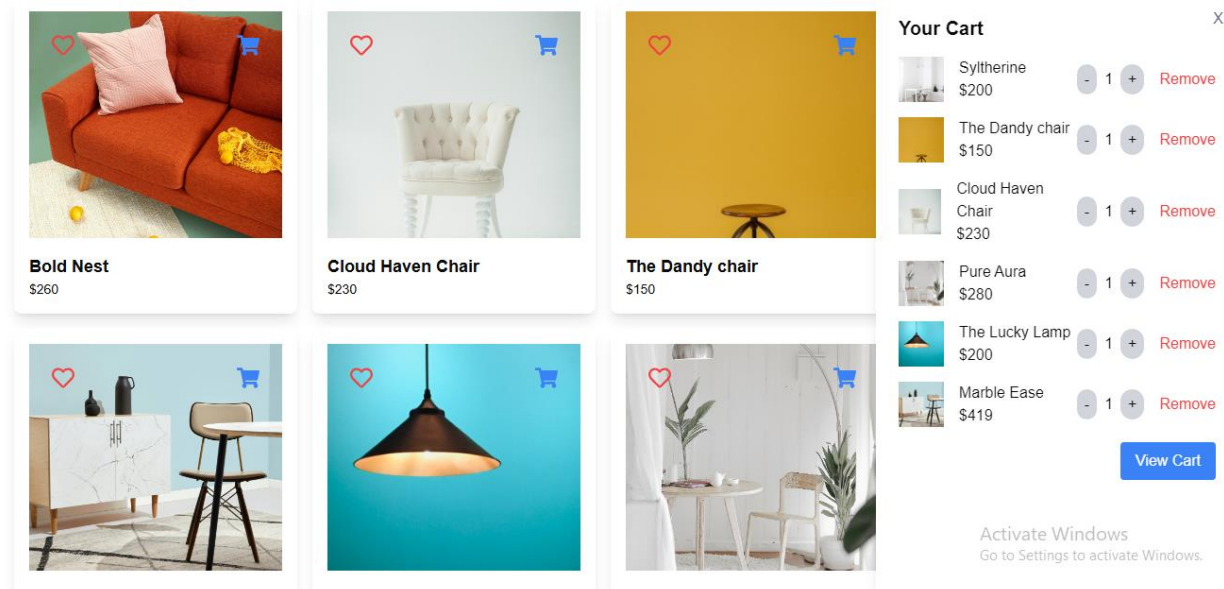


```
src > app > components > Furniture > page.tsx > Home > useEffect() callback > handleScroll

29
30
31 export default function Home() {
32   const [wishlist, setWishlist] = useState<Product[]>([]);
33   const [cart, setCart] = useState<CartItem[]>([]);
34   const [products, setProducts] = useState<Product[]>([]);
35   const [isWishlistOpen, setIsWishlistOpen] = useState(false);
36   const [isCartOpen, setIsCartOpen] = useState(false);
37   const [scrollY, setScrollY] = useState(0); // Track scroll position
38
39   useEffect(() => {
40     const fetchProducts = async () => {
41       const products: Product[] = await sanityFetch({ query: furniture });
42       setProducts(products);
43     };
44     fetchProducts();
45
46     // Listen to the scroll event
47     const handleScroll = () => {
48       setScrollY(window.scrollY); // Update scrollY on scroll
49     };
50
51     window.addEventListener("scroll", handleScroll); // Add scroll event listener
52     return () => {
53       window.removeEventListener("scroll", handleScroll); // Cleanup on component unmount
54     };
55   }, []);
56
57   const addToWishlist = (product: Product) => {
58     setWishlist((prevWishlist) => [...prevWishlist, product]);
59   };
60
```

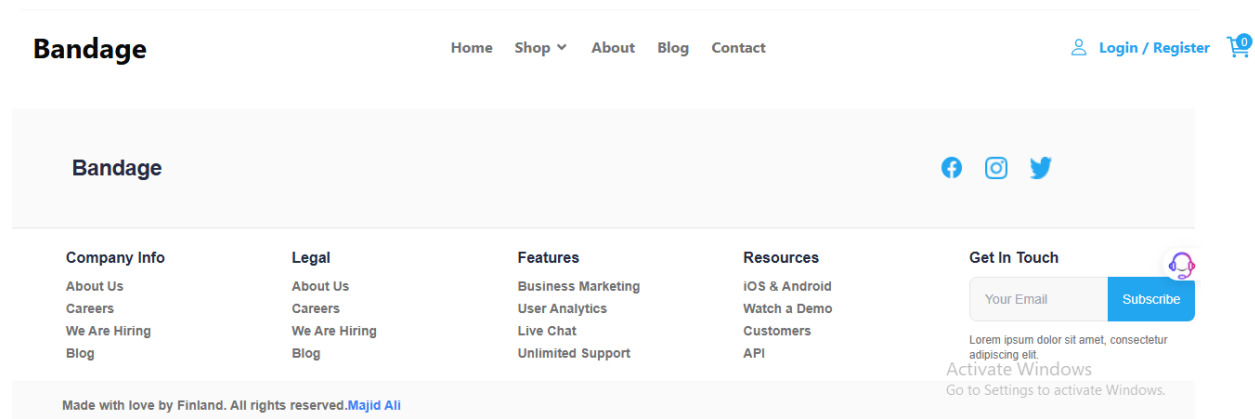
Activate Windows
Go to Settings to activate Windows.

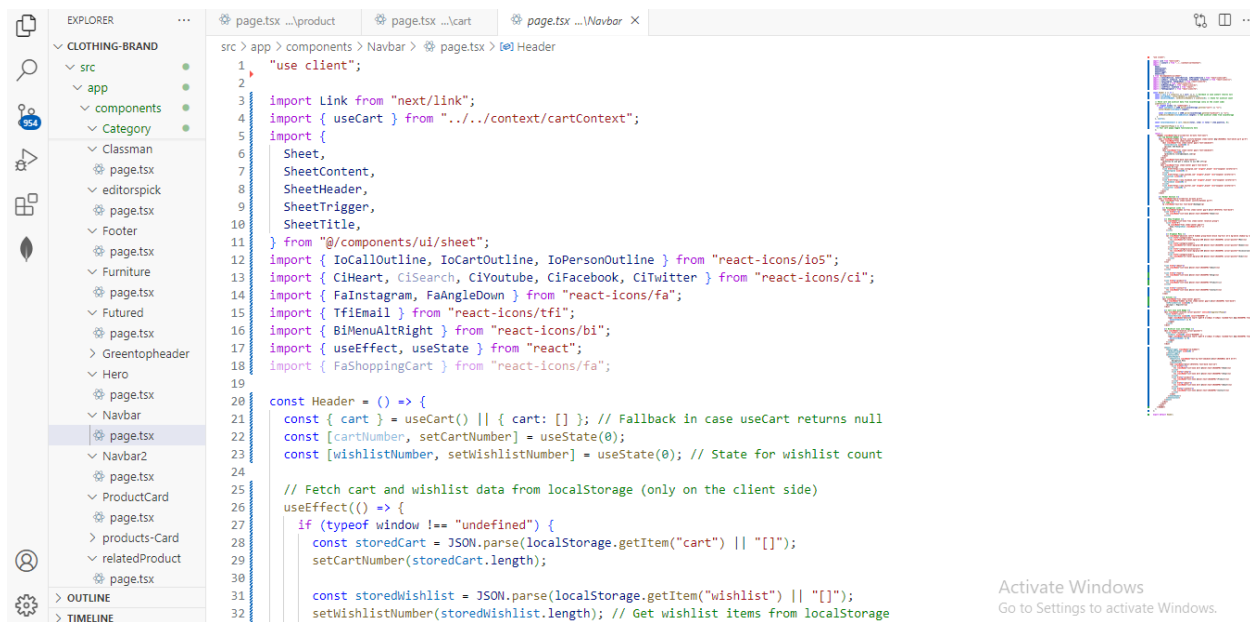
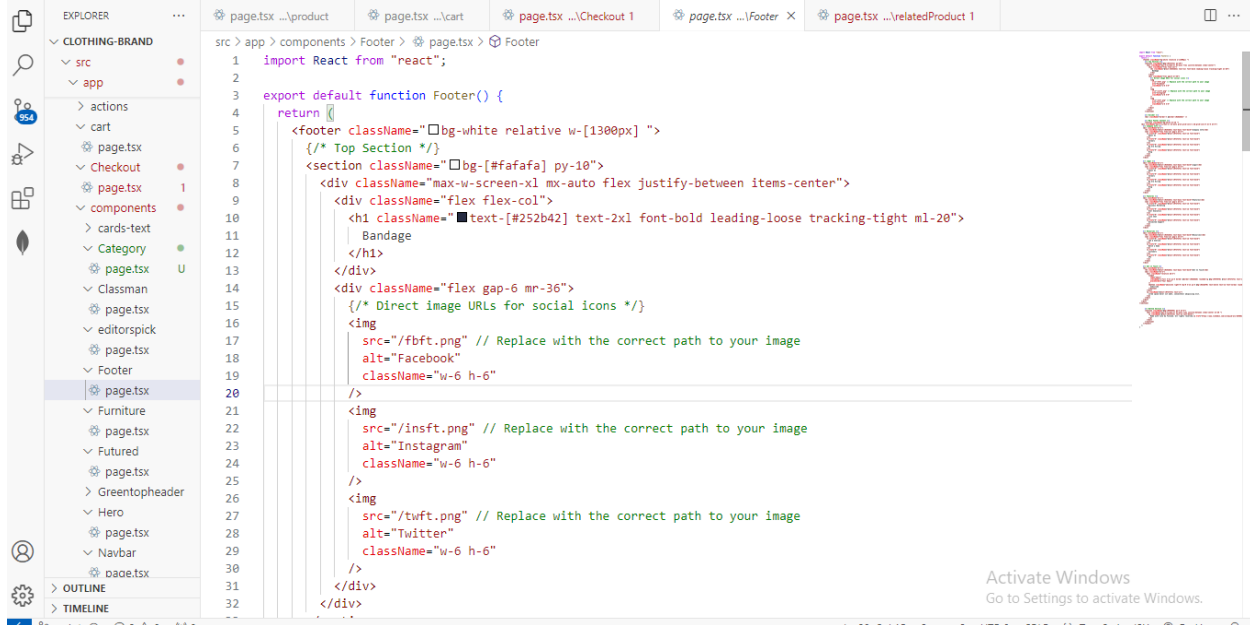




Header and Footer Component

The header component includes easy navigation links to essential pages such as Home, Shop, About, Blog, Product, and Contact. It also features options for users to log in or register, as well as quick access to the cart and wishlist. The cart icon displays the number of items in the cart, enhancing the user experience. The footer contains the website logo, making it clear and visually connected to the brand. It provides links to information pages like About Us, Contact, Privacy Policy, and Terms & Conditions. The footer also offers quick links to the Shop, Cart, Wishlist, and Login/Register pages. Social media links are included to allow users to follow the brand on Instagram, Facebook, Twitter, and YouTube. Lastly, the footer has a copyright notice, ensuring the website maintains professionalism and clarity.





Checkout Flow Component:

The Checkout Flow component displays an order summary that includes the items added to the cart, with a subtotal for the order. It then calculates and displays the shipping costs, followed by the grand total of the order. Below the order details, the customer's information is shown, including their name, shipping address, and contact details. This allows the customer to review their information before proceeding with the purchase. The component ensures a clear overview of all costs, including taxes and shipping. It also provides an easy-to-understand breakdown of the order for transparency. Finally, there's a confirmation step where customers can review and finalize their purchase details. This layout is designed to guide the user smoothly through the checkout process.

Main Info

Email

Password

Name

Telephone

Country

Pakistan



Zone

-- Please Select --



City

Order Summary

Sub-Total \$350.00

Flat Shipping Rate \$10.00

Total \$360.00

Update

Confirm Order

Activate Windows
Go to Settings to activate Windows.

I wanted to implement all the provided components into my UI, but due to time constraints, I have only been able to add the selected components for now. However, in the future, I plan to work on integrating all the components as described in the Day 4 document.