

Day 4 - Dynamic Frontend Components–AS Furniture

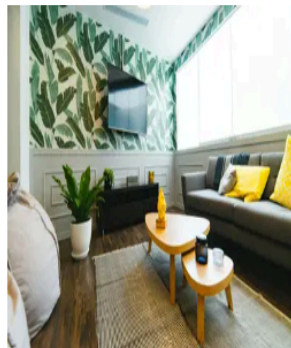
ScreenShots of:

Product Listing Component:



The Poplar suede sofa
980

popular products



Tropical Vibe
550



Sleek Living
300



Serene Seat
350



Nordic Elegance
280

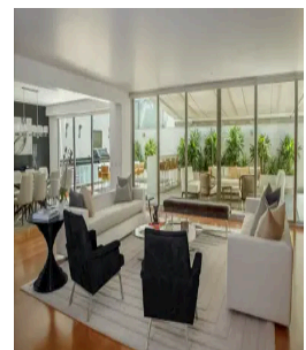


TimberCraft
320

popular products



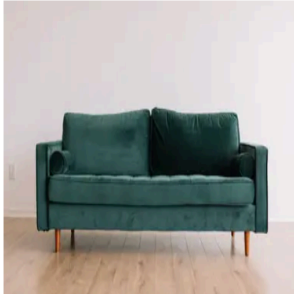
Timeless Elegance
320



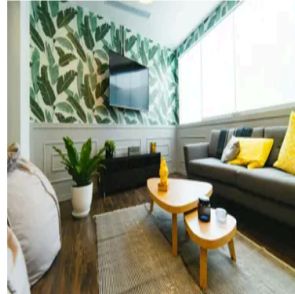
Modern Serenity
480

by Areesha khan

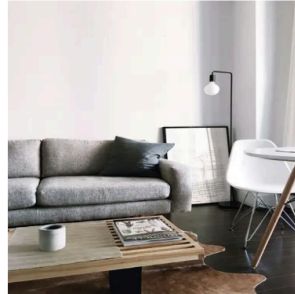
Category Component :



The Poplar suede sofa
980\$



Tropical Vibe
550\$



Sleek Living
300\$



Serene Seat
350\$


Product Detail Component :

Shows a product image, name, description, price ,tag Includes a "Add to cart" button and update quantity or decrease quantity button

Free delivery on all orders over £50 with code easter checkout

AvionHomeAboutAll ProductsCartSignUp

Q



Tropical Vibe

\$550

Description

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

Features:

- Premium material
- Handmade upholstery
- Quality timeless classic

Quantity : 50

Dimensions

Height

110cm

Width

75cm

Depth

50cm

Quantity

-

0

+

Add To Card

...

...

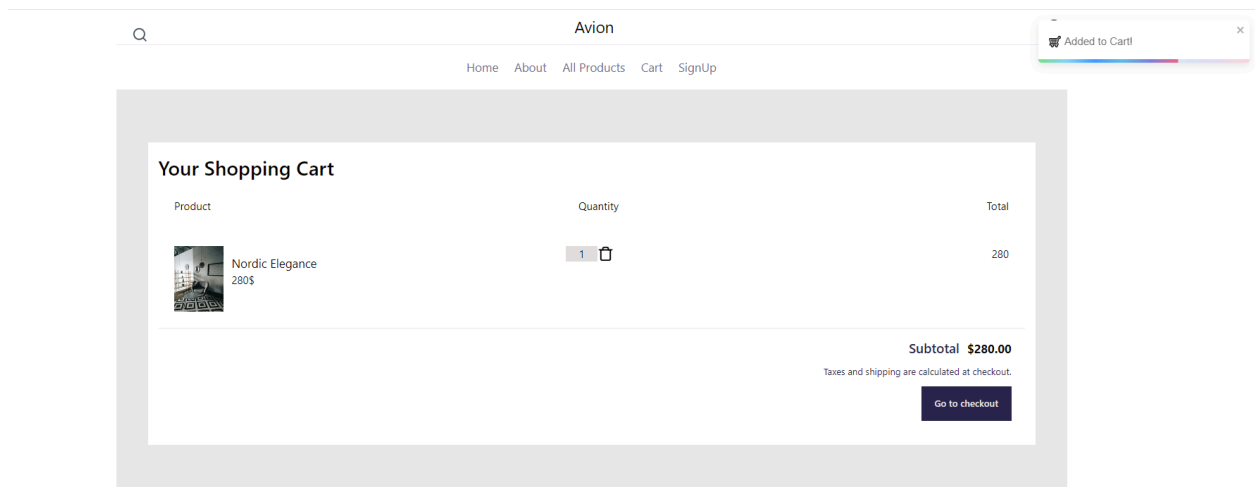
...

...

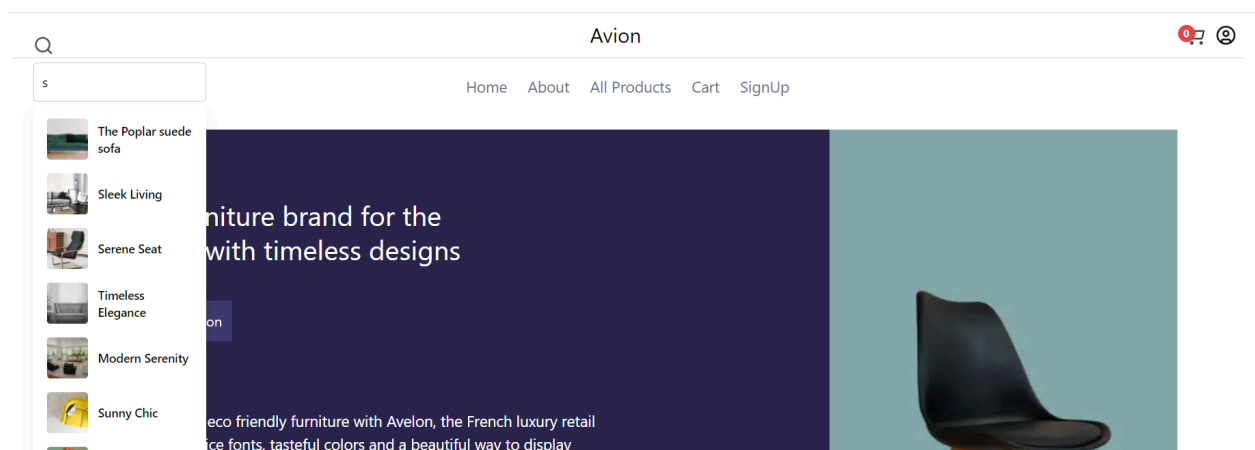
by Areesha khan

Cart Component

Displays each product in a card format, showing its image. name, description, price, and and subTotal amount



Search Bar:



by Areesha khan

Searching Product



Reflective Haven

\$300

Description

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

Features:

- Premium material
- Handmade upholstery
- Quality timeless classic

Quantity : 50

Dimensions

Height	Width	Depth
110cm	75cm	50cm

Quantity

- 0 +

Add To Card

by Areesha khan

Search bar Functionality:

- The searchUpdated function filters products based on the search query. It checks the name field of each product and displays matching results dynamically.
- The search bar updates the filteredProducts list in real-time as the user types in the search input.

```
"use client";
import React, { useState, useEffect } from "react"; 6.9k (gzipped: 2.7k)
import { useRouter } from "next/navigation"; 27.4k (gzipped: 6.5k)
import { client } from "@sanity/lib/client";
import { ProductTypes } from "../../types/products";
import { urlFor } from "@sanity/lib/image";
import Image from "next/image"; 16.9k (gzipped: 6.1k)
import SearchInput, { createFilter } from "react-search-input"; 17.7k (gzipped: 6.3k)
import { Search } from "lucide-react"; 1.3k (gzipped: 780)

const KEYS_TO_FILTERS = ["name"]; // Define the keys

const SearchComponent = () => {
  const [products, setProducts] = useState<ProductTypes[]>([]); // State for all products
  const [filteredProducts, setFilteredProducts] = useState<ProductTypes[]>([]); // Filtered product list
  const [loading, setLoading] = useState<boolean>(true); // Loading state
  const [showInput, setShowInput] = useState<boolean>(false); // Toggle search input visibility
  const [searchTerm, setSearchTerm] = useState<string>(""); // Track search term
  const router = useRouter(); // Next.js router hook

  // Fetch products from Sanity
  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const query = `[_type == "product"]`; // Query to fetch products
        const fetchedProducts: ProductTypes[] = await client.fetch(query);
        setProducts(fetchedProducts); // Save products to state
      } catch (error) {
        console.error("Error fetching products:", error);
      } finally {
        setLoading(false); // Set loading to false
      }
    };

    fetchProducts();
  }, []);

  // Update filtered products whenever the search term changes
  const searchUpdated = (term: string) => {
    setSearchTerm(term); // Update search term
    const filtered = products.filter(createFilter(term, KEYS_TO_FILTERS));
    setFilteredProducts(filtered);
  };

  // Handle product click and navigate to the product page
  const handleProductClick = (id: string) => {
    router.push(`/component/${id}`); // Navigate to the dynamic route
  };

  return (
    <div className="p-4 relative w-full">
      <div className="absolute w-auto z-50">
        { /* Search Icon */ }
        <div
          className="cursor-pointer text-gray-600 flex items-center gap-2 hover:scale-110"
          onClick={() => setShowInput(!showInput)} // Toggle input visibility
        >
          <Search size={24} />
        </div>

        { /* Show search input only when showInput is true */ }
        {showInput && (
```

by Areesha khan

Product List:

Displays each product in a card format, showing its image. name, description, price, and tags.

```
const Page = () => {
  const [products, setproducts] = useState<ProductTypes[]>([]);

  useEffect(() => {
    async function fetchData() {
      const fetchedData: ProductTypes[] = await client.fetch(allproducts);
      setproducts(fetchedData);
    }
    fetchData();
  }, []);

  return (
    <>
      <div className="padding-x max-container">
        <div className="max-lg:justify-items-center grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 mt-6 gap-4 py-6">
          {products.map((product) => (
            <div key={product._id} className="product-card">
              <Link
                href={`./sections/${product.name
                  .toLowerCase()
                  .replace(/ /g, "-")}`}
              >
                {/* Product Image */}
                <Image
                  src={urlFor(product.image).url()}
                  alt={product.name}
                  height={375}
                  width={375}
                  priority
                  className="h-[300px] w-[300px] max-lg:w-[270px] max-lg:h-[300px] translate transition duration-300 ease-in-out hover:scale-105"}
                />

                {/* Product Details */}
                <div className="mt-3">
                  <h2 className="text-DarkPrimary text-lg leading-normal">
                    {product.name}
                  </h2>
                  <p className="text-DarkPrimary text-lg leading-normal">
                    {product.price}$
                  </p>
                </div>
                <div className="mt-2 flex gap-2 flex-wrap">
                  {product.tags.map((tag, index) => (
                    <span
                      key={index}
                      className="px-2 py-1 text-sm bg-gray-200 text-gray-700 rounded-lg"
                    >
                      {tag}
                    </span>
                  ))}
                </div>
              </Link>
            </div>
          ))}
        </div>
      </div>
    </>
  );
};

export default Page;
```

By Areesha khan

Cart Functionality:

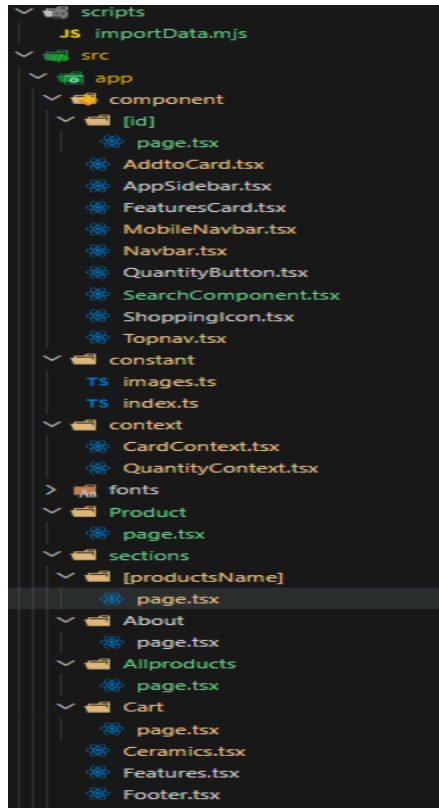
- **addToCart**: Allows users to add a product to the cart. It checks if the product already exists in the cart and either increments its quantity or adds it as a new entry.
- **getSubTotal**: Calculates the total price of all items in the cart by multiplying the quantity of each product by its price and summing up the values.
- **removeFromCart**: Enables users to remove a product from the cart.

```
1 // use client;
2 import { StaticImageData } from "next/image"; 16.9k (gzipped: 6.1k)
3 import { useContext, createContext, useState, ReactNode } from "react"; 4.5k (gzipped: 2k)
4
5 type CartItem = {
6   img: string | StaticImageData;
7   name: string;
8   price: number | string;
9   quantity: number;
10 };
11
12 type CartType = {
13   cartItems: CartItem[];
14   addToCart: (item: CartItem) => void;
15   getSubTotal: () => number;
16   removeFromCart: (itemName: string) => void;
17 };
18
19 const CartContext = createContext<CartType | undefined>(undefined);
20
21 export const CartProvider = ({ children }: { children: ReactNode }) => {
22   const [cartItems, setCartItems] = useState<CartItem[]>([]);
23
24   // add product function
25   const addToCart = (item: CartItem) => {
26     console.log("Cart Items:", cartItems);
27
28     setCartItems((prevItems) => {
29       const existingItem = prevItems.find((i) => i.name === item.name);
30
31       if (existingItem) {
32         return prevItems.map((i) =>
33           i.name === item.name ? { ...i, quantity: i.quantity + 1 } : i
34         );
35       }
36       return [...prevItems, item];
37     });
38   };
39
40   // Sub total function
41   const getSubTotal = () => {
42     const total = cartItems.reduce((acc, item) => {
43       // Extract the numeric part of the price string (e.g., "£250" becomes 250)
44       const price =
45         parseFloat(item.price.toString().replace(/[^d.-]/g, "")) || 0;
46       const quantity = parseInt(item.quantity.toString(), 10) || 1;
47       return acc + price * quantity; // Calculate total
48     }, 0);
49
50     return isNaN(total) ? 0 : total;
51   };
52
53   // Remove Product
54   const removeFromCart = (itemName: string) => {
55     setCartItems((prevItems) =>
56       prevItems.filter((item) => item.name !== itemName)
57     );
58   };
59
60   return (
61     <CartContext.Provider
62       value={{ cartItems, addToCart, getSubTotal, removeFromCart }}
63     >
64       {children}
65     </CartContext.Provider>
66   );
67 };
68
69 export const useCart = () => {
```

by Areesha khan

Product Details (Dynamic Routes):

File Structure



by Areesha khan

Code

Dynamic routing

Dynamic routing is used to handle routes like /productsNames/page.tsx for individual product pages.

1. Create Dynamic Route File: In the pages directory, create a [productsNames]/page.tsx file inside the sections folder: pages/sections/[productsNames]/page.tsx
2. Get Product Data Dynamically:

```
14 // pages/sections/[productsNames]/page.tsx
15
16 const ProductDetailPage = async ({
17   params,
18   // params: { productsName: string },
19   // => { productsName: string }
20   const { productsName } = params;
21
22   // Fetch the product data
23   const product: ProductTypes | null = await client.fetch(
24     `[_type == "product" && slug.current == ${slug}[0]]{
25       _id,
26       name,
27       price,
28       description,
29       "image": image.asset->url,
30       tag,
31       dimensions,
32       quantity,
33       features
34     }`
35   )( slug: productsName );
36
37   if (!product) {
38     notFound();
39   }
40
41   return (
42     <Topnav />
43     <MobileNav />
44     <div className="max-container pt-14 padding-x">
45       <div className="flex justify-start items-start gap-24 max-lg:flex-col">
46         <div>
47           <Image
48             src={urlFor(product.image).url()}
49             alt={ `Product: ${product.name} ` }
50             width={400}
51             height={300}
52             className="w-auto h-auto max-w-[400px] max-h-[300px]"
53             priority
54           />
55         </div>
56         <div className="flex justify-center flex-col">
57           <h1 className="text-2xl text-black tracking-wide">
58             {product.name}
59           </h1>
60           <p className="text-lg mt-2 text-black tracking-wide">
61             ${product.price}
62           </p>
63           <div className="mt-8 w-full">
64             <h4 className="text-lg text-gray-600">Description</h4>
65             <p className="max-w-[500px] mt-5 text-gray-600">
66               {product.description}
67             </p>
68             <div className="mt-5">
69               <h4 className="text-lg text-gray-600">Features</h4>
70               <ul className="mt-5 text-gray-600 list-disc pl-5">
71                 {product.features.map((feature, index) => (
72                   <li key={index}>{feature}</li>
73                 ))}
74               </ul>
75             </div>
76             <div className="mt-5 flex gap-2 flex-wrap">
77               {product.tags.map((tag, index) => (
78                 <span
79                   key={index}
80                   className="px-2 py-1 text-sm bg-gray-200 text-gray-700 rounded-lg"
81                 >
82                   {tag}
83                 </span>
84               ))}
85             </div>
86           </div>
87         </div>
88       </div>
89     </div>
90   );
91 }
```

by Areesha khan

API Integration:

1. Sanity API Fetching Logic: Since you're using Sanity CMS, you need to ensure data fetching is properly managed.

Sanity Client Setup (if not already set up): In your lib folder, configure the Sanity client:

Fetching Data in useEffect

```
12 import { useState } from 'react'
13
14 const ProductDetailPage = async ({
15   params,
16 }) => {
17   const { productsName } = params
18   // Fetch the product data
19   const product = ProductTypes | null = await client.fetch(
20     `[_type == "product" && slug.current == ${slug}[0]{
21       _id,
22       name,
23       price,
24       description,
25       "image": image.asset->url,
26       tags,
27       dimensions,
28       quantity,
29       features
30     }`,
31     { slug: productsName }
32   )
33
34   if (!product) {
35     notFound()
36   }
37
38   return (
39     <>
40       <Topnav />
41       <MobileNav />
42       <div className="max-container pt-14 padding-x">
43         <div className="flex justify-start items-start gap-24 max-lg:flex-col">
44           <div>
45             <Image
46               src={urlFor(product.image).url()}
47               alt={`Product: ${product.name}`}
48               width={400}
49               height={300}
50               className="w-auto h-auto max-w-[400px] max-h-[300px]"
51               priority
52             />
53           </div>
54           <div className="flex justify-center flex-col">
55             <h1 className="text-2xl text-black tracking-wide">
56               {product.name}
57             </h1>
58             <p className="text-lg mt-2 text-black tracking-wide">
59               ${product.price}
60             </p>
61             <div className="mt-8 w-full">
62               <h4 className="text-lg text-gray-600">Description</h4>
63               <p className="max-w-[500px] mt-5 text-gray-600">
64                 {product.description}
65               </p>
66               <div className="mt-5">
67                 <h4 className="text-lg text-gray-600">Features</h4>
68                 <ul className="mt-5 text-gray-600 list-disc pl-5">
69                   {product.features.map((feature, index) => (
70                     <li key={index}>{feature}</li>
71                   ))}
72                 </ul>
73               </div>
74             </div>
75             <div className="mt-5 flex gap-2 flex-wrap">
76               {product.tags.map((tag, index) => (
77                 <span
78                   key={index}
79                   className="px-2 py-1 text-sm bg-gray-200 text-gray-700 rounded-lg"
80                 >{tag}</span>
81               ))}
82             </div>
83           </div>
84         </div>
85       </div>
86     </>
87   )
88 }
```

by Areesha khan

Final checkList:

Day 4

Frontend Component Development ✓

Styling and Responsiveness ✓

Code Quality ✓

Documentation and Submission ✓

Final Review ✓

By Areesha khan