

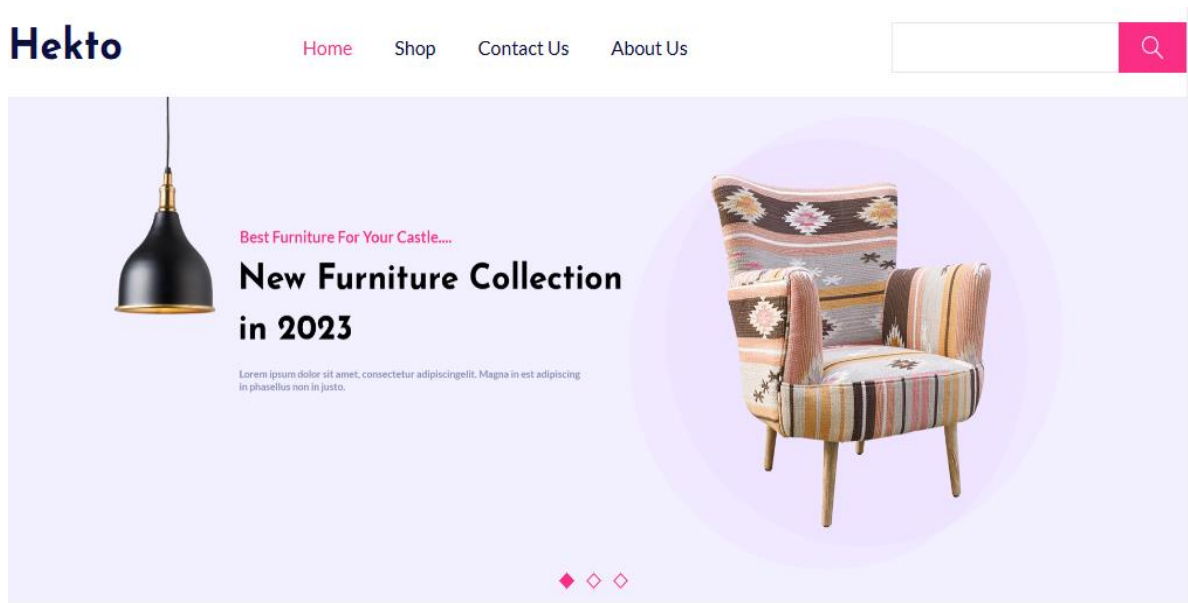
Name: Hammad Noor Khan

Roll No: 00457174

Day 4 - Building Dynamic Frontend Components for Your Marketplace **[Bandage-E-Commerce]**

Introduction

This report documents the entire process I followed to develop the dynamic frontend components for my e-commerce platform. The goal of this platform is to create a seamless shopping experience for users, featuring dynamic product listings, category filtering, search functionality, and other essential e-commerce features. Below, I outline each step of the development process, challenges I faced, solutions I implemented, and additional ideas for improvement.



Process Overview

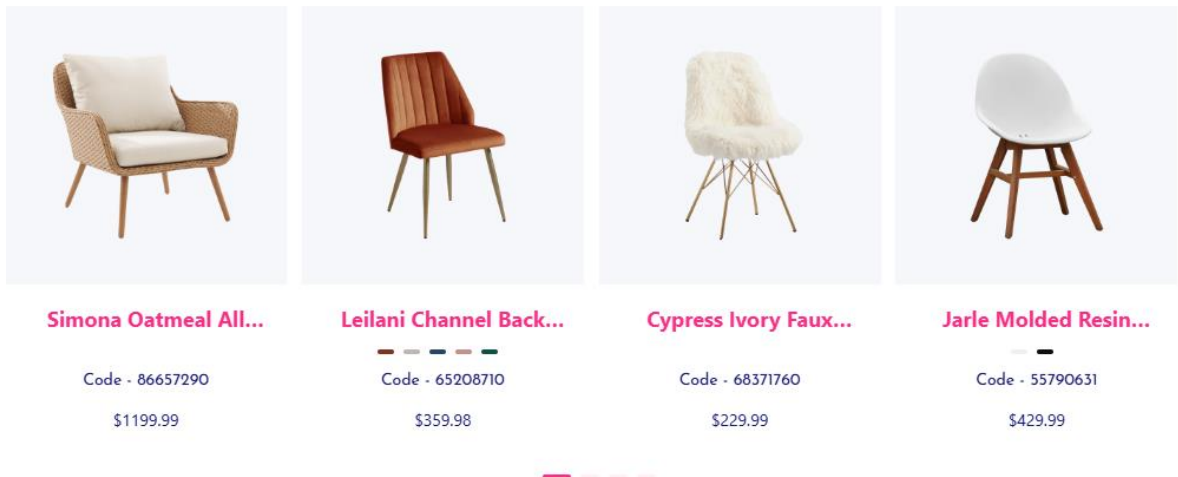
1. Product Card

- **What I Did:**
 - Designed a ProductCard component to display details like name, price, and image.
 - Styled the cards using Tailwind CSS for responsiveness.
 - Added interactive button for "Add to Cart".
- **Challenges I Faced:**
 - Handling inconsistent product image sizes.
- **Solution:**
 - Adjust it by Tailwind CSS properties to maintain uniformity.

- **Additional Features Can Be Added:**

- Displayed a quick “View Details” button for a modal preview.
- Highlighted items on sale with a discount badge.

Featured Products



```

1  "use client"
2
3  import type React from "react"
4  import { useState, useEffect } from "react"
5  import Image from "next/image"
6  import Link from "next/link"
7  import { client } from "@sanity/lib/client"
8  import { useCart } from "../ContextApi/CartProvider"
9  import { NavBar } from "../components/NavBar"
10 import Footer from "../components/Footer"
11
12 interface Product {
13   _id: string
14   title: string
15   price: number
16   description: string
17   discountPercentage: number
18   imageUrl: string
19   tags: string[]
20   slug: { current: string }
21 }
22
23 const ProductCard: React.FC = () => {
24   const [products, setProducts] = useState<Product[]>([])
25   const { addToCart } = useCart()
26
27   const fetchProducts = async () => {
28     try {
29       const query = `
30         *[_type == "product"] {
31           _id,
32           title,
33           price,
34           description,
35           discountPercentage,
36           "imageUrl": productImage.asset->url,
37           tags,
38           slug
39         }
40       `
41       const data = await client.fetch<Product[]>(query)
42       setProducts(data)
43     } catch (error) {
44       console.error("Error fetching products:", error)
45     }
46   }
47
48   useEffect(() => {
49     fetchProducts()
50   }, [])
51
52   return (
53     <div>
54       <h3>Product Card</h3>
55       <ul>
56         {products.map((product) => (
57           <li>
58             <div>
59               <img alt={product.imageUrl} />
60               <div>
61                 <h4>{product.title}</h4>
62                 <p>{product.description}</p>
63                 <p>Price: {product.price}</p>
64                 <p>Discount: {product.discountPercentage}</p>
65                 <p>Tags: {product.tags}</p>
66                 <p>Slug: {product.slug}</p>
67               </div>
68             </div>
69           </li>
70         ))}
71       </ul>
72     </div>
73   )
74 }

```

```
src > app > Products > [slug] > page.tsx > ProductPage
1  import { client } from "@sanity/lib/client";
2  import { urlFor } from "@sanity/lib/image";
3  import Image from "next/image";
4  import { NotFound } from "next/navigation";
5  import { PortableText } from "@portabletext/react";
6  import { CiHeart } from "react-icons/ci";
7  import { FiShoppingCart } from "react-icons/fi";
8  import { FaEye } from "react-icons/fa6";
9  import { NavBar } from "@app/components/NavBar";
10 import Footer from "@app/components/Footer";
11
12 export const revalidate = 60; // Revalidation time in seconds
13
14 // `generateStaticParams` generates the paths for dynamic routes
15 export async function generateStaticParams() {
16   const query = `*[_type == 'product']{ "slug": slug.current }`;
17   const slugs = await client.fetch(query);
18   const slugRoutes = slugs.map((item: { slug: string }) => item.slug);
19
20   return slugRoutes.map((slug: string) => ({
21     slug,
22   }));
23 }
24
25 // The page function for rendering a single product
26 export default async function ProductPage({
27   params: { slug },
28 }: {
29   params: { slug: string };
30 }) {
31   const query = `*[_type == 'product' && slug.current == ${slug}][0]{
32     title, price, description, "imageUrl": productImage.asset->url, tags, content
33   }`;
34   const product = await client.fetch(query, { slug });
35
36   if (!product) {
37     NotFound();
38   }
39 }
```

2. Top Product Listing

- **What I Did:**
 - Created a Top Product List component to display a grid of products dynamically fetched from the API which fetch 8 products.
- **Challenges I Faced:**
 - Managing large datasets.
- **Solution:**
 - Added a view more button and link it to page having more products
- **Additional Features Can Be Added:**
 - Implement control over infinite scrolling.

Featured Products



Simona Oatmeal All...

Code - 86657290

\$1199.99



Leilani Channel Back...

Code - 65208710

\$359.98



Cypress Ivory Faux...

Code - 68371760

\$229.99



Jarle Molded Resin...

Code - 55790631

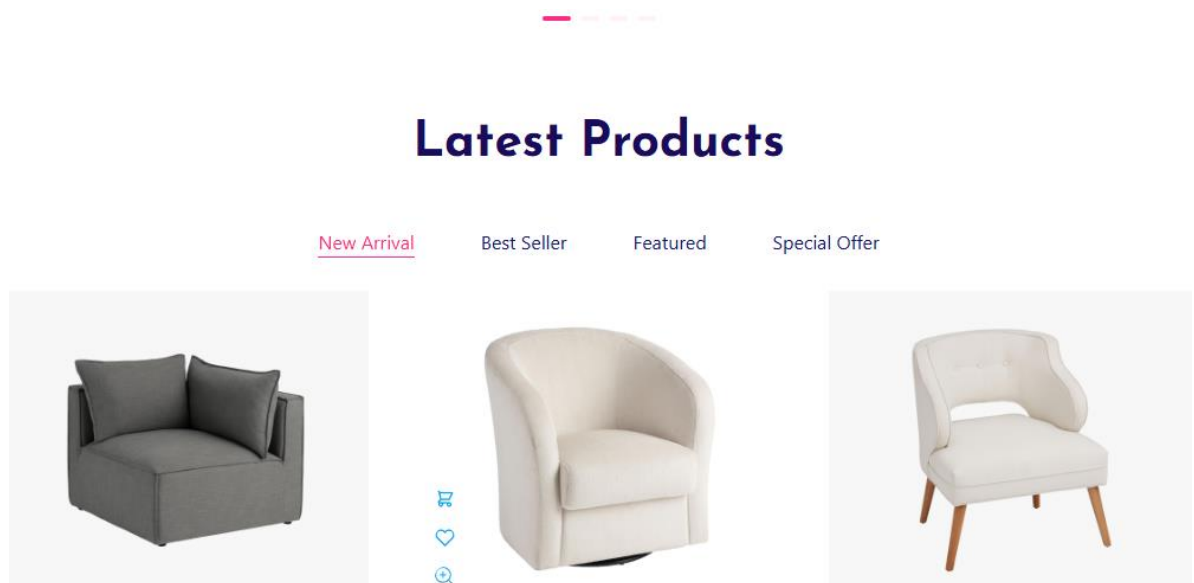
\$429.99

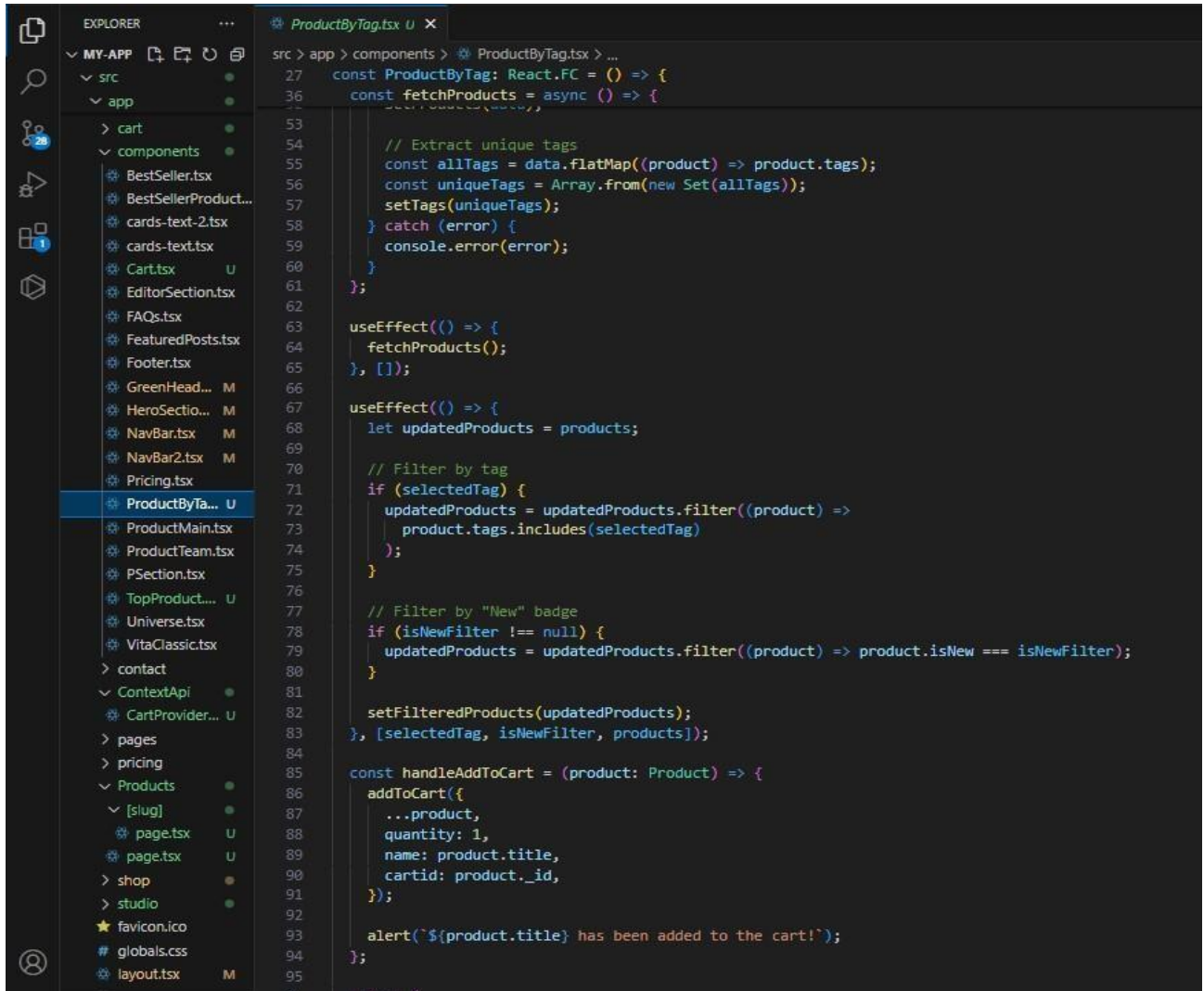
```
EXPLORER
MY-APP
  src
    app
      cart
      components
        BestSeller.tsx
        BestSellerProduct...
        cards-text-2.tsx
        cards-text.tsx
        Cart.tsx
        EditorSection.tsx
        FAQs.tsx
        FeaturedPosts.tsx
        Footer.tsx
        GreenHead... M
        HeroSectio... M
        NavBar.tsx M
        NavBar2.tsx M
        Pricing.tsx
        ProductByTa... U
        ProductMain.tsx
        ProductTeam.tsx
        PSection.tsx
        TopProduct.... U
        Universe.tsx
        VitaClassic.tsx
      contact
      ContextApi
      CartProvider... U
      pages
      pricing
      Products
        [slug]
          page.tsx U
          page.tsx U
      shop
      studio
      favicon.ico
      globals.css
      layout.tsx M
    OUTLINE

TopProduct.tsx U X
src > app > components > TopProduct.tsx > [0] ProductCard > products.map() callback
1  "use client";
2
3  import type React from "react";
4  import { useState, useEffect } from "react";
5  import Image from "next/image";
6  import Link from "next/link";
7  import { client } from "@sanity/lib/client";
8  import { useCart } from "../ContextApi/CartProvider";
9  import { FaArrowAltCircleRight } from "react-icons/fa";
10
11 interface Product {
12   _id: string;
13   title: string;
14   price: number;
15   description: string;
16   discountPercentage: number;
17   imageUrl: string;
18   tags: string[];
19   slug: { current: string };
20 }
21
22 Tabnine | Edit | Explain
23 const ProductCard: React.FC = () => {
24   const [products, setProducts] = useState<Product[]>([]);
25   const { addToCart } = useCart();
26
27   // Fetch 8 products from the Sanity database
28   const fetchProducts = async () => {
29     try {
30       const query = `
31         *[_type == "product"][0..7] { // Fetch the first 8 products
32           _id,
33           title,
34           price,
35           description,
36           discountPercentage,
37           "imageUrl": productImage.asset->url,
38           tags,
39           slug
40         }
41       `;
42       const data = await client.fetch<Product[]>(query);
43       setProducts(data);
44     } catch (error) {
45       console.error("Error fetching products:", error);
46     }
47   };
48 }
```

3. Tags and New Arrival

- **What I Did:**
 - Implemented dynamic filtering based on New Arrival and tags.
 - Used dropdowns for user input.
- **Challenges I Faced:**
 - Efficiently handling API calls.
- **Solution:**
 - Applied debouncing and caching mechanisms to optimize performance.
- **Additional Features Can Be Added:**
 - Implemented a "Clear All Filters" button to reset filters for better usability.





```
src > app > components > ProductByTag.tsx > ...
27 const ProductByTag: React.FC = () => {
36   const fetchProducts = async () => {
37     // ...
53   }
54
55   // Extract unique tags
56   const allTags = data.flatMap((product) => product.tags);
57   const uniqueTags = Array.from(new Set(allTags));
58   setTags(uniqueTags);
59 } catch (error) {
60   console.error(error);
61 }
62
63 useEffect(() => {
64   fetchProducts();
65 }, []);
66
67 useEffect(() => {
68   let updatedProducts = products;
69
70   // Filter by tag
71   if (selectedTag) {
72     updatedProducts = updatedProducts.filter((product) =>
73       product.tags.includes(selectedTag)
74     );
75   }
76
77   // Filter by "New" badge
78   if (isNewFilter !== null) {
79     updatedProducts = updatedProducts.filter((product) => product.isNew === isNewFilter);
80   }
81
82   setFilteredProducts(updatedProducts);
83 }, [selectedTag, isNewFilter, products]);
84
85 const handleAddToCart = (product: Product) => {
86   addToCart({
87     ...product,
88     quantity: 1,
89     name: product.title,
90     cartid: product._id,
91   });
92
93   alert(`${product.title} has been added to the cart!`);
94 }
95
```

4. Pagination

- **What I Did:**
 - Developed a Pagination Component to divide products into smaller, navigable pages.
 - Styled navigation buttons for user-friendly interaction.
- **Challenges I Faced:**
 - Smooth page transitions.
- **Solution:**
 - I will solve it after hackathon.
- **Additional Features Can Be Added:**
 - Integrated page number highlighting to indicate the current page. ○ Used
 - Next.js's `getStaticProps` and `getServerSideProps` for efficient data fetching. ○
 - Added "Jump to Page" functionality for quicker navigation.

Latest Products

New Arrival

Best Seller

Featured

Special Offer



5. Add to Cart

- **What I Did:**
 - Developed the cart functionality using the React Context API to manage the global state.
 - Enabled dynamic quantity updates for cart items.
- **Challenges I Faced:**
 - Syncing cart state across multiple components.
- **Solution:**
 - Used local storage to persist cart data.
- **Additional Features Added:**
 - Integrated a cart summary that dynamically updates the total price and discounts

Latest Products

New Arrival

Best Seller

Featured

Special Offer



(225) 555-9118

michelle.rivera@example.com

Follow Us and get a chance to win 80% off

Follow Us:   

Bandage

[Home](#) [Shop](#) [About](#) [Blog](#) [Contact](#) [Pages](#) [Products](#)

[Login / Register](#)   1

Your Shopping Cart

IMAGE	NAME	PRICE	QUANTITY	ACTION
	Retro Vibe	\$340	1	Remove
	Marble Ease	\$419	1	Remove
	Rustic Vase Set	\$210	1	Remove
	Zen Table	\$250	1	Remove

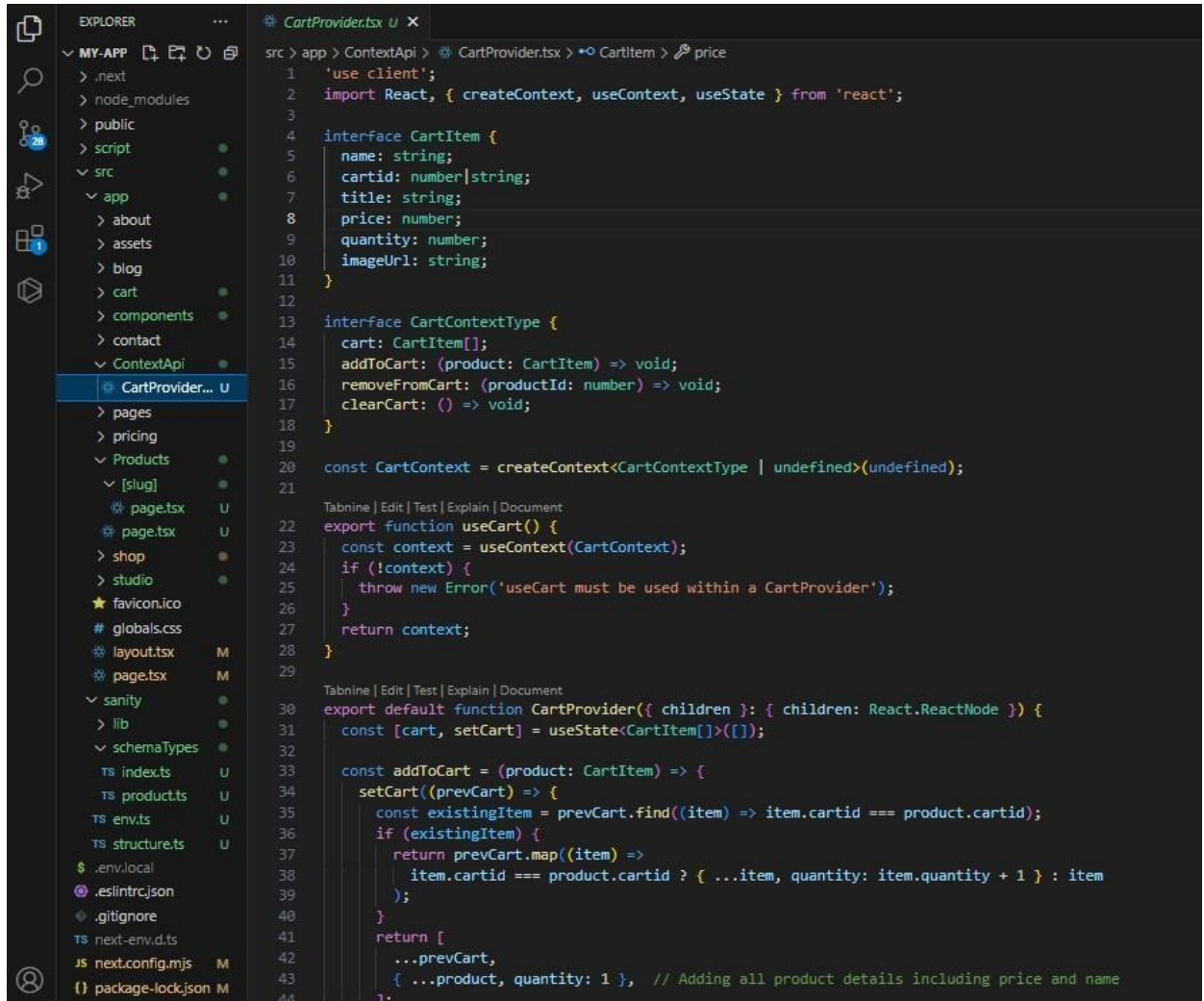
Total Summary

Total Price:

\$1219.00

[Continue Shopping](#)

[Proceed to Checkout](#)



The screenshot shows a VS Code editor with a project named 'MY-APP'. The Explorer sidebar on the left shows the file structure, including folders like 'src', 'app', and 'ContextApi', and files like 'CartProvider.tsx'. The main editor area displays the code for 'CartProvider.tsx'. The code defines a 'CartItem' interface, a 'CartContextType' interface, and a 'CartContext' using 'createContext'. It also defines a 'useCart' hook and a 'CartProvider' component that uses 'useState' to manage the cart state and 'useContext' to provide it. The 'addToCart' function is implemented to update the cart state by adding a new item or increasing the quantity of an existing item.

```
src > app > ContextApi > CartProvider.tsx > CartItem > price
1 'use client';
2 import React, { createContext, useContext, useState } from 'react';
3
4 interface CartItem {
5   name: string;
6   cartid: number|string;
7   title: string;
8   price: number;
9   quantity: number;
10  imageUrl: string;
11 }
12
13 interface CartContextType {
14   cart: CartItem[];
15   addToCart: (product: CartItem) => void;
16   removeFromCart: (productId: number) => void;
17   clearCart: () => void;
18 }
19
20 const CartContext = createContext<CartContextType | undefined>(undefined);
21
22 Tabnine | Edit | Test | Explain | Document
23 export function useCart() {
24   const context = useContext(CartContext);
25   if (!context) {
26     throw new Error('useCart must be used within a CartProvider');
27   }
28   return context;
29 }
30
31 Tabnine | Edit | Test | Explain | Document
32 export default function CartProvider({ children }: { children: React.ReactNode }) {
33   const [cart, setCart] = useState<CartItem[]>([]);
34
35   const addToCart = (product: CartItem) => {
36     setCart((prevCart) => {
37       const existingItem = prevCart.find((item) => item.cartid === product.cartid);
38       if (existingItem) {
39         return prevCart.map((item) =>
40           item.cartid === product.cartid ? { ...item, quantity: item.quantity + 1 } : item
41         );
42       }
43       return [
44         ...prevCart,
45         { ...product, quantity: 1 }, // Adding all product details including price and name
46       ];
47     });
48   };
49 }
```

6. Login and Sign-Up Page

- **What I Did:**
 - Built user authentication pages for login and sign-up with form validation.
 - Integrated Clerk for secure authentication and user management.
- **Challenges I Faced:**
 - Handling errors during login and sign-up.
- **Solution:**
 - Provided user-friendly error messages for better feedback.
- **Additional Features Can Be Added:**
 - Added "Forgot Password" functionality with email recovery support.
 - Enabled login via social accounts like Google.

Summary:

By the end of the development process, I successfully delivered the following:

1. A fully functional product listing page displaying dynamic data from the API.
2. Individual product detail pages implemented with dynamic routing.
3. Advanced filters for New Arrival and tags.
4. Pagination for better user experience with large datasets.
5. Responsive and professional styling for all components.
6. Modular and reusable components for future scalability.
7. Enhanced cart and user authentication functionality for a smoother shopping experience.