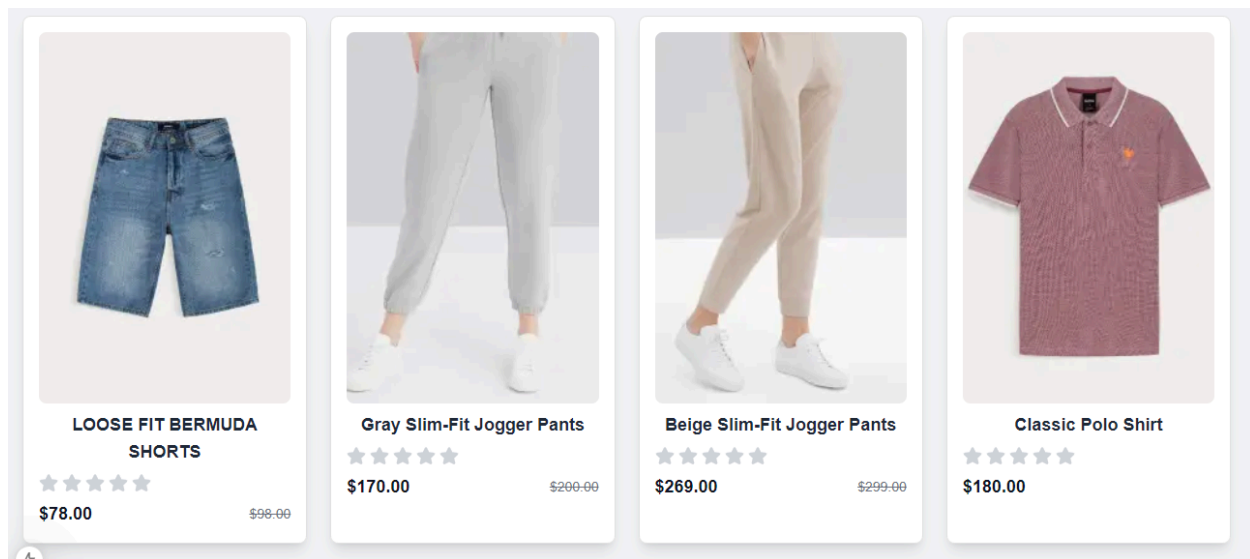


DAY 4 - BUILDING DYNAMIC FRONTEND COMPONENTS - "[General E-Commerce]"

1. Functional Deliverables:

Productlist



Top Selling



Vertical Striped Shirt



\$229.00

~~\$458.00~~



Casual Green Bomber Jacket



\$300.00

~~\$375.00~~



COURAGE GRAPHIC T-SHIRT



\$145.00



Black Striped T-Shirt



\$120.00

Product detail pages



Classic Polo Shirt

Classic Polo Shirt Upgrade your wardrobe with this timeless classic polo shirt, perfect for any occasion. Crafted from premium-quality fabric, it offers a soft, breathable, and comfortable fit that lasts all day. Featuring a stylish collar, button placket, and short sleeves, this versatile polo blends elegance with a casual touch. Key features: Made with durable and lightweight material Available in a variety of colors and sizes Easy to pair with jeans, chinos, or shorts for a polished look Ideal for work, casual outings, or weekend wear Stay effortlessly stylish with this must-have polo shirt!

\$180

Category: **tshirt**

Rating: **/5**

Add to Cart

Product Details



Black Striped T-Shirt

Elevate your casual style with this sporty and timeless raglan t-shirt. Featuring a crisp white base adorned with vertical pinstripes, it blends a retro vibe with modern appeal. The contrasting black raglan sleeves add a bold, athletic touch, making it a versatile piece for any wardrobe. Key Features: Soft and breathable fabric for all-day comfort Classic pinstripe design for a clean, stylish look Contrasting raglan sleeves for a sporty edge Regular fit with a crew neckline Perfect for casual outings or active days, pair this tee with your favorite jeans or joggers for effortless

\$120

Category: **tshirt**

Rating: **5**

Add to Cart

Searchbar

t-shirt

Q



Gradient Graphic T-shirt

Add a bold and artistic touch to your wardrobe with this unique graphic t-shirt. Featuring an eye-catching abstract swirl design in vibrant colors, it exudes energy and individuality. The "Just Walk Forward" slogan adds a motivational element, making this tee perfect for those who love to express



Gradient Graphic T-shirt

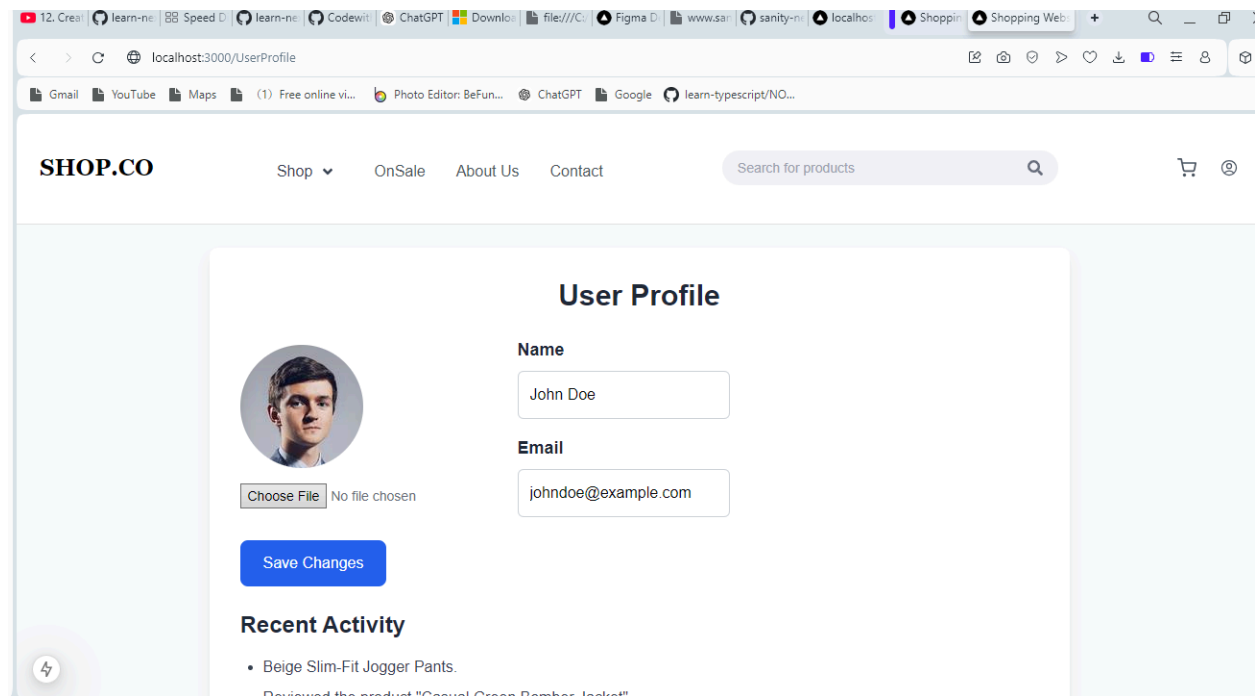
Add a bold and artistic touch to your wardrobe with this unique graphic t-shirt. Featuring an eye-catching abstract swirl design in vibrant colors, it exudes energy and individuality. The "Just Walk Forward" slogan adds a motivational element, making this tee perfect for those who love to express



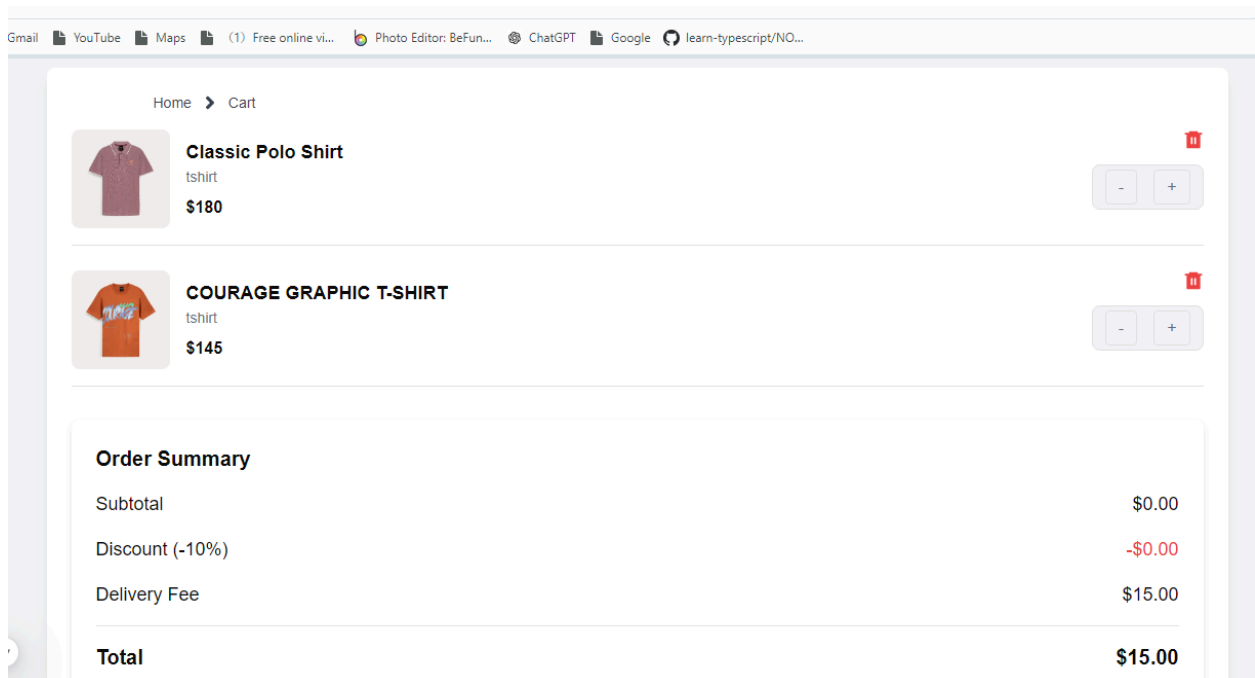
Sleeve Stripe T-Shirt

This product is a vibrant orange and black striped t-shirt with a sporty design. The shirt features vertical pinstripes in black on an orange base, complemented by contrasting black raglan sleeves for a casual, athletic-inspired look. Ideal for adding a bold touch to your casual wardrobe, this tee combines comfort and

User profile



Cart page



Reviews

Write a Review

Rating: ★ ★ ★ ★ ★

Checkout

[Cart](#) < [Checkout](#)

Checkout

Full Name

Email

Address

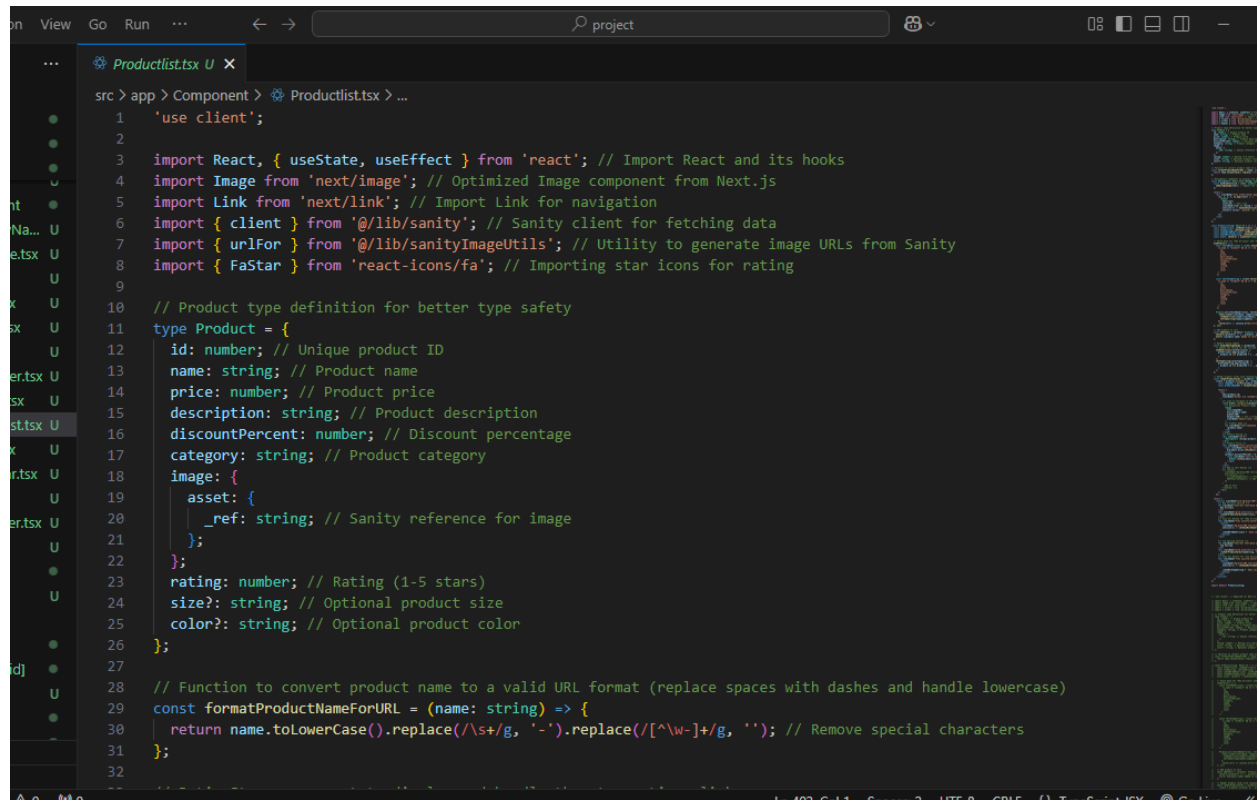
City

Postal Code

Order Summary

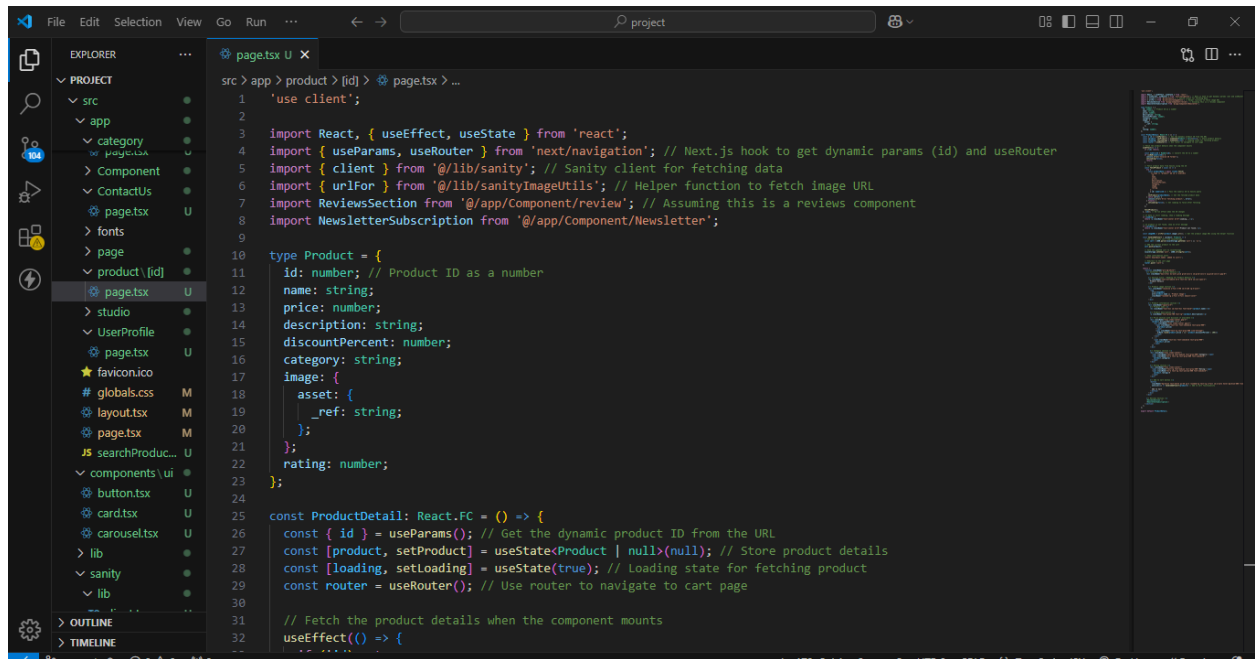
Code Deliverables

Productlist code

A screenshot of a code editor window showing a file named 'Productlist.tsx'. The editor has a dark theme. The file path in the breadcrumb is 'src > app > Component > Productlist.tsx'. The code is a TypeScript file for a React component. It starts with a 'use client' directive. Imports include React, useState, and useEffect from 'react'; Image from 'next/image'; Link from 'next/link'; client from '@lib/sanity'; urlFor from '@lib/sanityImageUtils'; and FaStar from 'react-icons/fa'. A TypeScript interface 'Product' is defined with fields: id (number), name (string), price (number), description (string), discountPercent (number), category (string), image (an object with an 'asset' field containing a '_ref' string), rating (number), size (optional string), and color (optional string). A utility function 'formatProductNameForURL' is defined, which takes a string and returns a formatted URL string by replacing spaces with dashes and removing special characters. The code is partially visible, with lines 1 through 32 shown.

```
1 'use client';
2
3 import React, { useState, useEffect } from 'react'; // Import React and its hooks
4 import Image from 'next/image'; // Optimized Image component from Next.js
5 import Link from 'next/link'; // Import Link for navigation
6 import { client } from '@lib/sanity'; // Sanity client for fetching data
7 import { urlFor } from '@lib/sanityImageUtils'; // Utility to generate image URLs from Sanity
8 import { FaStar } from 'react-icons/fa'; // Importing star icons for rating
9
10 // Product type definition for better type safety
11 type Product = {
12   id: number; // Unique product ID
13   name: string; // Product name
14   price: number; // Product price
15   description: string; // Product description
16   discountPercent: number; // Discount percentage
17   category: string; // Product category
18   image: {
19     asset: {
20       _ref: string; // Sanity reference for image
21     };
22   };
23   rating: number; // Rating (1-5 stars)
24   size?: string; // Optional product size
25   color?: string; // Optional product color
26 };
27
28 // Function to convert product name to a valid URL format (replace spaces with dashes and handle lowercase)
29 const formatProductNameForURL = (name: string) => {
30   return name.toLowerCase().replace(/\s+/g, '-').replace(/[^\w-]+/g, ''); // Remove special characters
31 };
32
```

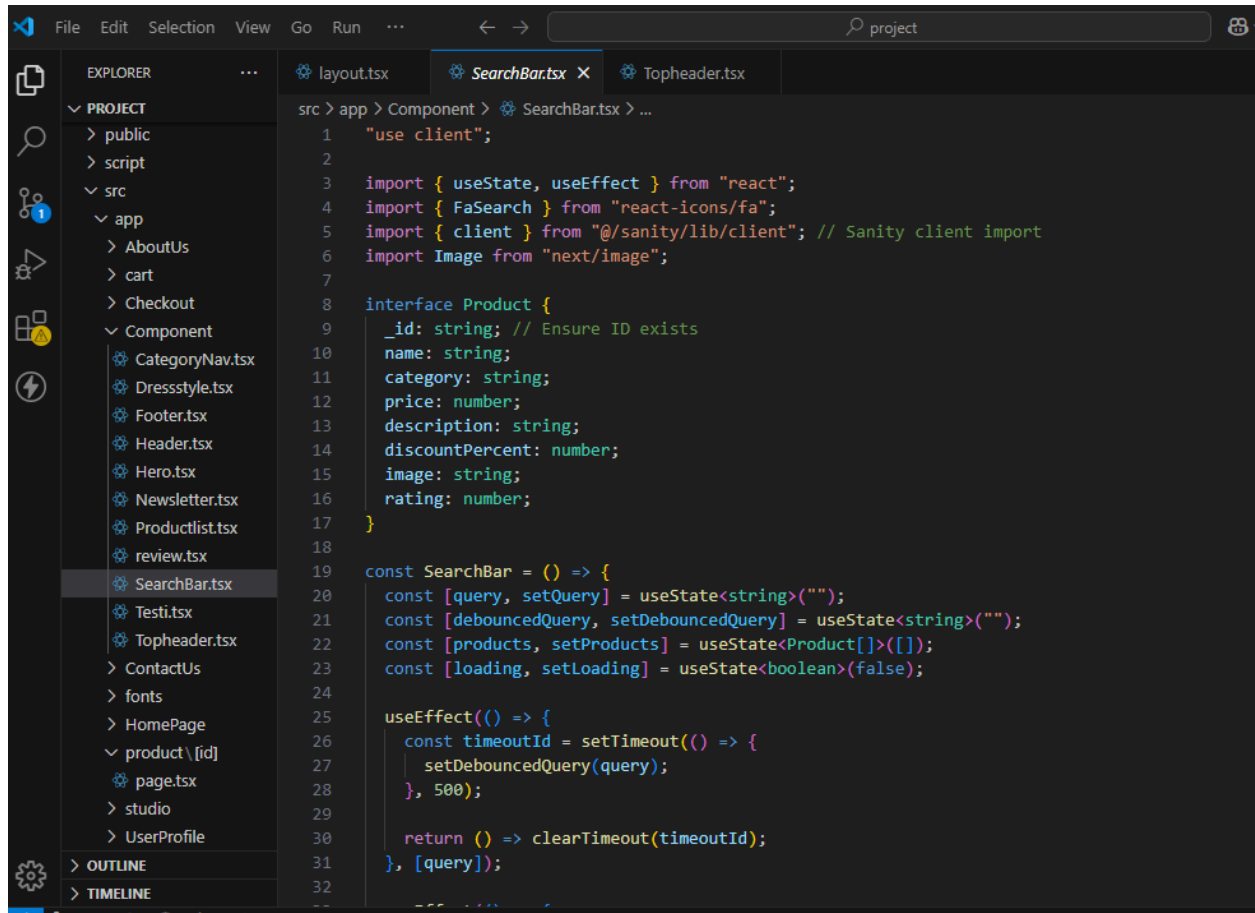
Product detail code



The screenshot shows a code editor with a dark theme. On the left, the Explorer panel displays a file tree for a project. The 'src' directory is expanded, showing 'app' and 'components'. The 'app' directory is further expanded, showing 'category', 'Component', 'ContactUs', 'pagetsx', 'fonts', 'page', 'product\[id]', 'pagetsx', 'studio', 'UserProfile', 'pagetsx', 'favicon.ico', 'globals.css', 'layout.tsx', 'page.tsx', 'searchProduct...', 'components\ui', 'button.tsx', 'card.tsx', 'carousel.tsx', 'lib', 'sanity', and 'lib'. The 'pagetsx' file under 'app' is selected. The main editor area shows the code for 'page.tsx'. The code includes imports for React, Next.js hooks, Sanity client, and helper functions. It defines a TypeScript interface for a product and a React functional component for the product details page. The component uses hooks to fetch product data from a Sanity database and handle routing.

```
src > app > product > [id] > pagetsx > ...
1  'use client';
2
3  import React, { useEffect, useState } from 'react';
4  import { useParams, useRouter } from 'next/navigation'; // Next.js hook to get dynamic params (id) and useRouter
5  import { client } from '@lib/sanity'; // Sanity client for fetching data
6  import { urlFor } from '@lib/sanityImageUtils'; // Helper function to fetch image URL
7  import ReviewsSection from '@app/Component/review'; // Assuming this is a reviews component
8  import NewsletterSubscription from '@app/Component/Newsletter';
9
10 type Product = {
11   id: number; // Product ID as a number
12   name: string;
13   price: number;
14   description: string;
15   discountPercent: number;
16   category: string;
17   image: {
18     asset: {
19       _ref: string;
20     };
21   };
22   rating: number;
23 };
24
25 const ProductDetail: React.FC = () => {
26   const { id } = useParams(); // Get the dynamic product ID from the URL
27   const [product, setProduct] = useState<Product | null>(null); // Store product details
28   const [loading, setLoading] = useState(true); // Loading state for fetching product
29   const router = useRouter(); // Use router to navigate to cart page
30
31   // Fetch the product details when the component mounts
32   useEffect(() => {
```


Search Barcode



The image shows a screenshot of the Visual Studio Code editor interface. The Explorer panel on the left displays a project structure with the following folders and files:

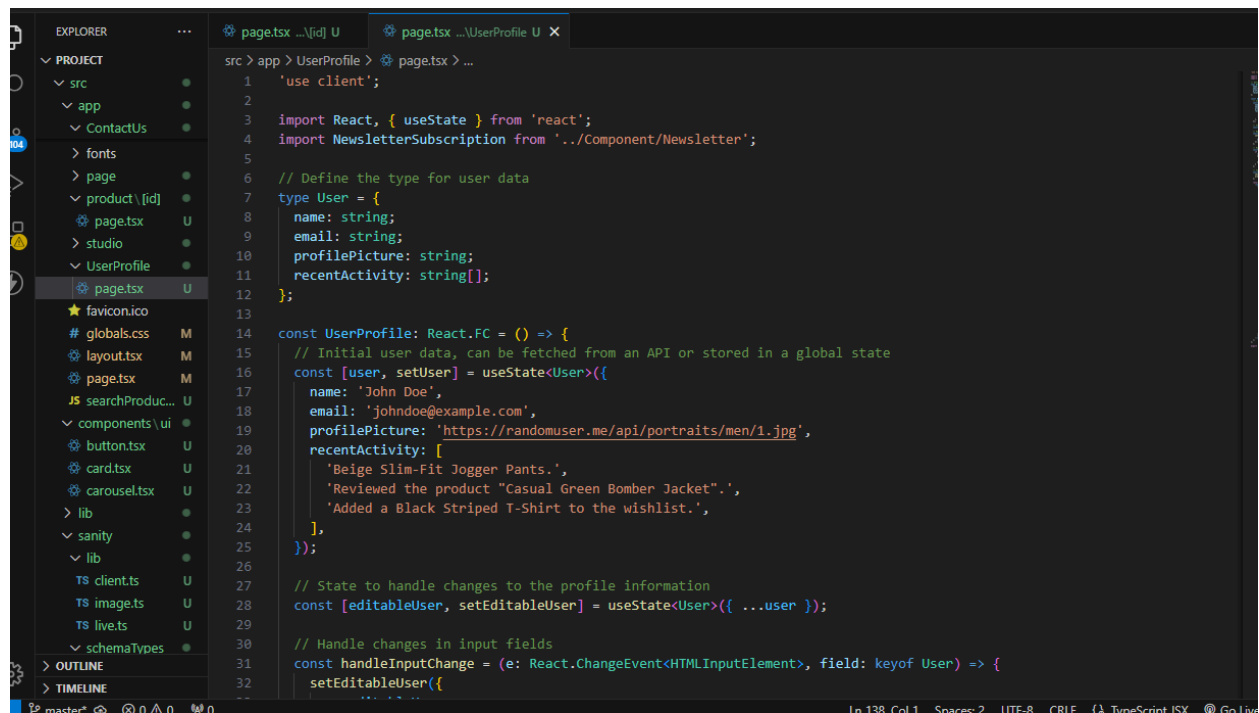
- PROJECT
 - public
 - script
 - src
 - app
 - AboutUs
 - cart
 - Checkout
 - Component
 - CategoryNav.tsx
 - Dressstyle.tsx
 - Footer.tsx
 - Header.tsx
 - Hero.tsx
 - Newsletter.tsx
 - Productlist.tsx
 - review.tsx
 - SearchBar.tsx**
 - Testi.tsx
 - Topheader.tsx
 - ContactUs
 - fonts
 - HomePage
 - product\[id]
 - page.tsx
 - studio
 - UserProfile

- OUTLINE
- TIMELINE

The main editor area shows the code for `SearchBar.tsx`. The code is as follows:

```
1  "use client";
2
3  import { useState, useEffect } from "react";
4  import { FaSearch } from "react-icons/fa";
5  import { client } from "@sanity/lib/client"; // Sanity client import
6  import Image from "next/image";
7
8  interface Product {
9    _id: string; // Ensure ID exists
10   name: string;
11   category: string;
12   price: number;
13   description: string;
14   discountPercent: number;
15   image: string;
16   rating: number;
17 }
18
19 const SearchBar = () => {
20   const [query, setQuery] = useState<string>("");
21   const [debouncedQuery, setDebouncedQuery] = useState<string>("");
22   const [products, setProducts] = useState<Product[]>([]);
23   const [loading, setLoading] = useState<boolean>(false);
24
25   useEffect(() => {
26     const timeoutId = setTimeout(() => {
27       setDebouncedQuery(query);
28     }, 500);
29
30     return () => clearTimeout(timeoutId);
31   }, [query]);
32 }
```

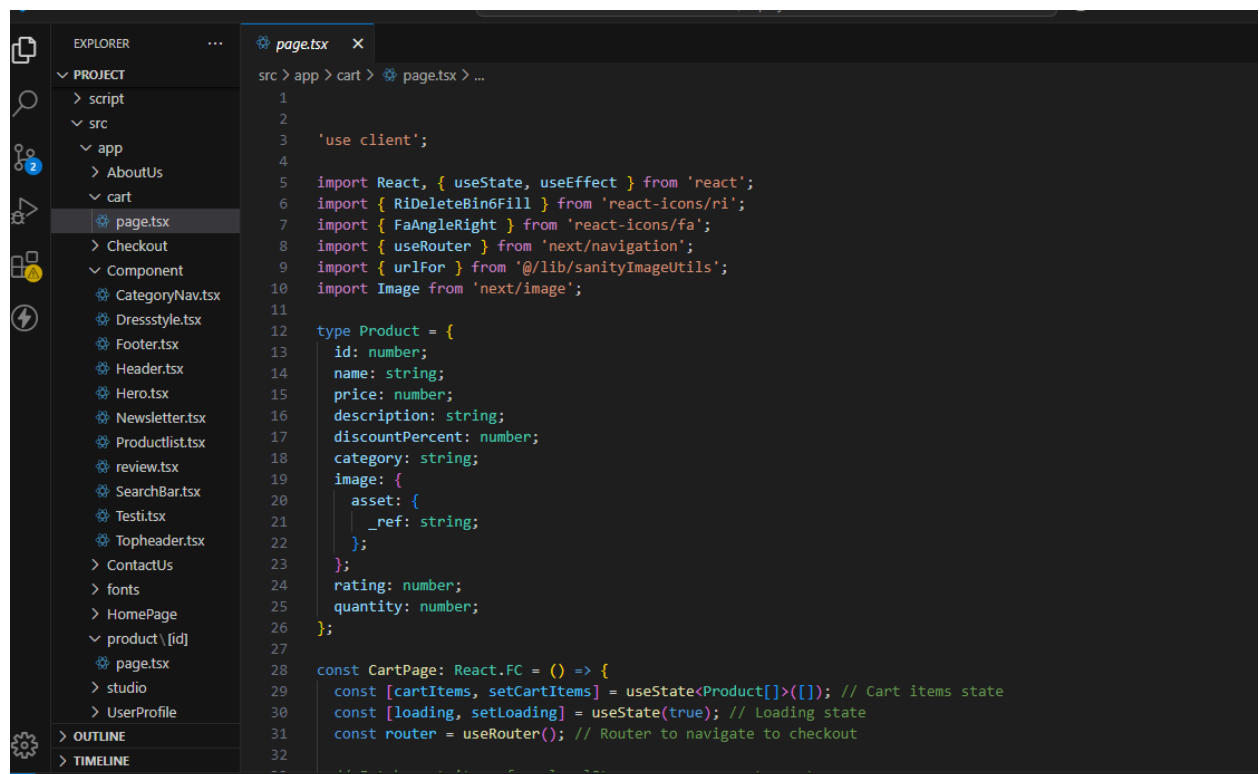
User profile code



The screenshot shows a VS Code editor with the Explorer sidebar on the left. The 'PROJECT' folder is expanded, showing a 'src' directory with an 'app' subdirectory. Inside 'app', there's a 'UserProfile' directory containing a 'page.tsx' file, which is selected. The main editor area shows the code for 'page.tsx'. The code starts with a 'use client' directive, followed by imports for React, useState, and NewsletterSubscription. A 'User' interface is defined with fields: name (string), email (string), profilePicture (string), and recentActivity (string[]). The UserProfile component is a React.FC that uses useState to manage user data. It initializes the user state with a default user object. The component also includes a function to handle input changes, which updates the user state and the editability of the user object.

```
src > app > UserProfile > page.tsx > ...
1  'use client';
2
3  import React, { useState } from 'react';
4  import NewsletterSubscription from '../Component/Newsletter';
5
6  // Define the type for user data
7  type User = {
8    name: string;
9    email: string;
10   profilePicture: string;
11   recentActivity: string[];
12 };
13
14 const UserProfile: React.FC = () => {
15   // Initial user data, can be fetched from an API or stored in a global state
16   const [user, setUser] = useState<User>({
17     name: 'John Doe',
18     email: 'johndoe@example.com',
19     profilePicture: 'https://randomuser.me/api/portraits/men/1.jpg',
20     recentActivity: [
21       'Beige Slim-Fit Jogger Pants.',
22       'Reviewed the product "Casual Green Bomber Jacket".',
23       'Added a Black Striped T-Shirt to the wishlist.',
24     ],
25   });
26
27   // State to handle changes to the profile information
28   const [editableUser, setEditableUser] = useState<User>({ ...user });
29
30   // Handle changes in input fields
31   const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>, field: keyof User) => {
32     setEditableUser({
```

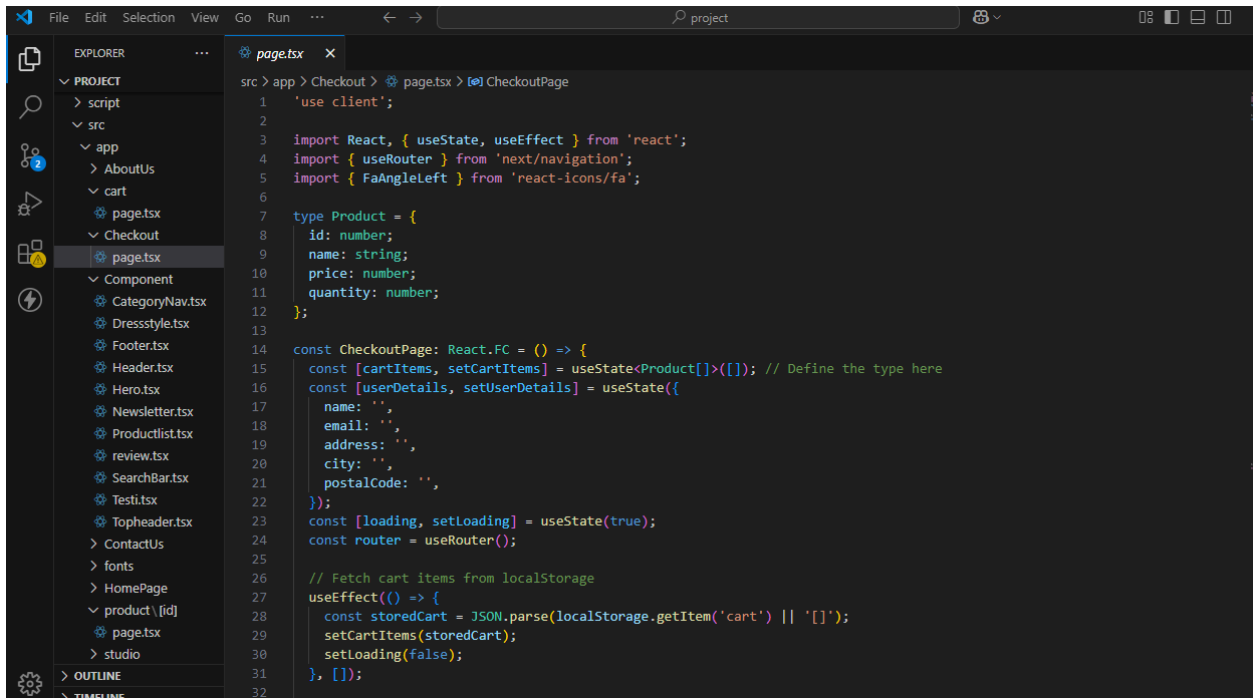
Cart page code



The screenshot shows a VS Code editor with the Explorer sidebar on the left. The 'PROJECT' folder is expanded, showing a 'src' directory with an 'app' subdirectory. Inside 'app', there's a 'cart' directory containing a 'page.tsx' file, which is selected. The main editor area shows the code for 'page.tsx'. The code starts with a 'use client' directive, followed by imports for React, useState, useEffect, RiDeleteBin6Fill, FaAngleRight, useRouter, urlFor, and Image. A 'Product' interface is defined with fields: id (number), name (string), price (number), description (string), discountPercent (number), category (string), image (object with _ref: string), rating (number), and quantity (number). The CartPage component is a React.FC that uses useState to manage cart items, loading state, and a router for navigation.

```
src > app > cart > page.tsx > ...
1
2
3  'use client';
4
5  import React, { useState, useEffect } from 'react';
6  import { RiDeleteBin6Fill } from 'react-icons/ri';
7  import { FaAngleRight } from 'react-icons/fa';
8  import { useRouter } from 'next/navigation';
9  import { urlFor } from '@lib/sanityImageUtils';
10 import Image from 'next/image';
11
12 type Product = {
13   id: number;
14   name: string;
15   price: number;
16   description: string;
17   discountPercent: number;
18   category: string;
19   image: {
20     asset: {
21       _ref: string;
22     };
23   };
24   rating: number;
25   quantity: number;
26 };
27
28 const CartPage: React.FC = () => {
29   const [cartItems, setCartItems] = useState<Product[]>([]); // Cart items state
30   const [loading, setLoading] = useState(true); // Loading state
31   const router = useRouter(); // Router to navigate to checkout
32
```

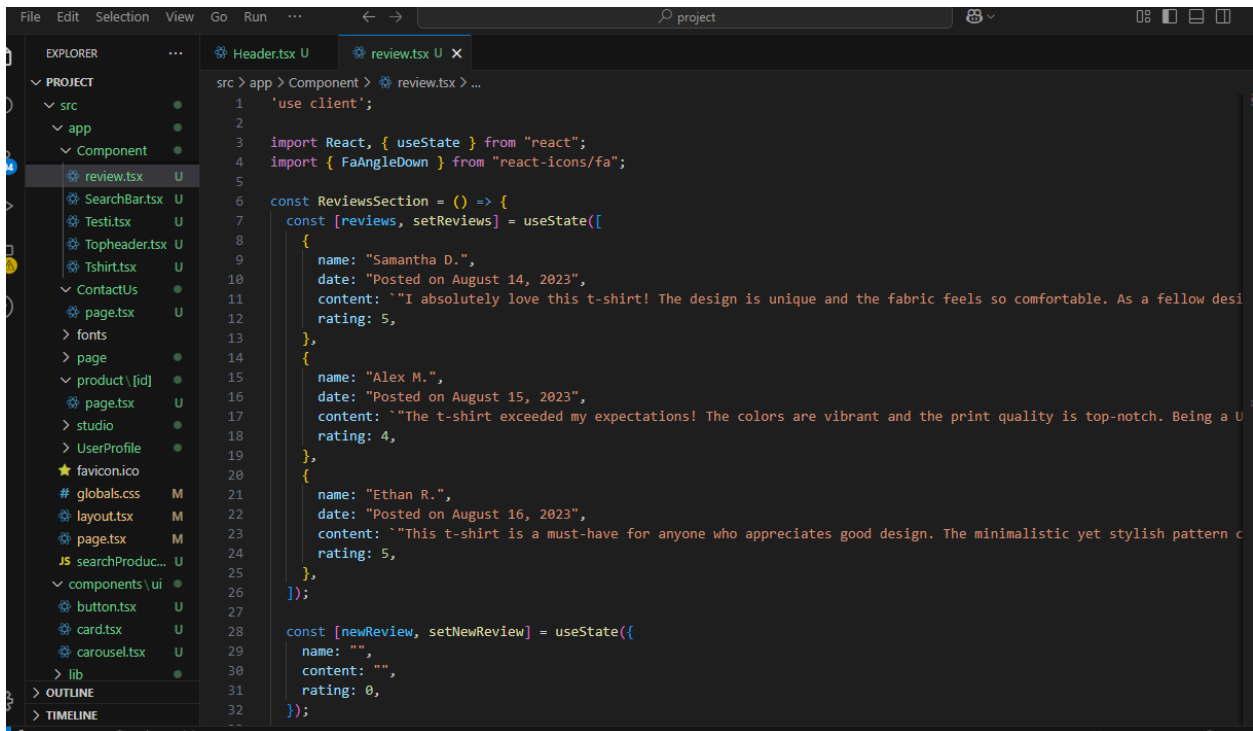
Checkout page code



The screenshot shows the VS Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the following files: script, src, app, AboutUs, cart, Checkout, Component, CategoryNav.tsx, Dressstyle.tsx, Footer.tsx, Header.tsx, Hero.tsx, Newsletter.tsx, Productlist.tsx, review.tsx, SearchBar.tsx, Test.tsx, Topheader.tsx, ContactUs, fonts, HomePage, product\id, page.tsx, studio, OUTLINE, and TIMELINE. The code editor shows the CheckoutPage.tsx file with the following code:

```
src > app > Checkout > CheckoutPage
1  'use client';
2
3  import React, { useState, useEffect } from 'react';
4  import { useRouter } from 'next/navigation';
5  import { FaAngleLeft } from 'react-icons/fa';
6
7  type Product = {
8    id: number;
9    name: string;
10   price: number;
11   quantity: number;
12 };
13
14 const CheckoutPage: React.FC = () => {
15   const [cartItems, setCartItems] = useState<Product[]>([]); // Define the type here
16   const [userDetails, setUserDetails] = useState({
17     name: '',
18     email: '',
19     address: '',
20     city: '',
21     postalCode: '',
22   });
23   const [loading, setLoading] = useState(true);
24   const router = useRouter();
25
26   // Fetch cart items from localStorage
27   useEffect(() => {
28     const storedCart = JSON.parse(localStorage.getItem('cart') || '[]');
29     setCartItems(storedCart);
30     setLoading(false);
31   }, []);
32 }
```

Reviews code



The screenshot shows the VS Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the following files: src, app, Component, review.tsx, SearchBar.tsx, Test.tsx, Topheader.tsx, Tshirt.tsx, ContactUs, page.tsx, fonts, page, product\id, page.tsx, studio, UserProfile, favicon.ico, # globals.css, layout.tsx, page.tsx, JS searchProduc..., components\ui, button.tsx, card.tsx, carousel.tsx, lib, OUTLINE, and TIMELINE. The code editor shows the review.tsx file with the following code:

```
src > app > Component > review.tsx
1  'use client';
2
3  import React, { useState } from 'react';
4  import { FaAngleDown } from 'react-icons/fa';
5
6  const ReviewsSection = () => {
7    const [reviews, setReviews] = useState([
8      {
9        name: "Samantha D.",
10       date: "Posted on August 14, 2023",
11       content: "I absolutely love this t-shirt! The design is unique and the fabric feels so comfortable. As a fellow desi
12       rating: 5,
13     },
14     {
15       name: "Alex M.",
16       date: "Posted on August 15, 2023",
17       content: "The t-shirt exceeded my expectations! The colors are vibrant and the print quality is top-notch. Being a U
18       rating: 4,
19     },
20     {
21       name: "Ethan R.",
22       date: "Posted on August 16, 2023",
23       content: "This t-shirt is a must-have for anyone who appreciates good design. The minimalistic yet stylish pattern c
24       rating: 5,
25     },
26   ]);
27
28   const [newReview, setNewReview] = useState({
29     name: "",
30     content: "",
31     rating: 0,
32   });
33 }
```

Documentation

Technical Report

1. Introduction

This document provides a technical summary of the development of dynamic frontend components for our marketplace application. It details the steps taken, challenges encountered, and best practices followed during the implementation of product listing, filtering, and dynamic routing.

2. Steps Taken

Frontend Components Implementation:

1. **Product Listing Page:**
 - Created a **ProductList** component to fetch and display products dynamically.
 - Implemented pagination for better user experience.
 2. **Product Detail Page:**
 - Developed a **ProductDetail** component that renders product-specific data based on dynamic routing.
 - Used React Router for navigation and parameterized routes.
 - Integrated a **Review Component** where users can submit reviews for products.
 3. **Search & Filters:**
 - Implemented a **SearchBar** component for real-time product searching.
 - Fixed search-related issues to ensure precise search functionality.
 - Added category filters to dynamically update the product listing based on user selections.
 4. **User Authentication:**
 - Implemented a **Login/Signup Component** using Clerk for secure user authentication.
 5. **Cart & Checkout:**
 - Created a **CartComponent** with an **Add to Cart** button, allowing users to add products to the cart.
 - Implemented navigation to the cart page upon clicking **Add to Cart**.
 - Added a **Checkout Button** in the cart component that directs users to the checkout page to input their information and complete the purchase.
-

3. Challenges Faced & Solutions Implemented

Challenge	Solution Implemented
API response delay affecting UI	Implemented loading states and skeleton loaders for smooth UX
Search and filters causing performance issues	Used debouncing and optimized API queries
Handling dynamic routes effectively	Utilized React Router's <code>useParams</code> for accurate routing
Managing global state across components	Implemented Context API for efficient state management
Search bar not functioning properly	Debugged the search component and ensured proper query execution

4. Best Practices Followed

- **Component-Based Architecture:** Ensured modularity and reusability by designing separate components for different functionalities.
 - **Optimized API Calls:** Reduced redundant API calls using caching techniques.
 - **Code Readability & Documentation:** Followed proper naming conventions and added comments for clarity.
 - **User Experience Enhancements:** Added smooth animations and responsive design for mobile-friendliness.
 - **Secure Authentication:** Used Clerk for user login/signup to enhance security and ease of access.
-

5. Conclusion

This project successfully implemented dynamic frontend components for the marketplace, incorporating product listing, filtering, search functionality, dynamic routing, user authentication, cart functionality, and checkout. By overcoming key challenges and following best practices, we ensured a scalable and maintainable application structure.
