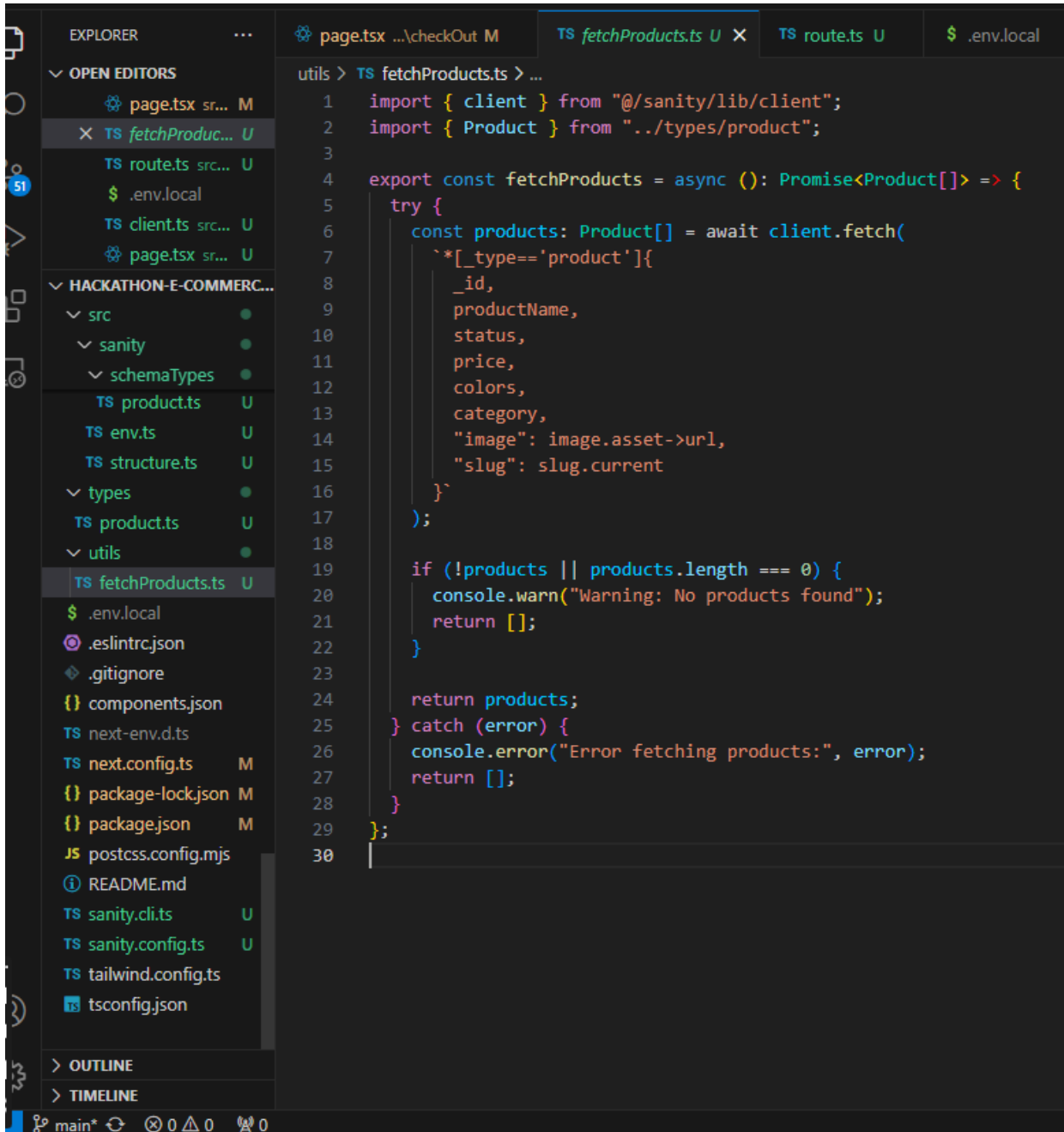


## Building Dynamic Frontend Components

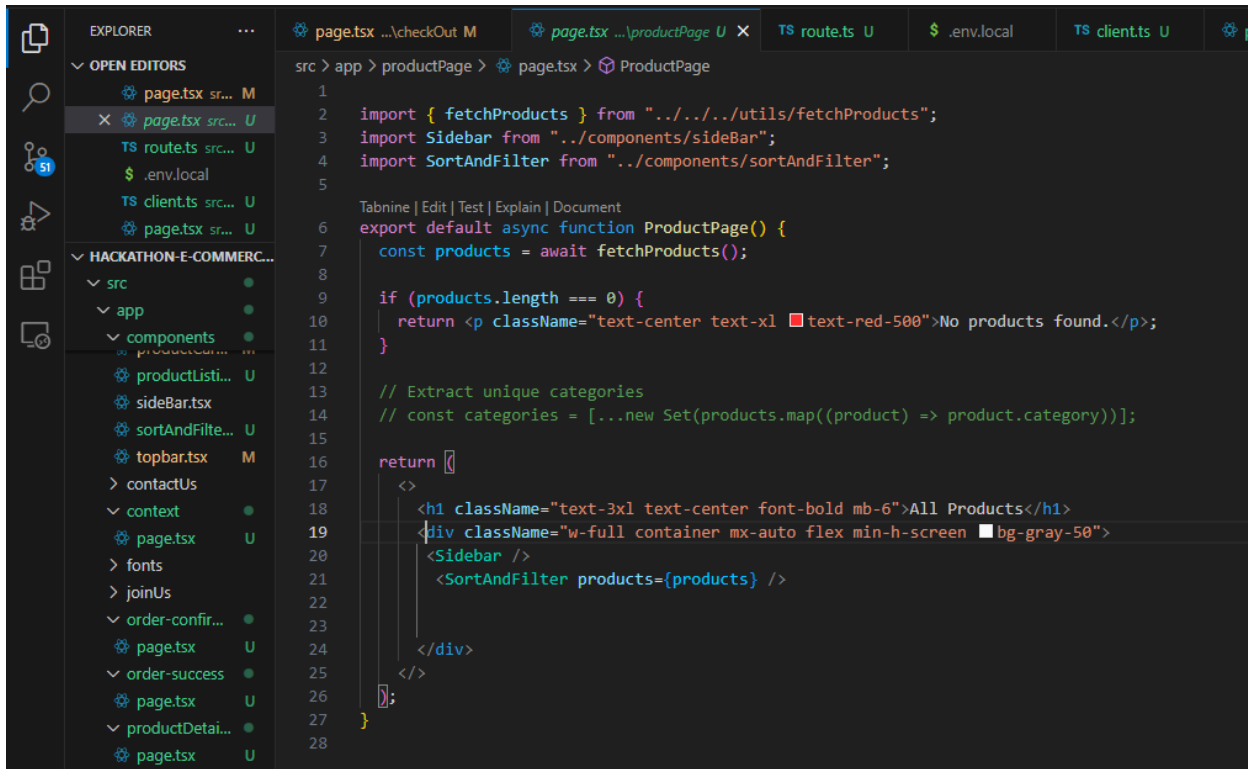
### Code Deliverables:

### Product fetching function:



```
1  import { client } from "@sanity/lib/client";
2  import { Product } from "../types/product";
3
4  export const fetchProducts = async (): Promise<Product[]> => {
5    try {
6      const products: Product[] = await client.fetch(
7        `*[_type=='product']{
8          _id,
9          productName,
10         status,
11         price,
12         colors,
13         category,
14         "image": image.asset->url,
15         "slug": slug.current
16       }`
17      );
18
19      if (!products || products.length === 0) {
20        console.warn("Warning: No products found");
21        return [];
22      }
23
24      return products;
25    } catch (error) {
26      console.error("Error fetching products:", error);
27      return [];
28    }
29  };
30
```

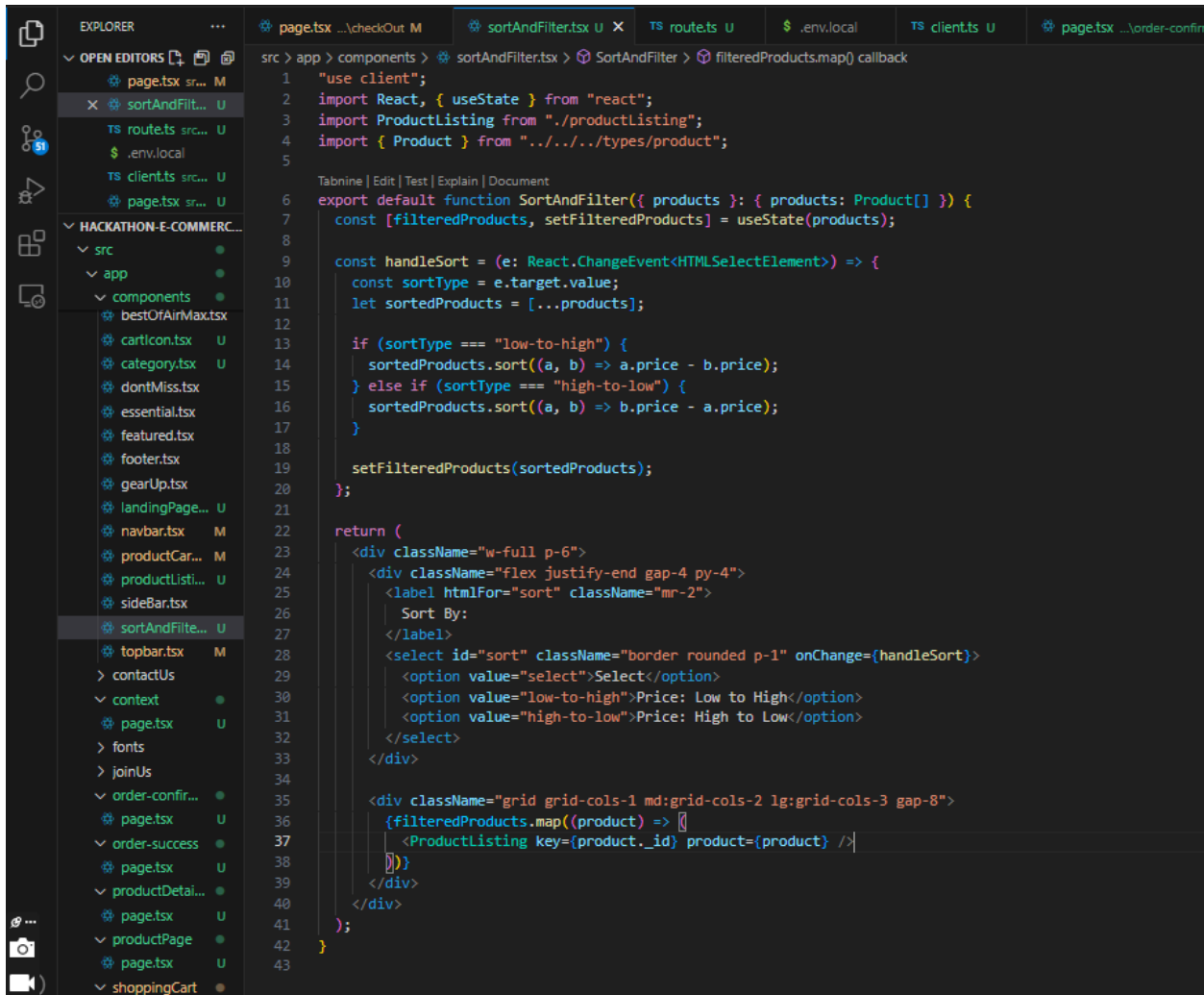
# Product Page:



The screenshot shows a VS Code editor with a dark theme. The Explorer sidebar on the left displays a project structure for 'HACKATHON-E-COMMERC...'. The 'src' directory contains 'app' and 'components'. The 'app' directory contains 'productPage.tsx', 'productList.tsx', 'sideBar.tsx', 'sortAndFilter.tsx', and 'topbar.tsx'. The 'components' directory contains 'contactUs', 'context', 'fonts', 'joinUs', 'order-confirm', 'order-success', 'productDetail', and 'productPage'. The 'productPage.tsx' file is selected and open in the editor. The editor shows the following code:

```
1
2 import { fetchProducts } from "../../utils/fetchProducts";
3 import Sidebar from "../components/sideBar";
4 import SortAndFilter from "../components/sortAndFilter";
5
6 export default async function ProductPage() {
7   const products = await fetchProducts();
8
9   if (products.length === 0) {
10    return <p className="text-center text-xl text-red-500">No products found.</p>;
11  }
12
13  // Extract unique categories
14  // const categories = [...new Set(products.map((product) => product.category))];
15
16  return (
17    <>
18      <h1 className="text-3xl text-center font-bold mb-6">All Products</h1>
19      <div className="w-full container mx-auto flex min-h-screen bg-gray-50">
20        <Sidebar />
21        <SortAndFilter products={products} />
22      </div>
23    </>
24  );
25
26 }
27
28
```

# Sort and Filter:



```
1 "use client";
2 import React, { useState } from "react";
3 import ProductListing from "../productListing";
4 import { Product } from "../../types/product";
5
6 export default function SortAndFilter({ products }: { products: Product[] }) {
7   const [filteredProducts, setFilteredProducts] = useState(products);
8
9   const handleSort = (e: React.ChangeEvent<HTMLSelectElement>) => {
10     const sortType = e.target.value;
11     let sortedProducts = [...products];
12
13     if (sortType === "low-to-high") {
14       sortedProducts.sort((a, b) => a.price - b.price);
15     } else if (sortType === "high-to-low") {
16       sortedProducts.sort((a, b) => b.price - a.price);
17     }
18
19     setFilteredProducts(sortedProducts);
20   };
21
22   return (
23     <div className="w-full p-6">
24       <div className="flex justify-end gap-4 py-4">
25         <label htmlFor="sort" className="mr-2">
26           Sort By:
27         </label>
28         <select id="sort" className="border rounded p-1" onChange={handleSort}>
29           <option value="select">Select</option>
30           <option value="low-to-high">Price: Low to High</option>
31           <option value="high-to-low">Price: High to Low</option>
32         </select>
33       </div>
34
35       <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-8">
36         {filteredProducts.map((product) => [
37           <ProductListing key={product._id} product={product} />
38         ])}
39       </div>
40     </div>
41   );
42 }
```

# Product Detail Page (Dynamic route):

```
src > app > productDetails > [slug] > page.tsx > ProductDetails
1  import React from "react";
2  import { client } from "@sanity/lib/client";
3  import Image from "next/image";
4  import AddToCart from "@app/components/addToCart";
5
6  Tabnine | Edit | Test | Explain | Document
7  async function ProductDetails({ params }: { params: { slug: string } }) {
8    const product = await client.fetch(
9      `[_type == "product" && slug.current == ${slug}[0]{
10        _id, productName, description, status, price, colors, category, quantity, "image": image.asset->url
11      }`,
12      { slug: params.slug }
13    );
14    if (!product) {
15      return <div className="text-center mt-10">Product not found.</div>;
16    }
17    return (
18      <div className="max-w-5xl mx-auto p-6 bg-white rounded-lg mt-20">
19        <div className="flex flex-col md:flex-row gap-8">
20          <div className="relative w-full md:w-1/2 aspect-square bg-white rounded-lg">
21            <Image
22              width={300}
23              height={300}
24              src={product.image}
25              alt={product.productName}
26              className="object-fill object-center overflow-hidden rounded-lg"
27            />
28          </div>
29
30          <div>
31            <h2 className="text-2xl font-bold text-gray-900">
32              {product.productName}
33            </h2>
34            <p className="text-gray-600 leading-relaxed">{product.description}</p>
35            <p className="text-2xl font-semibold">{product.price}</p>
36            <AddToCart product={product} />
37          </div>
38        </div>
39      </div>
40    );
41  }
42  export default ProductDetails;
43
44
45
```

## Scripts:

scripts > JS importSanityData.mjs > ...

```

1  |import { createClient } from '@sanity/client';
2  |import axios from 'axios';
3  |import dotenv from 'dotenv';
4  |import { fileURLToPath } from 'url';
5  |import path from 'path';
6
7  |// Load environment variables from .env.local
8  |const __filename = fileURLToPath(import.meta.url);
9  |const __dirname = path.dirname(__filename);
10 |dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12 |// Create Sanity client
13 |const client = createClient({
14 |  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15 |  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16 |  useCdn: false,
17 |  token: process.env.SANITY_API_TOKEN,
18 |  apiVersion: '2021-08-31'
19 |});
20
21
22 |Tabnine | Edit | Test | Explain | Document
23 |async function uploadImageToSanity(imageUrl) {
24 |  try {
25 |    console.log(`Uploading image: ${imageUrl}`);
26 |    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
27 |    const buffer = Buffer.from(response.data);
28 |    const asset = await client.assets.upload('image', buffer, {
29 |      filename: imageUrl.split('/').pop()
30 |    });
31 |    console.log(`Image uploaded successfully: ${asset._id}`);
32 |    return asset._id;
33 |  } catch (error) {
34 |    console.error('Error uploading image:', error);
35 |    return null;
36 |  }
37 |}
```

Acti  
Go to

scripts &gt; JS importSanityData.mjs &gt; uploadImageToSanity &gt; asset

```
22 async function uploadImageToSanity(imageUrl) {
23   try {
24     console.log(`Uploading image: ${imageUrl}`);
25     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
26     const buffer = Buffer.from(response.data);
27     const asset = await client.assets.upload('image', buffer, {
28       filename: imageUrl.split('/').pop()
29     });
30     console.log(`Image uploaded successfully: ${asset._id}`);
31     return asset._id;
32   } catch (error) {
33     console.error('Failed to upload image:', imageUrl, error);
34     return null;
35   }
36 }
37
```

Tabnine | Edit | Test | Explain | Document

```
38 async function importData() {
39   try {
40     console.log('migrating data please wait...');
41
42     // API endpoint containing car data
43     const response = await axios.get('https://template-03-api.vercel.app/api/products');
44     const products = response.data.data;
45     console.log("products ==>> ", products);
46
47
48     for (const product of products) {
49       let imageRef = null;
50       if (product.image) {
51         imageRef = await uploadImageToSanity(product.image);
52       }
53     }
54   }
55 }
```

```
...  page.tsx ...\checkOut M  sortAndFilter.tsx U  page.tsx ...\slug U  JS importSanityData.mjs U X  productCard.tsx M  TS route.ts M

scripts > JS importSanityData.mjs > importData > sanityProduct > image
M 38 async function importData() {
U 49     let imageRef = null;
U 50     if (product.image) {
U 51         imageRef = await uploadImageToSanity(product.image);
M 52     }
U 53
U 54     const sanityProduct = {
U 55         _type: 'product',
U 56         productName: product.productName,
U 57         category: product.category,
MERC... 58         price: product.price,
        59         inventory: product.inventory,
        60         colors: product.colors || [], // Optional, as per your schema
        61         status: product.status,
        62         description: product.description,
        63         image: imageRef ? {
        64             _type: 'image',
        65             asset: {
        66                 _type: 'reference',
        67                 _ref: imageRef,
        68             },
        69         } : undefined,
        70     };
U 71
        72     await client.create(sanityProduct);
U 73     }
        74
        75     console.log('Data migrated successfully!');
        76     } catch (error) {
U 77         console.error('Error in migrating data ==>> ', error);
U 78     }
        79 }
M 80
U 81 importData();
M
```

\