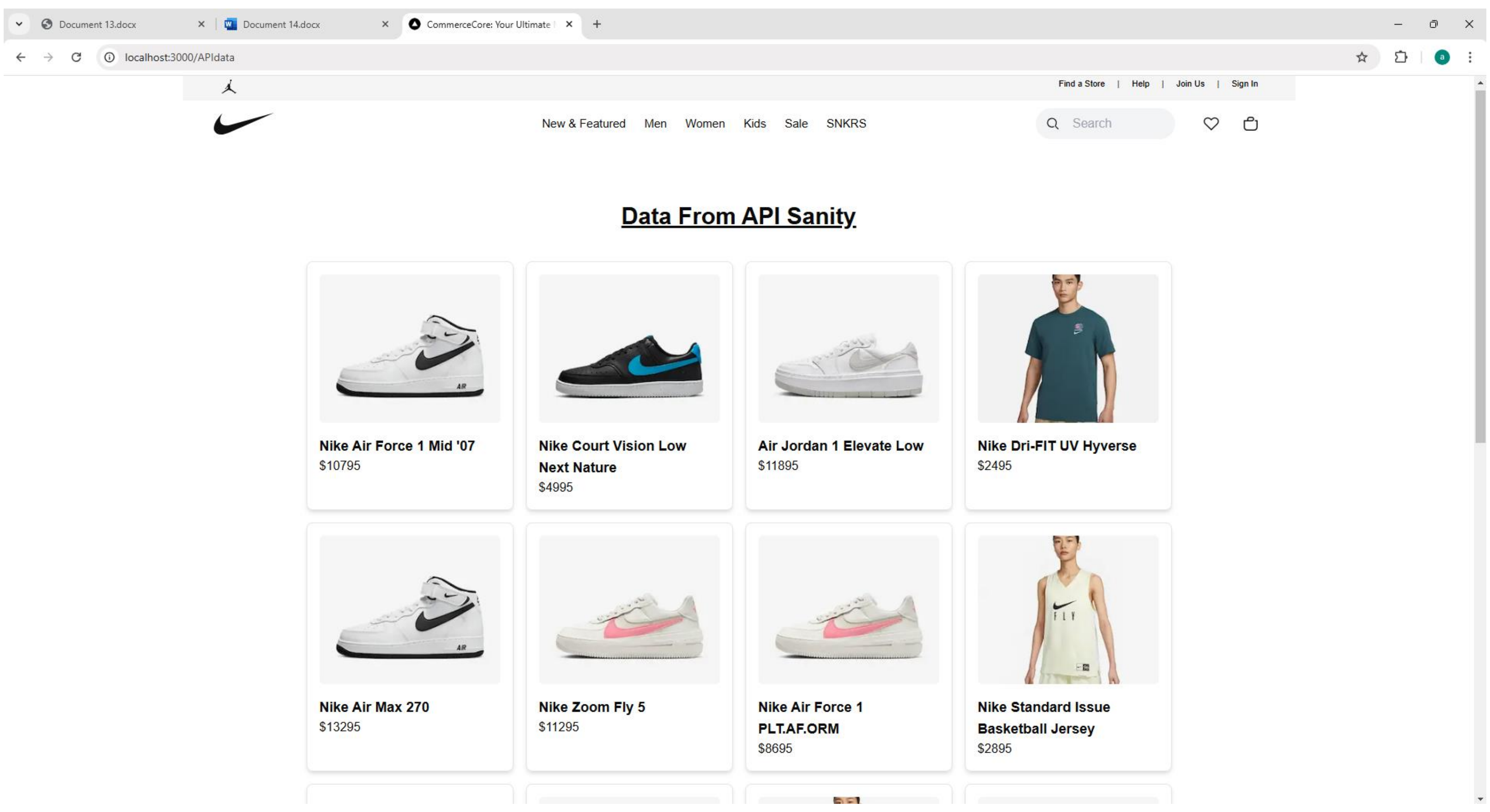
DAY 4 - BUILDING DYNAMIC FRONTEND COMPONENTS FOR YOUR MARKETPLACE

Objective:

On Day 4, I will have to focus on designing and developing dynamic frontend components to display marketplace data fetched from Sanity CMS or APIs. This step emphasizes modular, reusable component design and real-world practices for building scalable and responsive web applications.

1. Functional Deliverables:

The Product Listing Page with Dynamic Data



Data From API Sanity

Nike Air Force 1 Mid '07
$10795

Nike Court Vision Low Next Nature
$4995

Air Jordan 1 Elevate Low
$11895

Nike Dri-FIT UV Hyverse
$2495

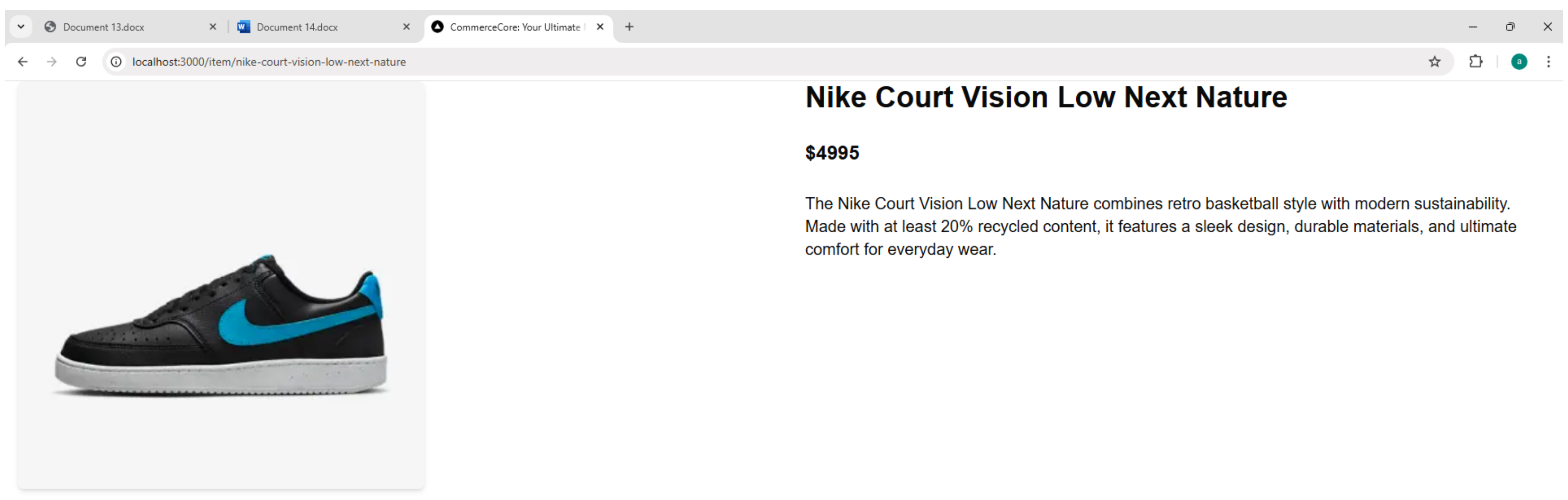
Nike Air Max 270
$13295

Nike Zoom Fly 5
$11295

Nike Air Force 1 PLT.AF.ORM
$8695

Nike Standard Issue Basketball Jersey
$2895

Individual product detail pages with accurate routing and data rendering.

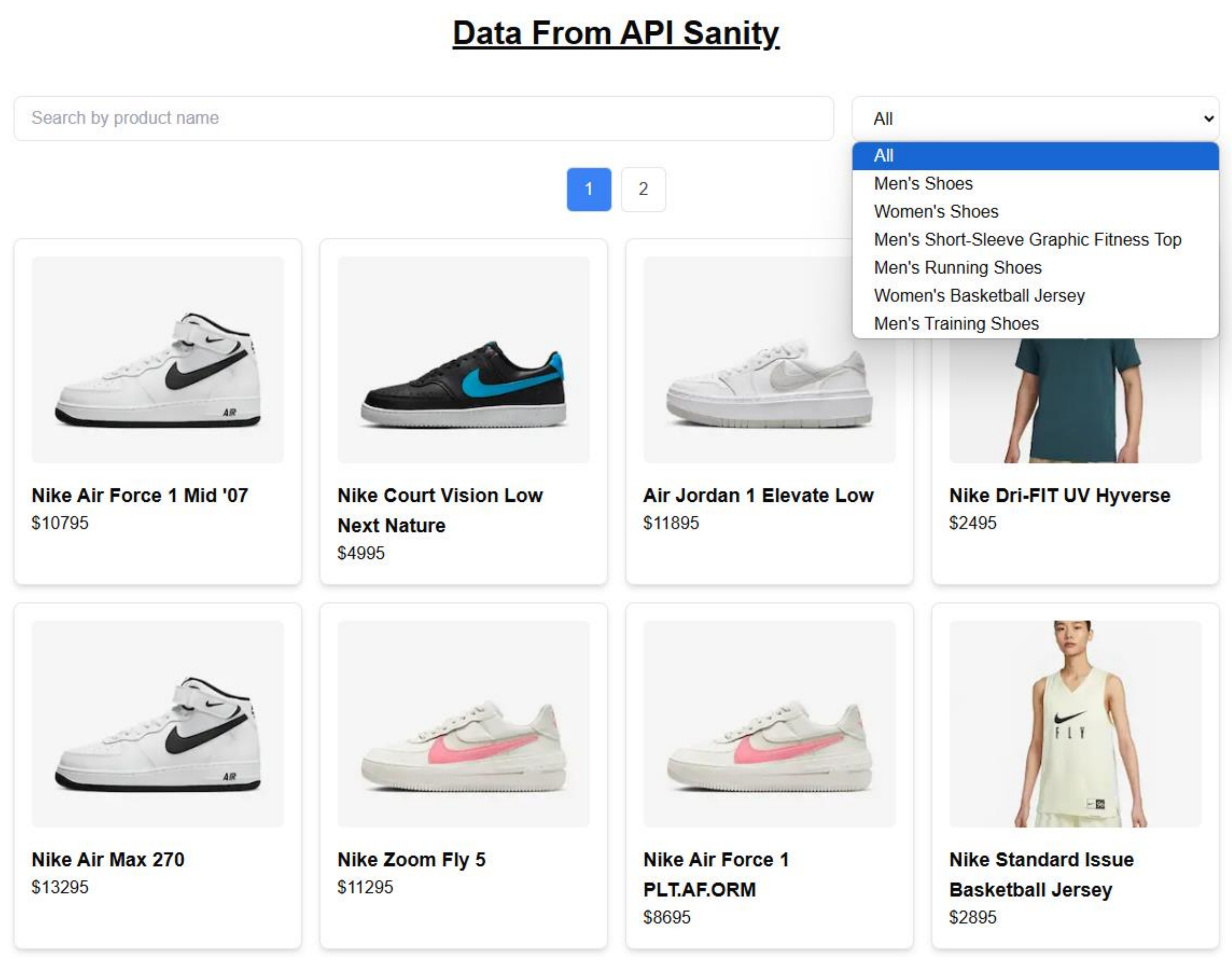

Document 13.docx
Document 14.docx
CommerceCore: Your Ultimate
localhost:3000/item/nike-court-vision-low-next-nature

Nike Court Vision Low Next Nature

$4995

The Nike Court Vision Low Next Nature combines retro basketball style with modern sustainability. Made with at least 20% recycled content, it features a sleek design, durable materials, and ultimate comfort for everyday wear.

category filters, search bar, and pagination.



Data From API Sanity

Search by product name

All

- All
- Men's Shoes
- Women's Shoes
- Men's Short-Sleeve Graphic Fitness Top
- Men's Running Shoes
- Women's Basketball Jersey
- Men's Training Shoes

1 2

Nike Air Force 1 Mid '07
$10795

Nike Court Vision Low Next Nature
$4995

Air Jordan 1 Elevate Low
$11895

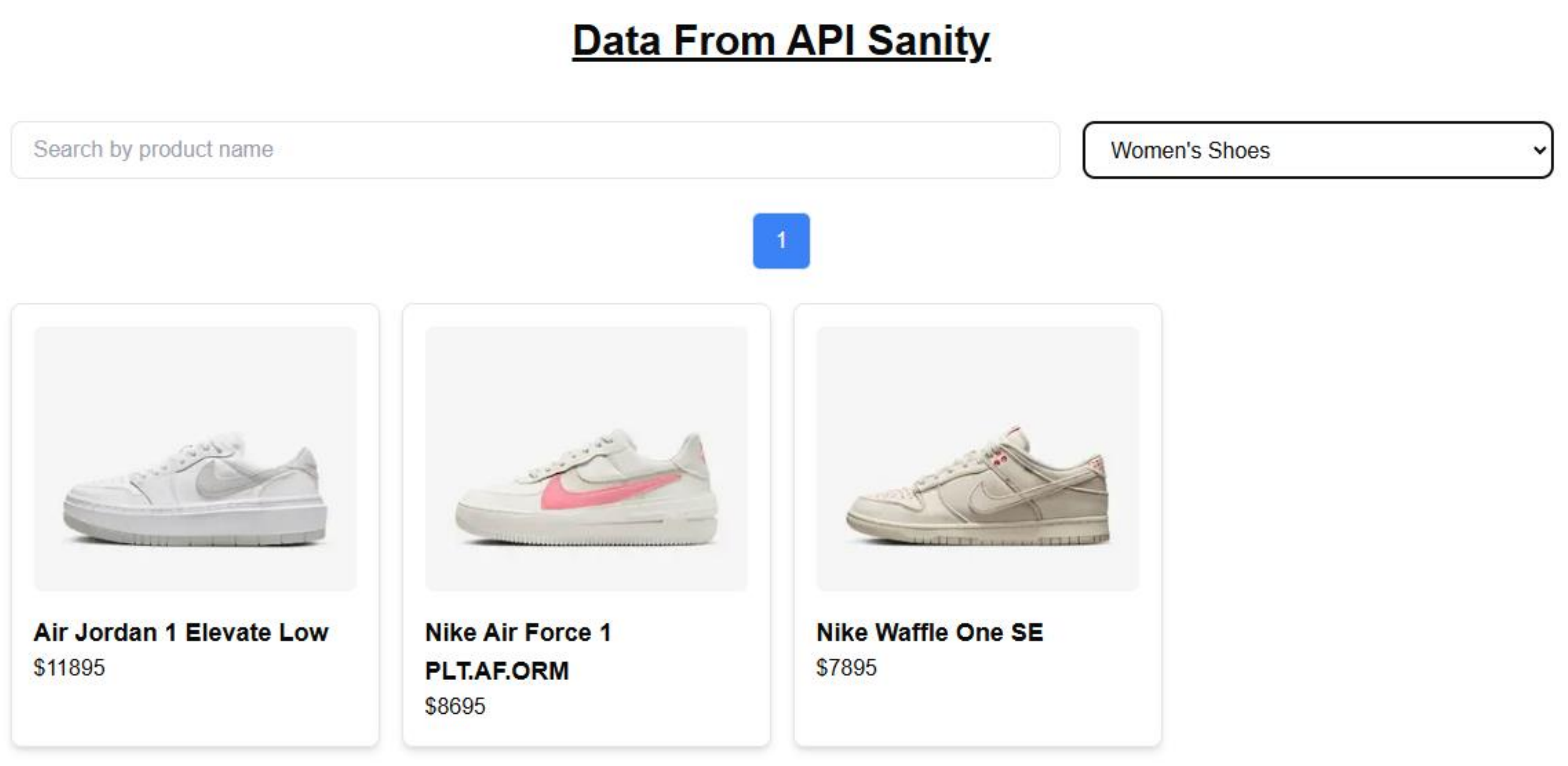
Nike Dri-FIT UV Hyverse
$2495

Nike Air Max 270
$13295

Nike Zoom Fly 5
$11295

Nike Air Force 1 PLT.AF.ORM
$8695

Nike Standard Issue Basketball Jersey
$2895



Data From API Sanity

Search by product name

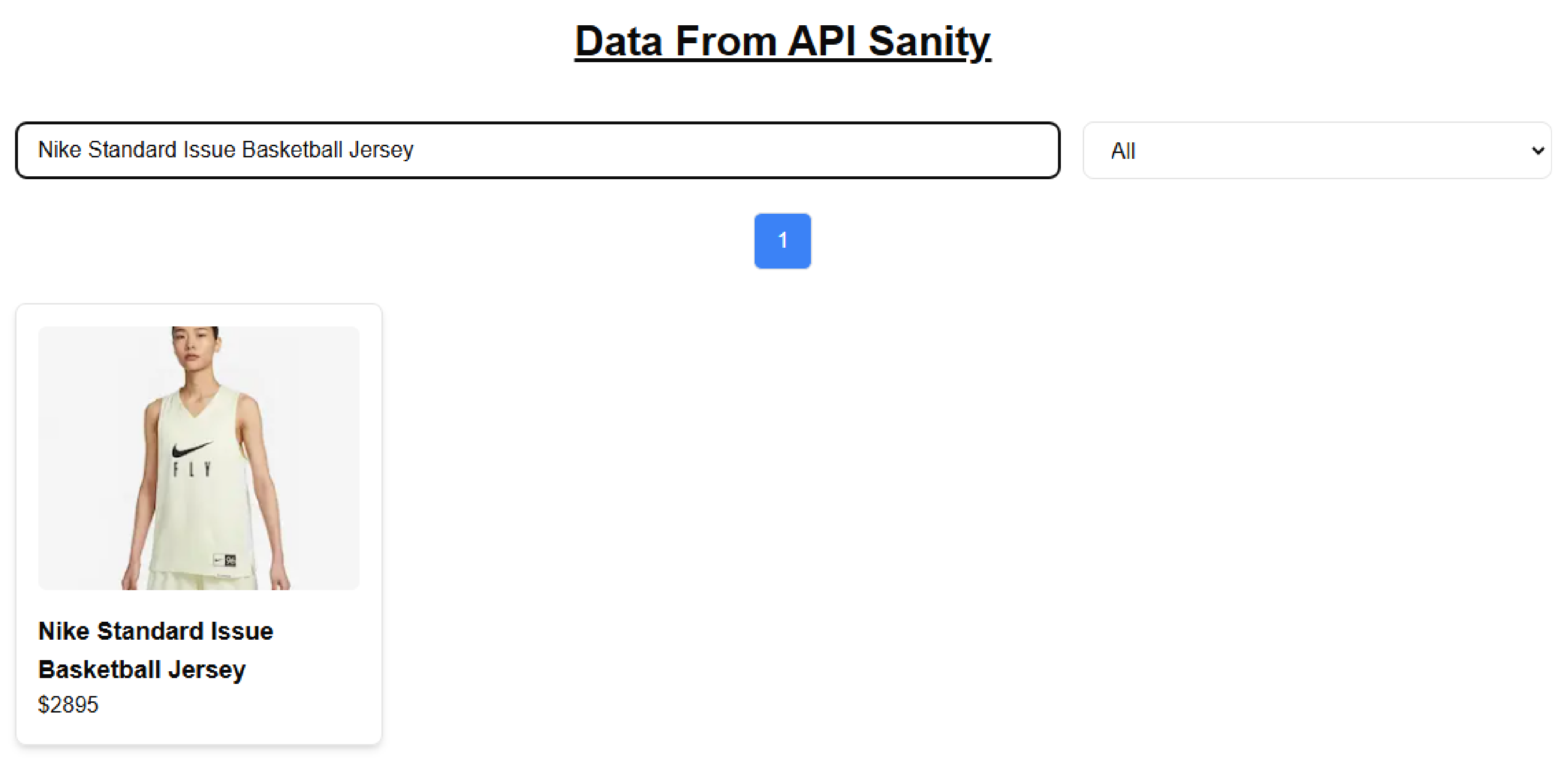
Women's Shoes

1

Air Jordan 1 Elevate Low
$11895

Nike Air Force 1 PLT.AF.ORM
$8695

Nike Waffle One SE
$7895



Data From API Sanity

Nike Standard Issue Basketball Jersey

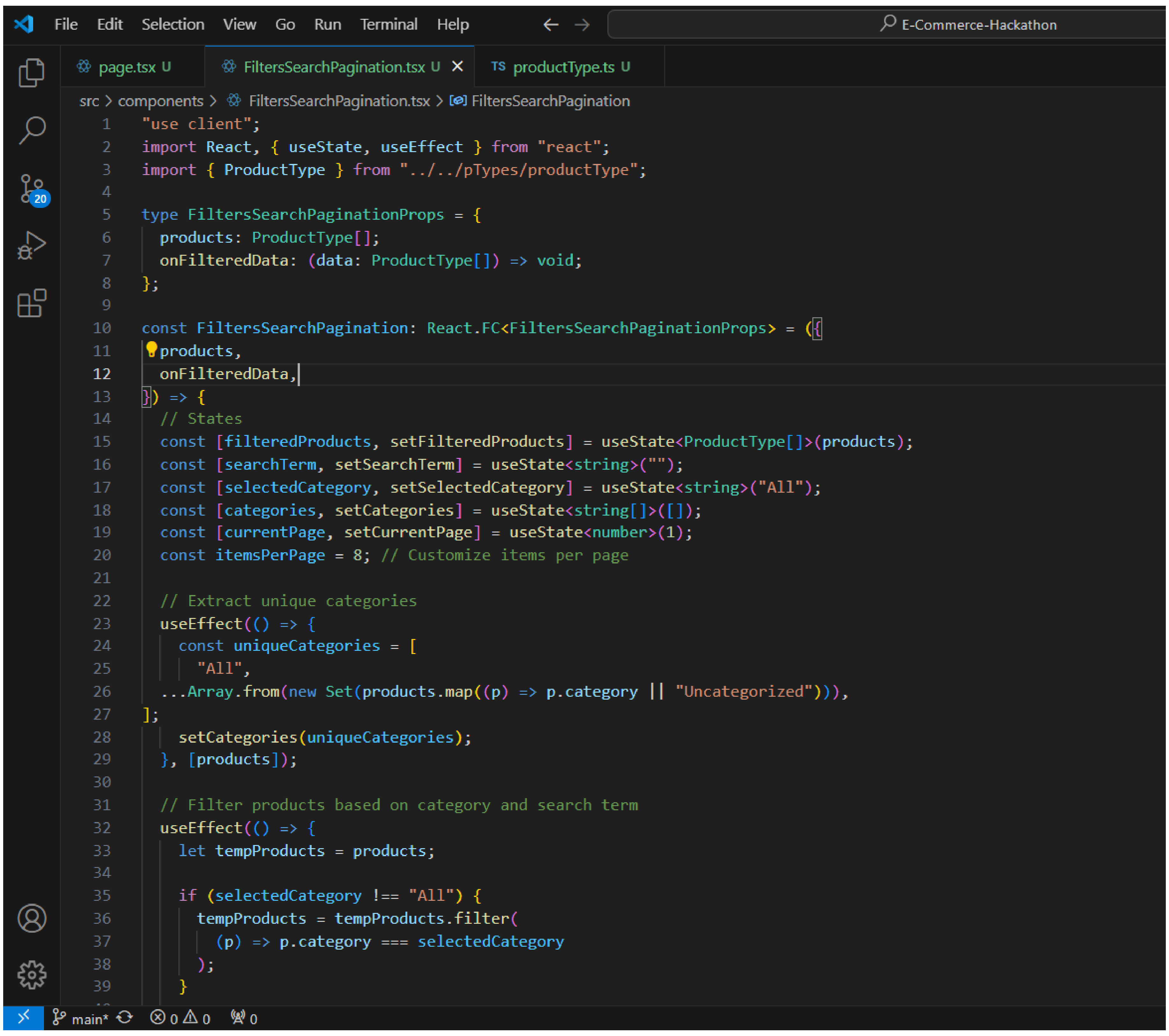
All

1

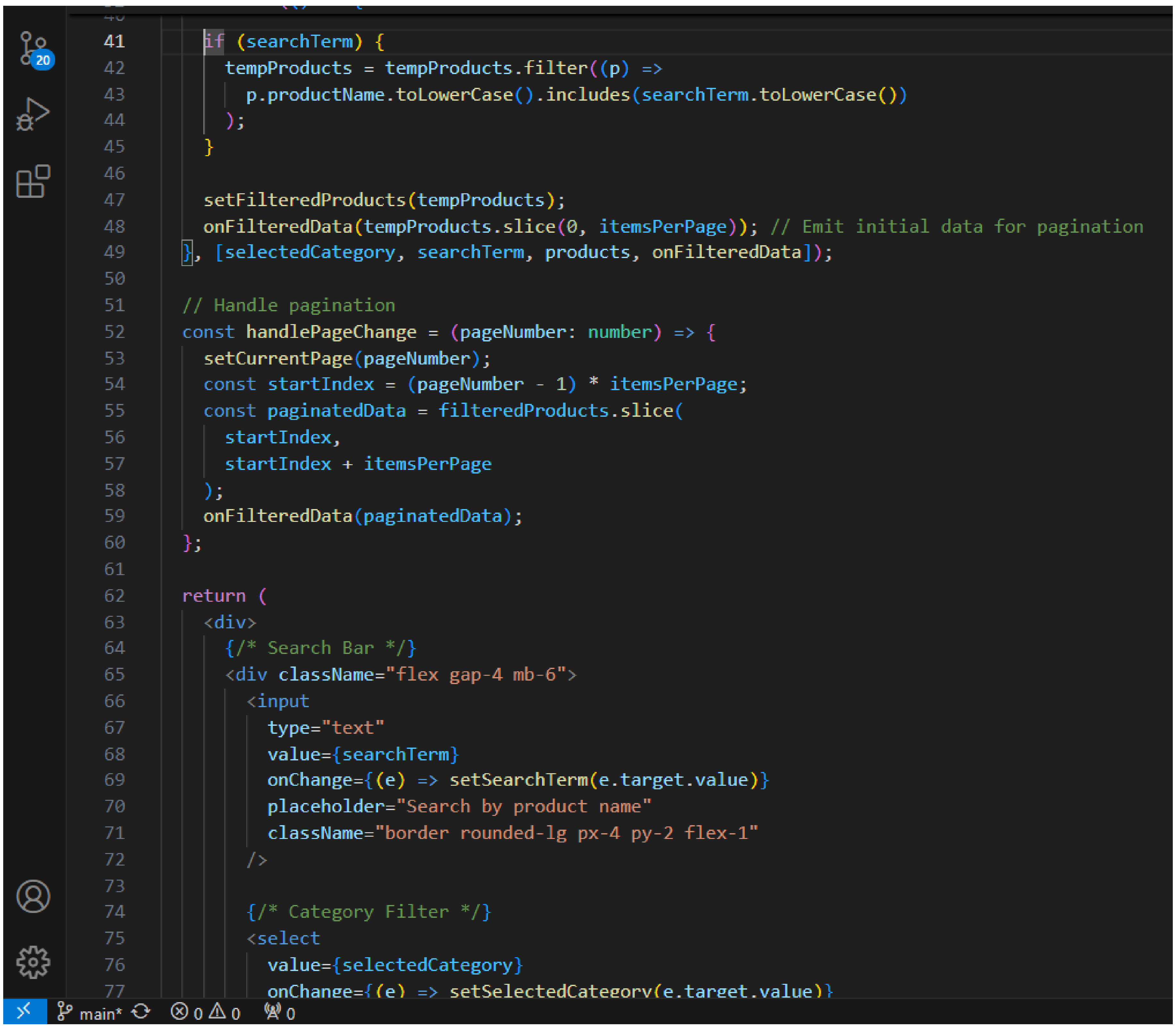
Nike Standard Issue Basketball Jersey
$2895

2. Code Deliverables:

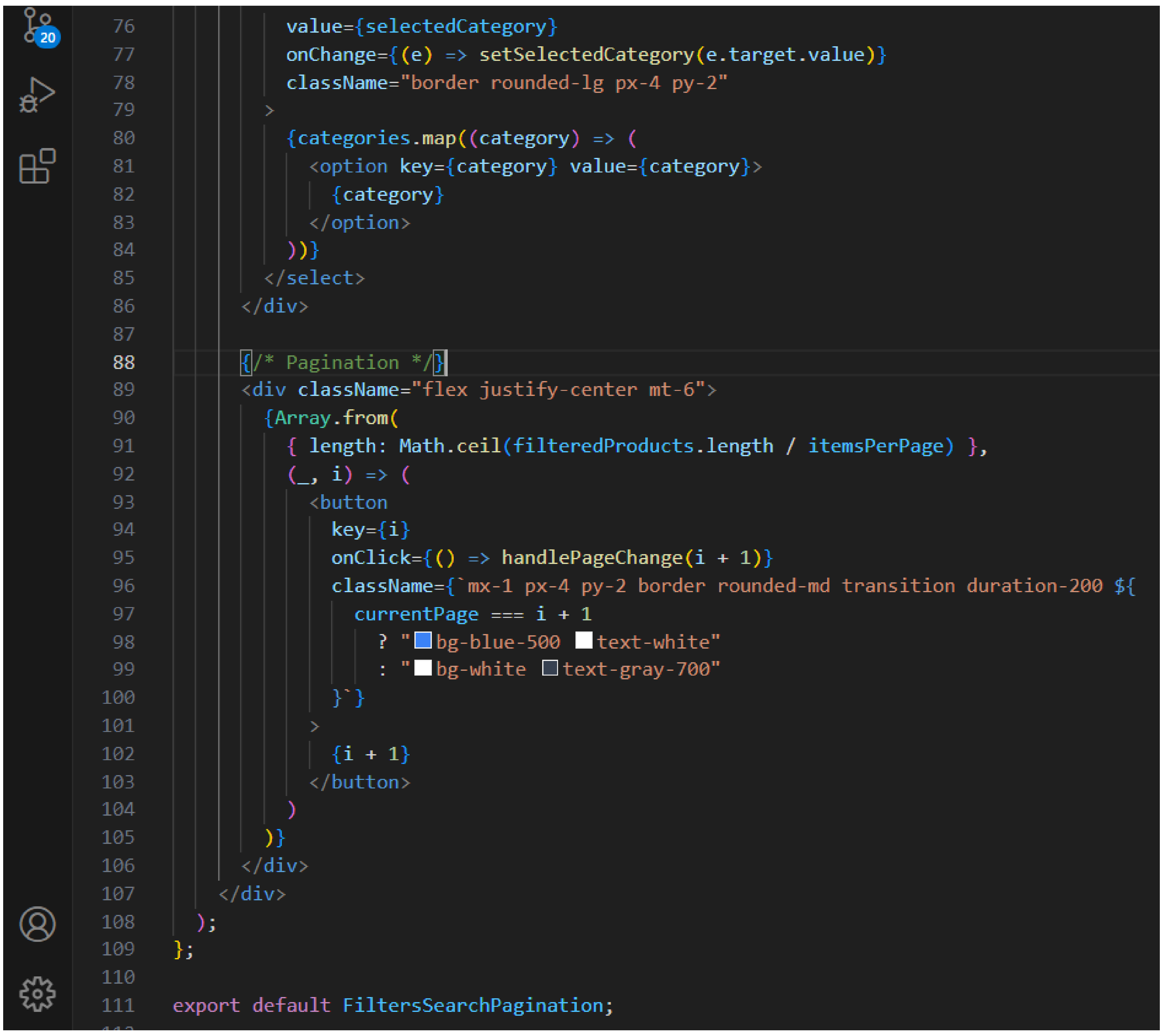
Code snippets for key component ‘FiltersSearchPagination’


File Edit Selection View Go Run Terminal Help
E-Commerce-Hackathon
page.tsx U
FiltersSearchPagination.tsx U
productType.ts U
src > components > FiltersSearchPagination.tsx > FiltersSearchPagination
main*
0 0 0

```tsx
"use client";
import React, { useState, useEffect } from "react";
import { ProductType } from "../../pTypes/productType";

type FiltersSearchPaginationProps = {
  products: ProductType[];
  onFilteredData: (data: ProductType[]) => void;
};

const FiltersSearchPagination: React.FC<FiltersSearchPaginationProps> = ({
  products,
  onFilteredData,
}) => {
  // States
  const [filteredProducts, setFilteredProducts] = useState<ProductType[]>(products);
  const [searchTerm, setSearchTerm] = useState<string>("");
  const [selectedCategory, setSelectedCategory] = useState<string>("All");
  const [categories, setCategories] = useState<string[]>([]);
  const [currentPage, setCurrentPage] = useState<number>(1);
  const itemsPerPage = 8; // Customize items per page

  // Extract unique categories
  useEffect(() => {
    const uniqueCategories = [
      "All",
  ...Array.from(new Set(products.map((p) => p.category || "Uncategorized"))),
];
    setCategories(uniqueCategories);
  }, [products]);

  // Filter products based on category and search term
  useEffect(() => {
    let tempProducts = products;

    if (selectedCategory !== "All") {
      tempProducts = tempProducts.filter(
        (p) => p.category === selectedCategory
      );
    }
```



```tsx
    if (searchTerm) {
      tempProducts = tempProducts.filter((p) =>
        p.productName.toLowerCase().includes(searchTerm.toLowerCase())
      );
    }

    setFilteredProducts(tempProducts);
    onFilteredData(tempProducts.slice(0, itemsPerPage)); // Emit initial data for pagination
  }, [selectedCategory, searchTerm, products, onFilteredData]);

  // Handle pagination
  const handlePageChange = (pageNumber: number) => {
    setCurrentPage(pageNumber);
    const startIndex = (pageNumber - 1) * itemsPerPage;
    const paginatedData = filteredProducts.slice(
      startIndex,
      startIndex + itemsPerPage
    );
    onFilteredData(paginatedData);
  };

  return (
    <div>
      {/* Search Bar */}
      <div className="flex gap-4 mb-6">
        <input
          type="text"
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          placeholder="Search by product name"
          className="border rounded-lg px-4 py-2 flex-1"
        />

        {/* Category Filter */}
        <select
          value={selectedCategory}
          onChange={(e) => setSelectedCategory(e.target.value)}
```



```
          value={selectedCategory}
          onChange={(e) => setSelectedCategory(e.target.value)}
          className="border rounded-lg px-4 py-2"
        >
          {categories.map((category) => (
            <option key={category} value={category}>
              {category}
            </option>
          ))}
        </select>
      </div>

      {/* Pagination */}
      <div className="flex justify-center mt-6">
        {Array.from(
          { length: Math.ceil(filteredProducts.length / itemsPerPage) },
          (_, i) => (
            <button
              key={i}
              onClick={() => handlePageChange(i + 1)}
              className={`mx-1 px-4 py-2 border rounded-md transition duration-200 ${
                currentPage === i + 1
                  ? "bg-blue-500 text-white"
                  : "bg-white text-gray-700"
              }`}
            >
              {i + 1}
            </button>
          )
        )}
      </div>
    </div>
  );
};

export default FiltersSearchPagination;
```

Technical Report

Steps Taken to Build and Integrate Components

1. Dynamic Route Creation:

- Implemented `app/item/[slug]/page.tsx` for dynamic product listing using slug-based routing.
- Integrated dynamic data rendering for individual product pages.

2. Type Definitions:

- Created pTypes/productType.ts to define types like id, productName, price, description, etc., ensuring type safety across the application.

3. FiltersSearchPagination Component:

- Built a reusable component for product search, category filtering, and pagination.
- Integrated it with dynamic product data to enhance user interactivity and navigation.

Challenges Faced and Solutions Implemented

1. Dynamic Data Handling:

- Challenge: Ensuring accurate slug-based routing for product pages.
- Solution: Used Next.js dynamic routing and validated slug data.

2. Type Safety:

- Challenge: Avoiding runtime errors with dynamic data.
- Solution: Defined comprehensive TypeScript types in

3. Search and Pagination Logic:

- Challenge: implementing efficient filtering and pagination.
- Solution: Utilized React state and hooks for dynamic filtering and paginated rendering.

Best Practices Followed

- Used TypeScript for type safety and maintainability.
- Ensured modularity by separating concerns into components and utility files.
- Followed React and Next.js best practices for performance and scalability