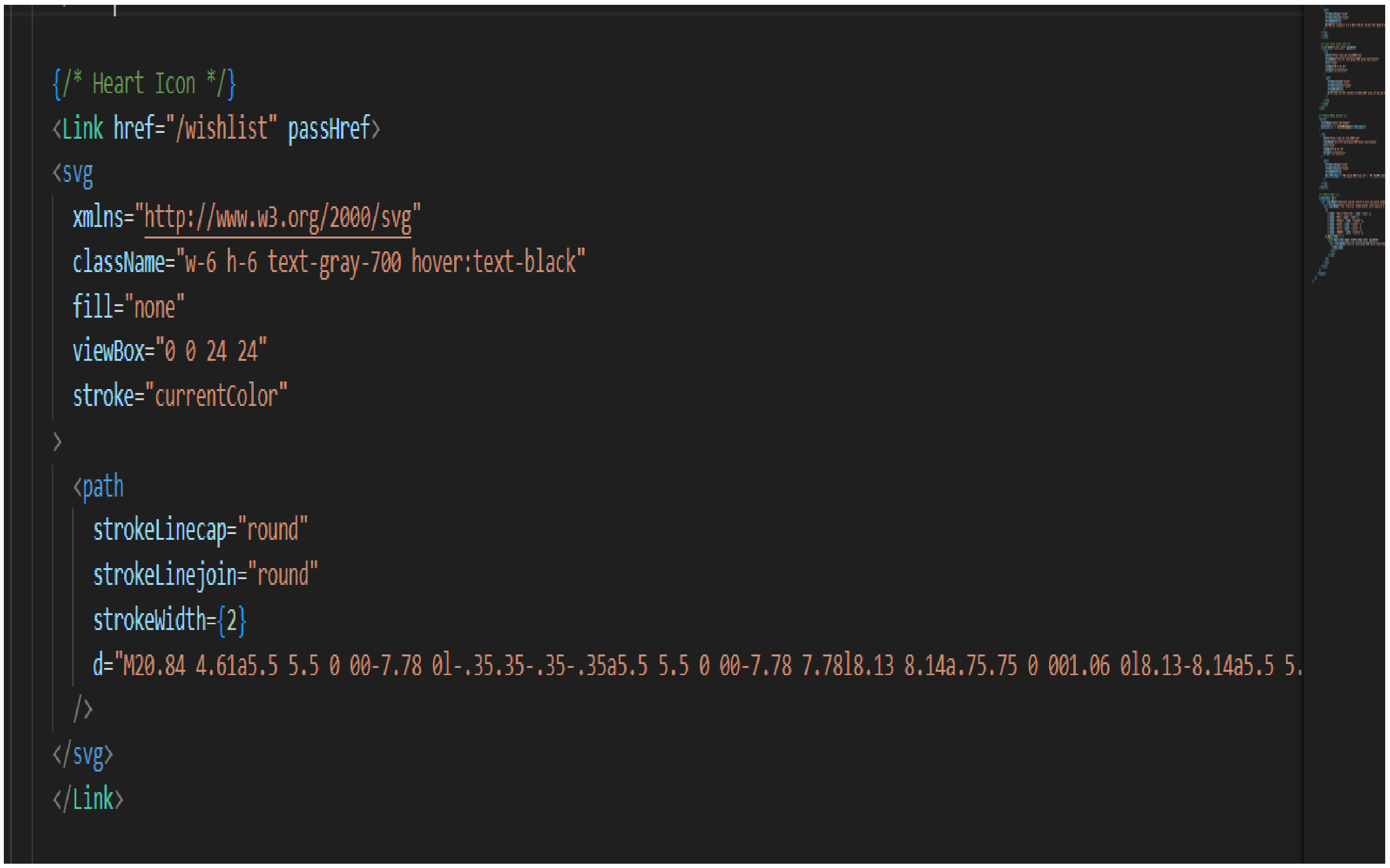
Hackathon - Day 4 Documentation: In-Depth Overview of Dynamic Components and Functionalities

Introduction

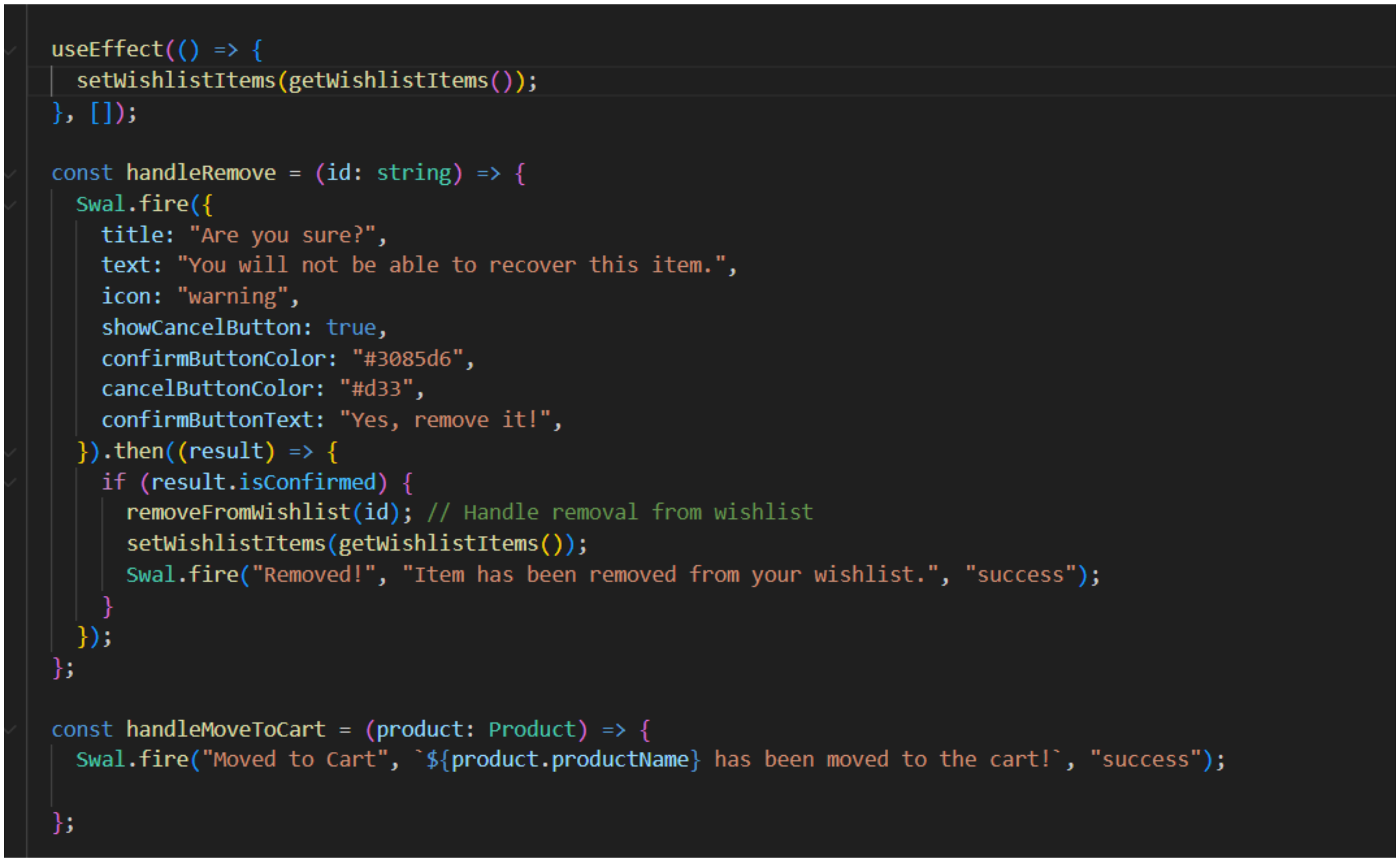
On Day 4 of the hackathon, the primary focus was on integrating dynamic components and improving the functionality of our application. This document presents a detailed overview of the components created, their functionalities, and how they dynamically interact with user inputs and backend systems.

Dynamic Components Overview

1. **Header Component**
   - **Location**: `components/header/page.tsx`
   - **Location**: `components/navbar.tsx`
   - **Description**: The Header component manages navigation and provides dynamic updates for cart items.
   - **Key Features**:
     - **Dynamic wishlist**: Shows the total number of items in the wishlist.
     - **Wishlist**: Updates to reflect the number of items added to the wishlist.



```
{/* Heart Icon */}
<Link href="/wishlist" passHref>
<svg
  xmlns="http://www.w3.org/2000/svg"
  className="w-6 h-6 text-gray-700 hover:text-black"
  fill="none"
  viewBox="0 0 24 24"
  stroke="currentColor"
>
  <path
    strokeLinecap="round"
    strokeLinejoin="round"
    strokeWidth={2}
    d="M20.84 4.61a5.5 5.5 0 00-7.78 0l-.35.35-.35-.35a5.5 5.5 0 00-7.78 7.78l8.13 8.14a.75.75 0 001.06 0l8.13-8.14a5.5 5.
  />
</svg>
</Link>
```



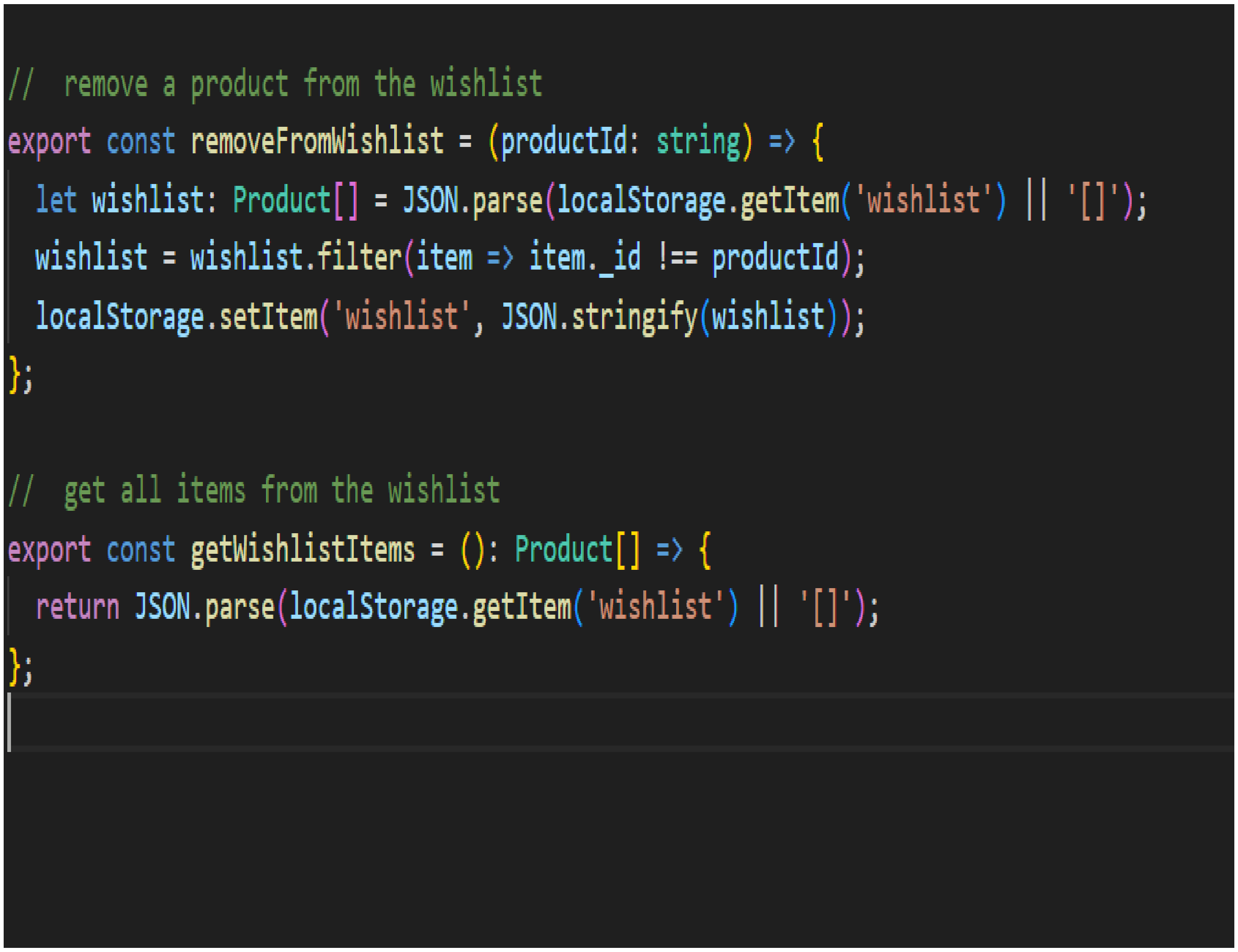
```
useEffect(() => {
  setWishlistItems(getWishlistItems());
}, []);

const handleRemove = (id: string) => {
  Swal.fire({
    title: "Are you sure?",
    text: "You will not be able to recover this item.",
    icon: "warning",
    showCancelButton: true,
    confirmButtonColor: "#3085d6",
    cancelButtonColor: "#d33",
    confirmButtonText: "Yes, remove it!",
  }).then((result) => {
    if (result.isConfirmed) {
      removeFromWishlist(id); // Handle removal from wishlist
      setWishlistItems(getWishlistItems());
      Swal.fire("Removed!", "Item has been removed from your wishlist.", "success");
    }
  });
};

const handleMoveToCart = (product: Product) => {
  Swal.fire("Moved to Cart", `${product.productName} has been moved to the cart!`, "success");

};
```

2. Product Page Component
   - Location: actions/actions.ts
   - Description: Dynamically renders the product details based on the product ID.
   - Functionalities: o Fetches product details from the backend
     - Handles adding items to the wishlist.
     - Stores wishlist data in localStorage for persistence.

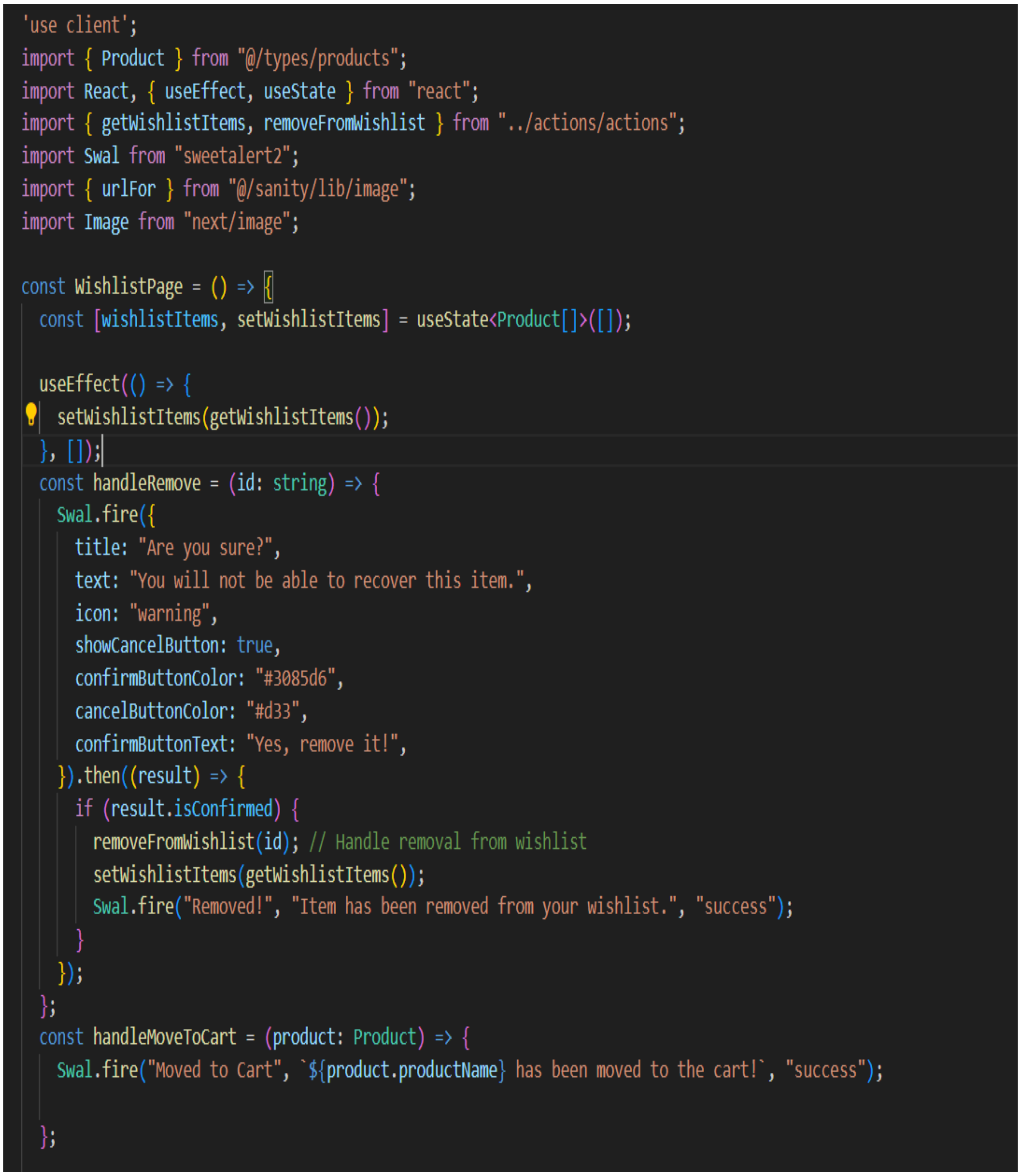


```
//  remove a product from the wishlist
export const removeFromWishlist = (productId: string) => {
  let wishlist: Product[] = JSON.parse(localStorage.getItem('wishlist') || '[]');
  wishlist = wishlist.filter(item => item._id !== productId);
  localStorage.setItem('wishlist', JSON.stringify(wishlist));
};

//  get all items from the wishlist
export const getWishlistItems = (): Product[] => {
  return JSON.parse(localStorage.getItem('wishlist') || '[]');
};
```

3. Wishlist Route • Location: Wishlist / page.tsx
   - Description: Displays all items added to the wishlist.
   - Functionalities: o Fetches wishlist data from localStorage.

   Allows users to remove items from the wishlist dynamically



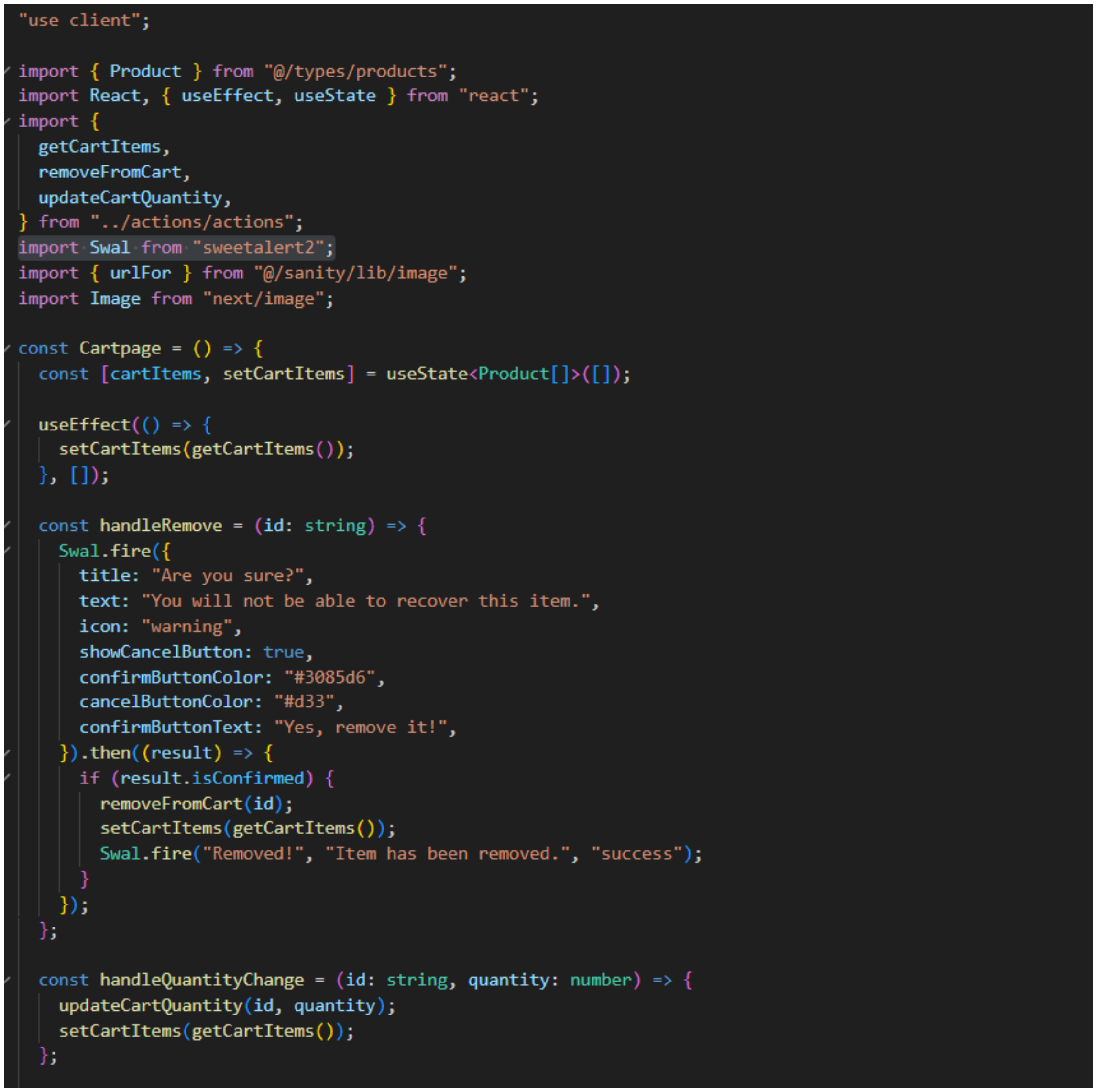
```tsx
'use client';
import { Product } from "@/types/products";
import React, { useEffect, useState } from "react";
import { getWishlistItems, removeFromWishlist } from "../actions/actions";
import Swal from "sweetalert2";
import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";

const WishlistPage = () => {
  const [wishlistItems, setWishlistItems] = useState<Product[]>([]);

  useEffect(() => {
    setWishlistItems(getWishlistItems());
  }, []);
  const handleRemove = (id: string) => {
    Swal.fire({
      title: "Are you sure?",
      text: "You will not be able to recover this item.",
      icon: "warning",
      showCancelButton: true,
      confirmButtonColor: "#3085d6",
      cancelButtonColor: "#d33",
      confirmButtonText: "Yes, remove it!",
    }).then((result) => {
      if (result.isConfirmed) {
        removeFromWishlist(id); // Handle removal from wishlist
        setWishlistItems(getWishlistItems());
        Swal.fire("Removed!", "Item has been removed from your wishlist.", "success");
      }
    });
  };
  const handleMoveToCart = (product: Product) => {
    Swal.fire("Moved to Cart", `${product.productName} has been moved to the cart!`, "success");

  };
```

4. Cart Slice • Location: redux/cartslice.ts
   • Description: Manages cart state
   • Functionalities: Adds items to the cart.
   o Removes items from the cart.
   o Updates item quantities dynamically.



```tsx
"use client";

import { Product } from "@/types/products";
import React, { useEffect, useState } from "react";
import {
  getCartItems,
  removeFromCart,
  updateCartQuantity,
} from "../actions/actions";
import Swal from "sweetalert2";
import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";

const Cartpage = () => {
  const [cartItems, setCartItems] = useState<Product[]>([]);

  useEffect(() => {
    setCartItems(getCartItems());
  }, []);

  const handleRemove = (id: string) => {
    Swal.fire({
      title: "Are you sure?",
      text: "You will not be able to recover this item.",
      icon: "warning",
      showCancelButton: true,
      confirmButtonColor: "#3085d6",
      cancelButtonColor: "#d33",
      confirmButtonText: "Yes, remove it!",
    }).then((result) => {
      if (result.isConfirmed) {
        removeFromCart(id);
        setCartItems(getCartItems());
        Swal.fire("Removed!", "Item has been removed.", "success");
      }
    });
  };

  const handleQuantityChange = (id: string, quantity: number) => {
    updateCartQuantity(id, quantity);
    setCartItems(getCartItems());
  };
```

Functionalities Implemented

1. **Dynamic Badge Update**
   - **Description**: Automatically updates the cart and wishlist badges in the navbar component based on state changes.
2. **Persistent Wishlist**
   - **Description**: Ensures the wishlist data persists even after a page reload by storing it in localStorage.
   - **Implementation Steps**:
     - On component mount, fetched the wishlist data from localStorage.
     - Updated localStorage every time the wishlist state changed to keep data consistent.
3. **Dynamic Product Details**
   - **Description**: Renders product details dynamically based on the product ID present in the URL.
   - **Implementation Steps**:
     - Displayed the data dynamically in the product details component.
4. **Interactive Wishlist Button**
   - **Description**: Changes the heart icon color to red when an item is added to the wishlist.

Conclusion

On Day 4, the focus was on enhancing the user experience by building dynamic and interactive components that respond to user actions. With the use of tools like localStorage, and conditional rendering, the application now offers a seamless and engaging experience for users.
