S Farzana shah

Day 4 - Building Dynamic Frontend Components for EasyBuy Marketplace

On Day 4, I focused on building dynamic frontend components for the EasyBuy Marketplace. This project involved the creation and integration of reusable, responsive, and modular components that enhance the user experience. I learned how to structure these components efficiently, handle state management, and optimize performance. In this documentation, I'll walk you through the key components I've developed, how I implemented them, and the challenges I faced.

The components built in this session include:

1. **Product Listing Component**
2. **Product Detail Component**
3. **Cart Component**
4. **Wishlist Component**
5. **Checkout Flow Component**
6. **Footer Components**
7. **Header Components**
8. **Order Tracking Component**
9. **Search Bar Component**

S Farzana shah

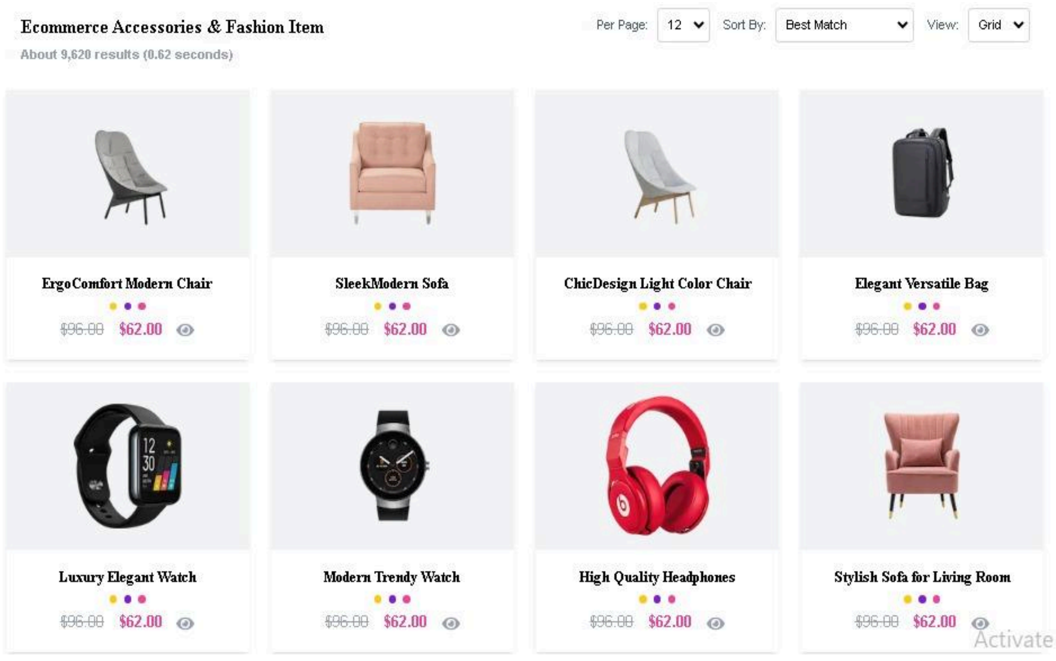
1. Product Listing Component

What I Learned:

The Product Listing component is essential for displaying a list of products dynamically. This task helped me understand how to handle data in React and render multiple items in a list. I also implemented filtering capabilities using categories and tags, allowing users to sort products based on their preferences.

How I Implemented It:

- I fetched product data and mapped it to render each product's name, image, and price.
- Added filters by categories and tags to refine the product list based on user selection.


Ecommerce Accessories & Fashion Item

About 9,620 results (0.62 seconds)

Per Page: 12 Sort By: Best Match View: Grid

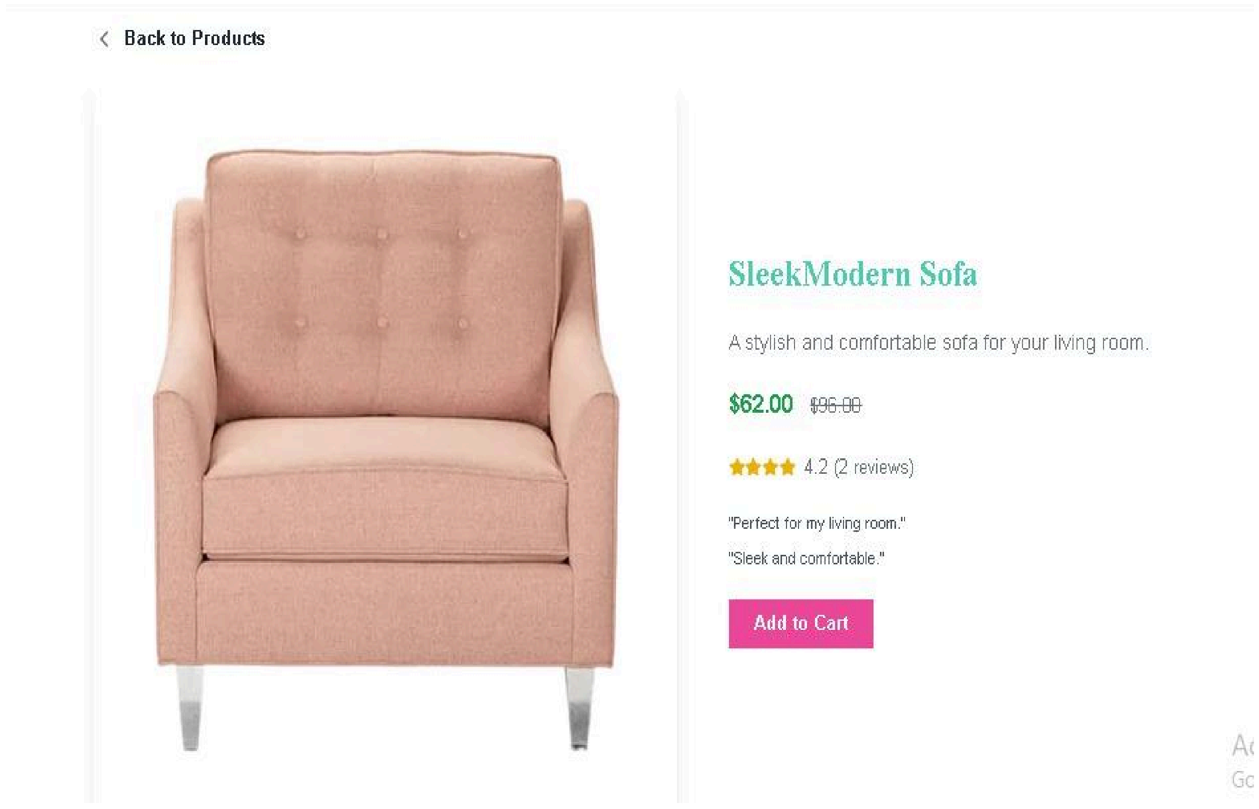
2. Product Detail Component

What I Learned:

This component displays detailed information for a specific product. I learned how to use React's state management to handle dynamic content (like images, price, and description) based on the product the user selects.

How I Implemented It:

- **I** created a dedicated page where users can click on a product to see detailed information.
- The product detail includes an image gallery, description, rating, price, and a button to add the product to the cart.


< Back to Products

SleekModern Sofa

A stylish and comfortable sofa for your living room.

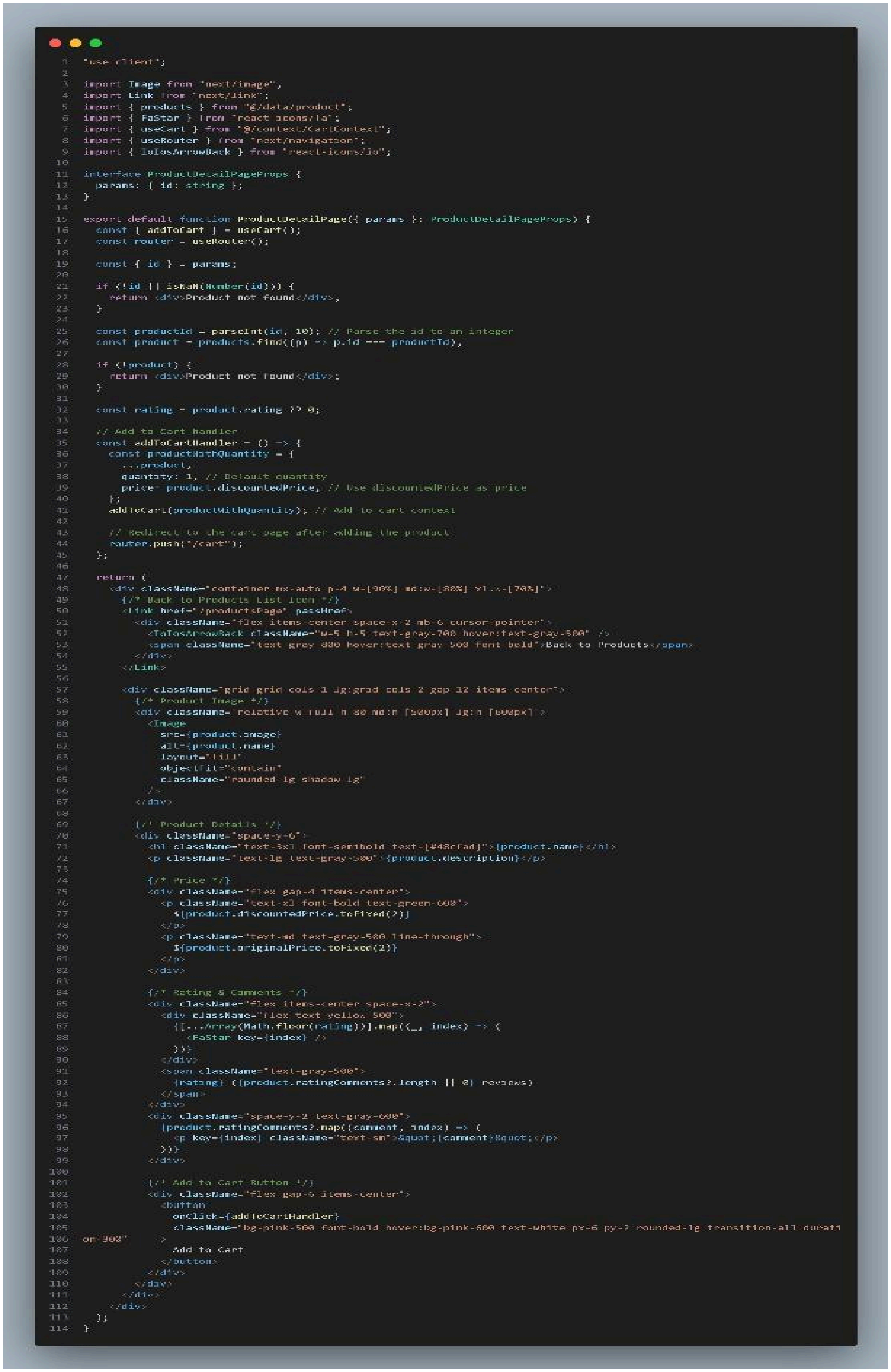
$62.00 ~~$96.00~~

★★★★ 4.2 (2 reviews)

"Perfect for my living room."

"Sleek and comfortable."

S Farzana shah



```
"use client";

import Image from "next/image";
import Link from "next/link";
import { products } from "@/data/product";
import { FaStar } from "react-icons/fa";
import { useCart } from "@/context/CartContext";
import { useRouter } from "next/navigation";
import { IoIosArrowBack } from "react-icons/io";

interface ProductDetailPageProps {
  params: { id: string };
}

export default function ProductDetailPage({ params }: ProductDetailPageProps) {
  const { addToCart } = useCart();
  const router = useRouter();

  const { id } = params;

  if (!id || isNaN(Number(id))) {
    return <div>Product not found</div>;
  }

  const productId = parseInt(id, 10); // Parse the id to an integer
  const product = products.find((p) => p.id === productId);

  if (!product) {
    return <div>Product not found</div>;
  }

  const rating = product.rating ?? 0;

  // Add to Cart handler
  const addToCartHandler = () => {
    const productWithQuantity = {
      ...product,
      quantity: 1, // Default quantity
      price: product.discountedPrice, // Use discountedPrice as price
    };
    addToCart(productWithQuantity); // Add to cart context

    // Redirect to the cart page after adding the product
    router.push("/cart");
  };

  return (
    <div className="container mx-auto p-4 w-[90%] md:w-[80%] xl:w-[70%]">
      {/* Back to Products List Icon */}
      <Link href="/productsPage" passHref>
        <div className="flex items-center space-x-2 mb-6 cursor-pointer">
          <IoIosArrowBack className="w-5 h-5 text-gray-700 hover:text-gray-500" />
          <span className="text-gray-800 hover:text-gray-500 font-bold">Back to Products</span>
        </div>
      </Link>

      <div className="grid grid-cols-1 lg:grid-cols-2 gap-12 items-center">
        {/* Product Image */}
        <div className="relative w-full h-80 md:h-[500px] lg:h-[600px]">
          <Image
            src={product.image}
            alt={product.name}
            layout="fill"
            objectFit="contain"
            className="rounded-lg shadow-lg"
          />
        </div>

        {/* Product Details */}
        <div className="space-y-6">
          <h1 className="text-3xl font-semibold text-[#48cfad]">{product.name}</h1>
          <p className="text-lg text-gray-500">{product.description}</p>

          {/* Price */}
          <div className="flex gap-4 items-center">
            <p className="text-xl font-bold text-green-600">
              ${product.discountedPrice.toFixed(2)}
            </p>
            <p className="text-md text-gray-500 line-through">
              ${product.originalPrice.toFixed(2)}
            </p>
          </div>

          {/* Rating & Comments */}
          <div className="flex items-center space-x-2">
            <div className="flex text-yellow-500">
              {[...Array(Math.floor(rating))].map((_, index) => (
                <FaStar key={index} />
              ))}
            </div>
            <span className="text-gray-500">
              {rating} ({product.ratingComments?.length || 0} reviews)
            </span>
          </div>
          <div className="space-y-2 text-gray-600">
            {product.ratingComments?.map((comment, index) => (
              <p key={index} className="text-sm">&quot;{comment}&quot;</p>
            ))}
          </div>

          {/* Add to Cart Button */}
          <div className="flex gap-6 items-center">
            <button
              onClick={addToCartHandler}
              className="bg-pink-500 font-bold hover:bg-pink-600 text-white px-6 py-3 rounded-lg transition-all durati
on-300"       >
              Add to Cart
            </button>
          </div>
        </div>
      </div>
    </div>
  );
}
```

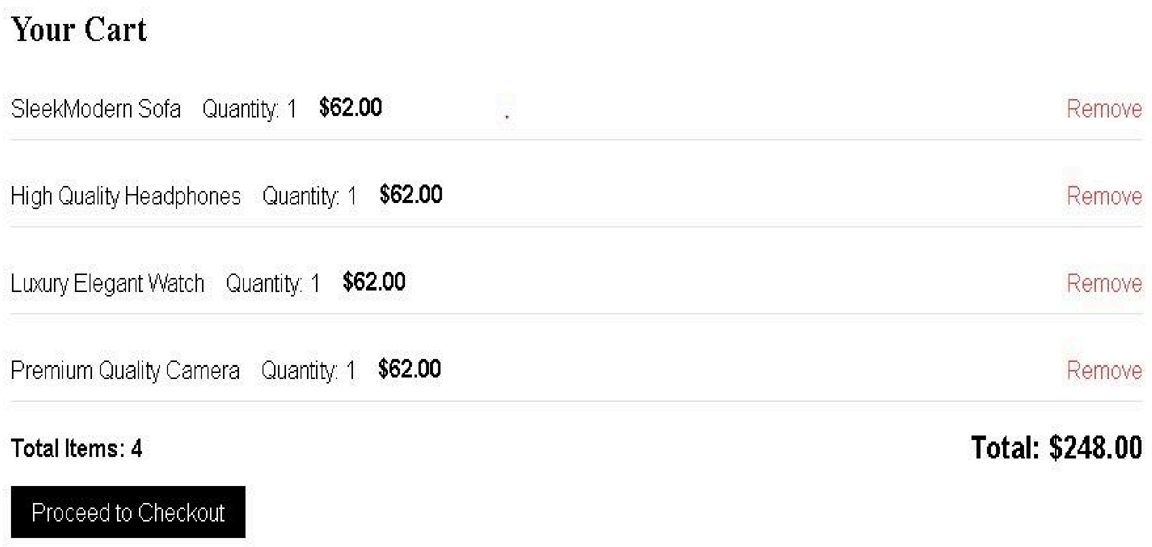
3. Cart Component :

What I Learned:

Building the Cart component helped me grasp how to manage the cart state and update it as users add or remove items. I learned how to dynamically adjust the cart's total price based on the products inside it.

How I Implemented It:

- The cart component lists all the products added by the user, including quantity, price, and subtotal.
- Users can update the quantity or remove products, and the total price dynamically updates accordingly.



Your Cart

SleekModern Sofa Quantity: 1 **$62.00** Remove

High Quality Headphones Quantity: 1 **$62.00** Remove

Luxury Elegant Watch Quantity: 1 **$62.00** Remove

Premium Quality Camera Quantity: 1 **$62.00** Remove

Total Items: 4 **Total: $248.00**

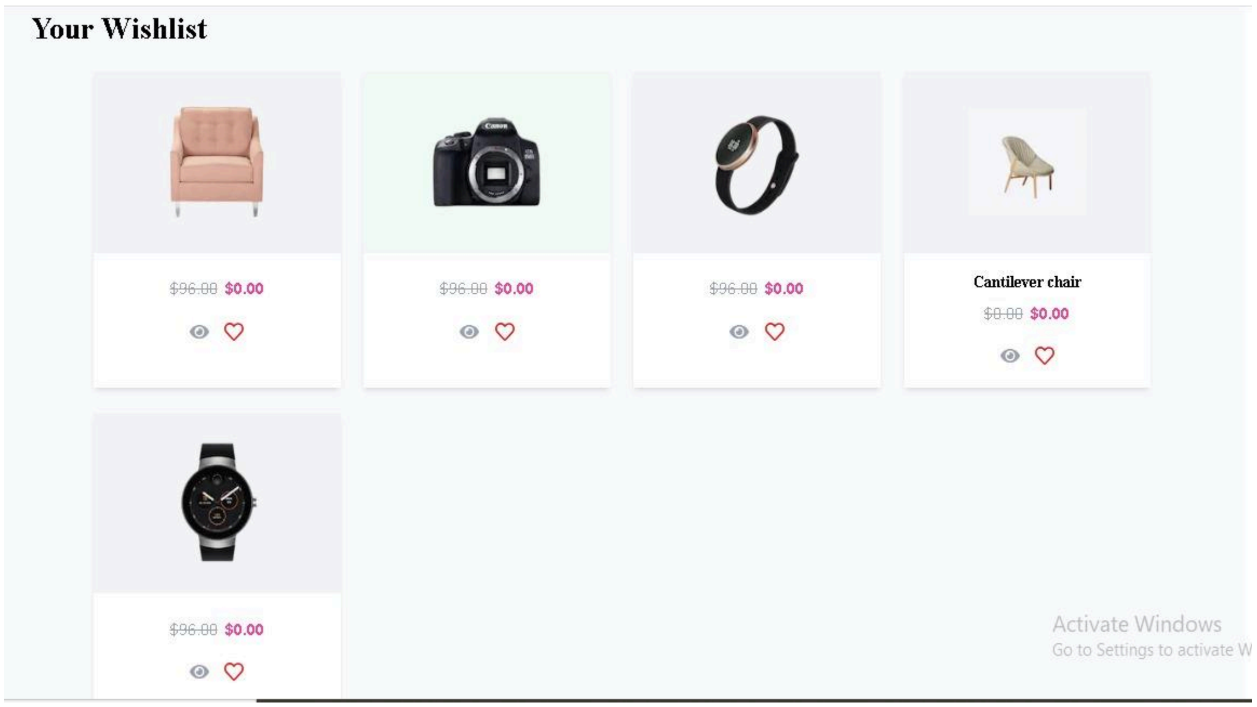
4. Wishlist Component

What I Learned:

The Wishlist component was similar to the Cart component but without the purchasing option. I learned how to save products for later and how to transfer them to the cart.

How I Implemented It:

- Products can be saved to the wishlist and viewed at any time.
- I added a button to move products from the wishlist to the cart when the user is ready to purchase.



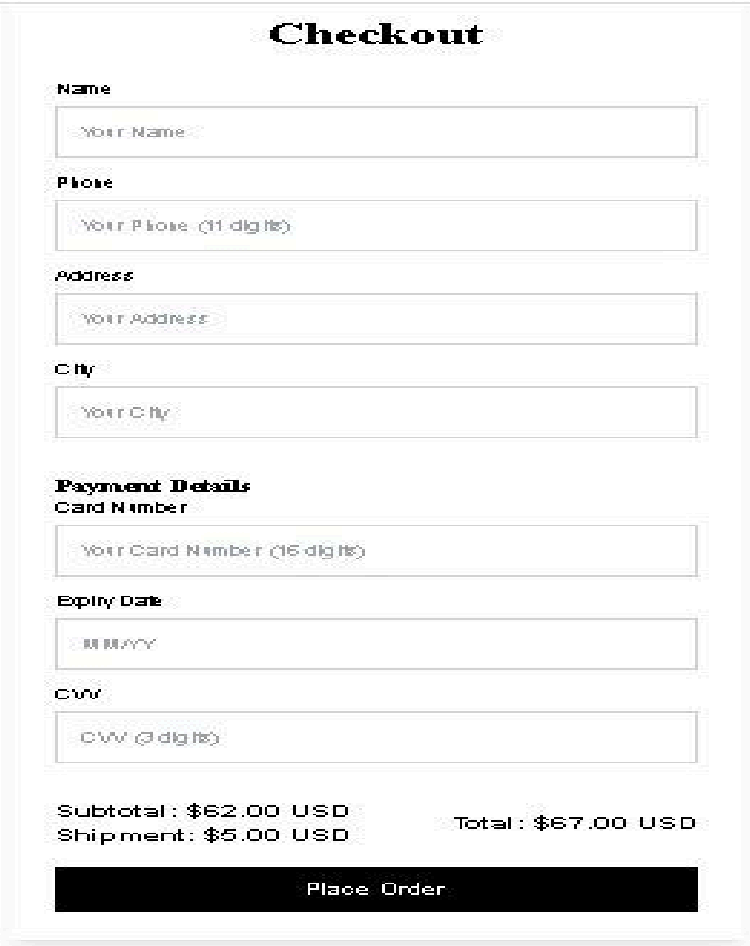
5. Checkout Flow Component

What I Learned:

This was one of the more complex components to build. The Checkout Flow involves handling user input for shipping, billing, and payment information. I learned how to manage multiple steps in a process and collect all necessary details to complete an order.

How I Implemented It:

- I broke down the checkout process into steps: shipping information, payment details, and order summary.
- Each step collects relevant information and, at the end, confirms the order before submission.



Checkout

Name

Your Name

Phone

Your Phone (11 digits)

Address

Your Address

City

Your City

Payment Details

Card Number

Your Card Number (16 digits)

Expiry Date

MM/YY

CVV

CVV (3 digits)

Subtotal: $62.00 USD
Shipment: $5.00 USD

Total: $67.00 USD

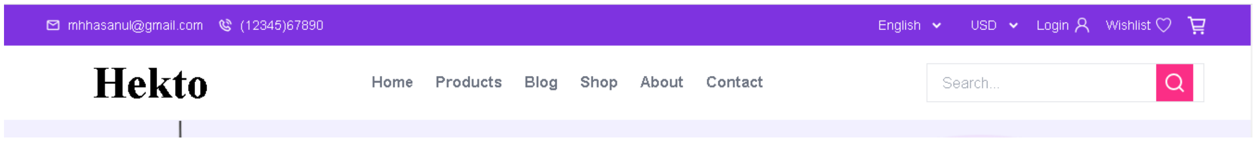
6. Footer and Header Components

What I Learned:

I learned the importance of having consistent navigation elements like the header and footer throughout the website. These components help users navigate between different pages and find useful information, such as links to contact details, social media, and the shopping cart.

How I Implemented It:

- The **Header** includes the logo, navigation links, and a search bar.
- The **Footer** contains additional links like About, Contact, and social media icons.

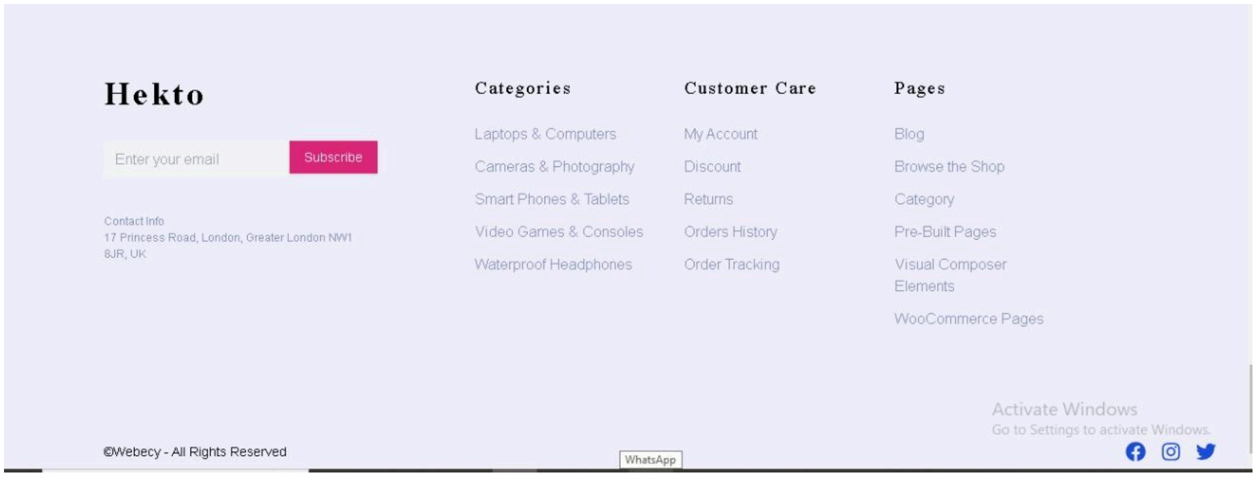


mhhasanul@gmail.com (12345)67890 English USD Login Wishlist

Hekto

Home Products Blog Shop About Contact

Search...



Hekto

Enter your email Subscribe

Contact Info
17 Princess Road, London, Greater London NW1 8JR, UK

Categories	Customer Care	Pages
Laptops & Computers	My Account	Blog
Cameras & Photography	Discount	Browse the Shop
Smart Phones & Tablets	Returns	Category
Video Games & Consoles	Orders History	Pre-Built Pages
Waterproof Headphones	Order Tracking	Visual Composer Elements
		WooCommerce Pages

Activate Windows
Go to Settings to activate Windows.

©Webecy - All Rights Reserved

WhatsApp

---

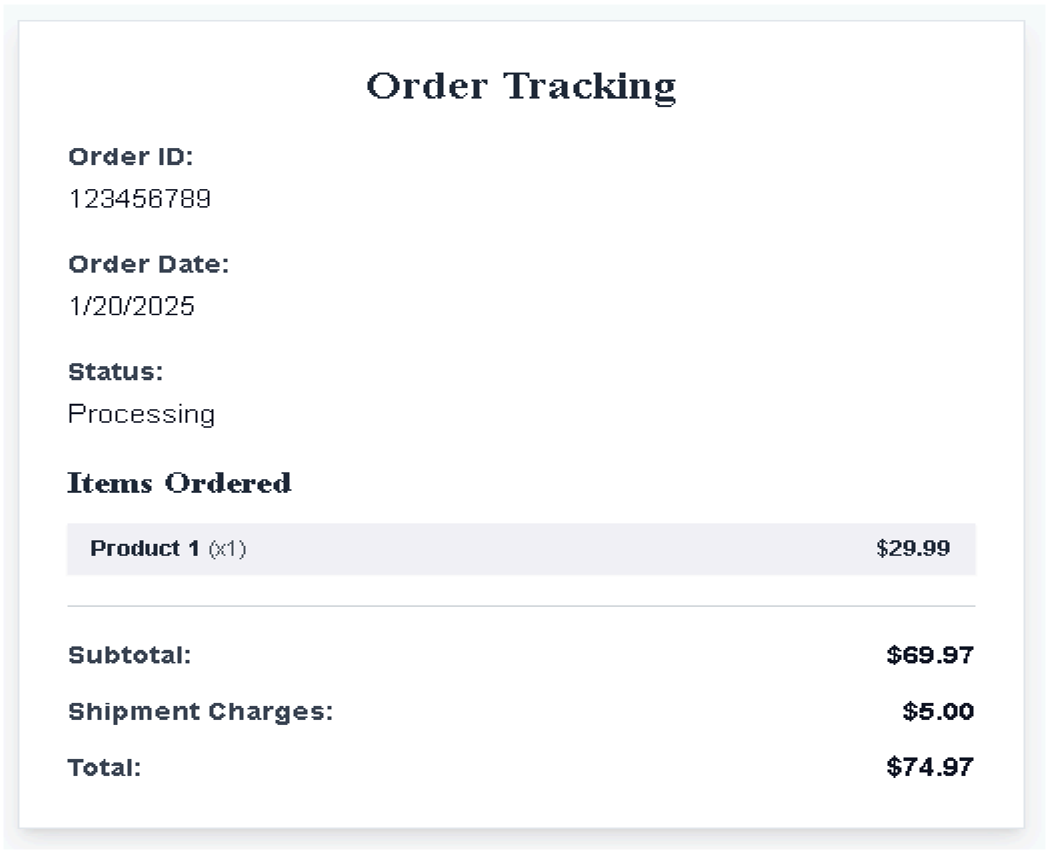
7. Order Tracking Component

What I Learned:

The Order Tracking component helped me understand how to provide users with real-time information about their order's status. I learned how to fetch order status data and display updates as the order moves through different stages (e.g., processing, shipped, delivered).

How I Implemented It:

- Users can view the current status of their order and any tracking details.
- The component fetches the tracking information from an API and displays it in a user-friendly format.



Order Tracking

Order ID:
123456789

Order Date:
1/20/2025

Status:
Processing

Items Ordered

Product 1 (x1)	$29.99

Subtotal:	**$69.97**
Shipment Charges:	**$5.00**
Total:	**$74.97**

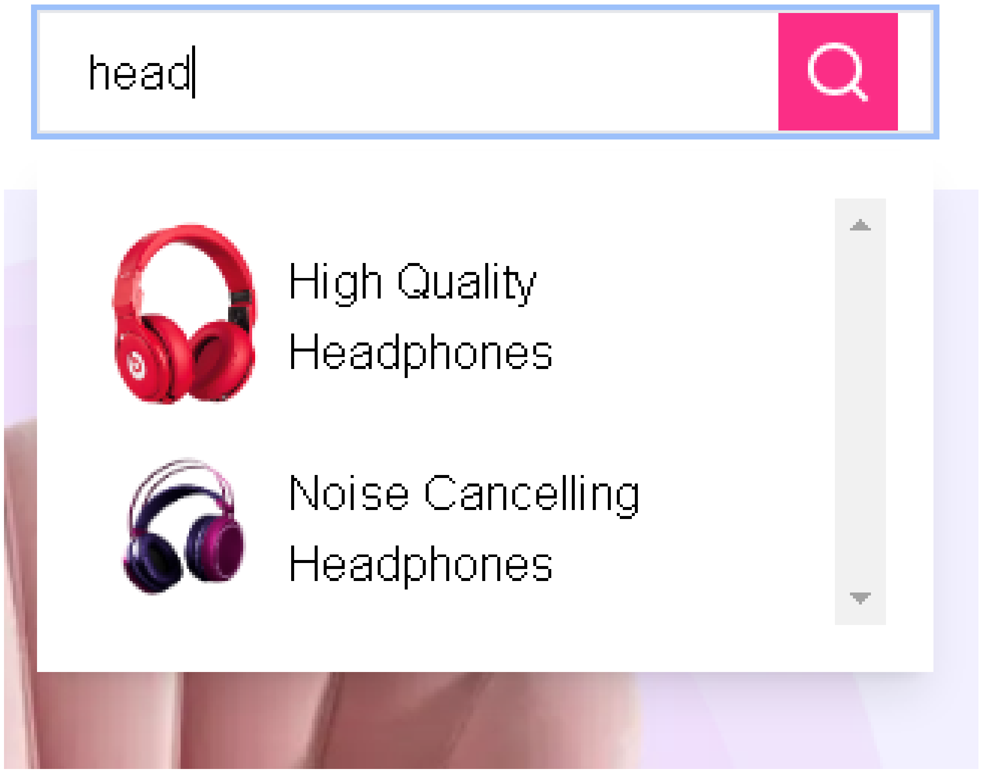
8. Search Bar Component

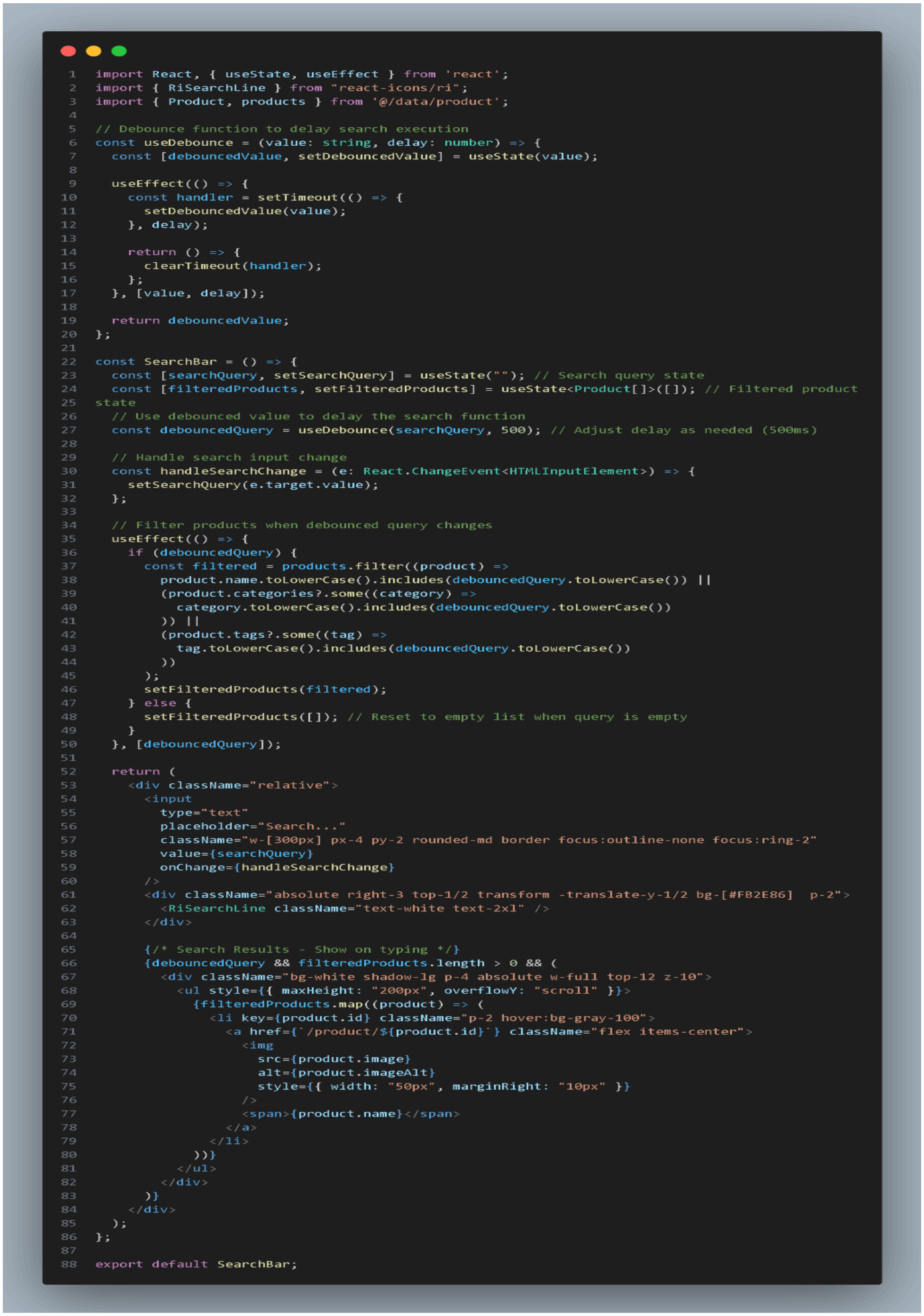
What I Learned:

The Search Bar component was an interesting challenge. I learned how to implement real-time search functionality, allowing users to search for products by name, category, or tags.

How I Implemented It:

- As the user types, the search bar filters through the product list and displays suggestions based on the input.
- I used dynamic search filtering to instantly show relevant results.





```tsx
import React, { useState, useEffect } from 'react';
import { RiSearchLine } from "react-icons/ri";
import { Product, products } from '@/data/product';

// Debounce function to delay search execution
const useDebounce = (value: string, delay: number) => {
  const [debouncedValue, setDebouncedValue] = useState(value);

  useEffect(() => {
    const handler = setTimeout(() => {
      setDebouncedValue(value);
    }, delay);

    return () => {
      clearTimeout(handler);
    };
  }, [value, delay]);

  return debouncedValue;
};

const SearchBar = () => {
  const [searchQuery, setSearchQuery] = useState(""); // Search query state
  const [filteredProducts, setFilteredProducts] = useState<Product[]>([]); // Filtered product
state
  // Use debounced value to delay the search function
  const debouncedQuery = useDebounce(searchQuery, 500); // Adjust delay as needed (500ms)

  // Handle search input change
  const handleSearchChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setSearchQuery(e.target.value);
  };

  // Filter products when debounced query changes
  useEffect(() => {
    if (debouncedQuery) {
      const filtered = products.filter((product) =>
        product.name.toLowerCase().includes(debouncedQuery.toLowerCase()) ||
        (product.categories?.some((category) =>
          category.toLowerCase().includes(debouncedQuery.toLowerCase())
        )) ||
        (product.tags?.some((tag) =>
          tag.toLowerCase().includes(debouncedQuery.toLowerCase())
        ))
      );
      setFilteredProducts(filtered);
    } else {
      setFilteredProducts([]); // Reset to empty list when query is empty
    }
  }, [debouncedQuery]);

  return (
    <div className="relative">
      <input
        type="text"
        placeholder="Search..."
        className="w-[300px] px-4 py-2 rounded-md border focus:outline-none focus:ring-2"
        value={searchQuery}
        onChange={handleSearchChange}
      />
      <div className="absolute right-3 top-1/2 transform -translate-y-1/2 bg-[#FB2E86]  p-2">
        <RiSearchLine className="text-white text-2xl" />
      </div>

      {/* Search Results - Show on typing */}
      {debouncedQuery && filteredProducts.length > 0 && (
        <div className="bg-white shadow-lg p-4 absolute w-full top-12 z-10">
          <ul style={{ maxHeight: "200px", overflowY: "scroll" }}>
            {filteredProducts.map((product) => (
              <li key={product.id} className="p-2 hover:bg-gray-100">
                <a href={`/product/${product.id}`} className="flex items-center">
                  <img
                    src={product.image}
                    alt={product.imageAlt}
                    style={{ width: "50px", marginRight: "10px" }}
                  />
                  <span>{product.name}</span>
                </a>
              </li>
            ))}
          </ul>
        </div>
      )}
    </div>
  );
};

export default SearchBar;
```

A Technical Report Summarizing

Today, I focused on building and integrating key dynamic components for the EasyBuy Marketplace. This included implementing product listing, cart management, checkout flow, and order tracking functionality. Along with the development of these components, I have also outlined the challenges I faced and how I addressed them, while following best practices.

- **Product Listing:**
  - **I started by building the Product Listing component. This component dynamically displays products by mapping over an array of product data.**
  - **I added filters such as category and price range to allow users to narrow down their product search.**
- **Cart Management:**
  - **Next, I created the Cart component, which allows users to add, update, or remove products from their shopping cart.**
- **Checkout Flow:**
  - **I then worked on building a multi-step Checkout Flow. This includes steps for user information (name, phone, address) and payment details (credit card number, CVV, expiry date).**
  - **I implemented form validation for fields such as phone number and card details to ensure users enter the correct information.**
  - **After the user submits the checkout form, an order ID is generated and passed to the Order Tracking page, which simulates order confirmation.**
- **Order Tracking:**
  - **Finally, I developed an Order Tracking page that fetches the order details using the order ID passed from the checkout flow.**
  - **On the Order Tracking page, users can see the order status (e.g., "Processing", "Shipped", etc.), the items ordered, and the total cost.**

2. Challenges Faced:

- **Handling Complex State:**
  - **One of the challenges I encountered was managing complex state across multiple components (cart, checkout, etc.).**
  - **A challenge I faced was ensuring the user inputs (such as phone number and card details) were correctly validated.**
  - **I implemented checks for fields like phone number (11 digits) and card number (16 digits), ensuring that incorrect data doesn't get submitted.**
  - **For the CVV, I added validation for exactly 3 digits.**
- **Responsive Design:**
  - **To ensure the app works on various screen sizes, I focused on making the components responsive.**

<u>3. Best Practices Followed During Development:</u>

- **Component Reusability:**
  - I followed the principle of reusability for all components. For example, the cart component can be reused in the checkout and product detail pages.
  - This helps in making the code more modular and maintainable.
- **Code Structure and Readability:**
  - I followed best practices for code readability by breaking the app into small, self-contained components. This helps in making the code easier to maintain and understand.

In summary, Day 4 involved building and integrating key components for the EasyBuy Marketplace. I followed best practices such as component reusability, state management , and ensuring form validations. I also documented the challenges faced, such as complex state management, form validation, and responsive design, and provided solutions to address these challenges.

Overall, the project has progressed well, and the key features like product listing, cart management, checkout flow, and order tracking are functioning as expected.
