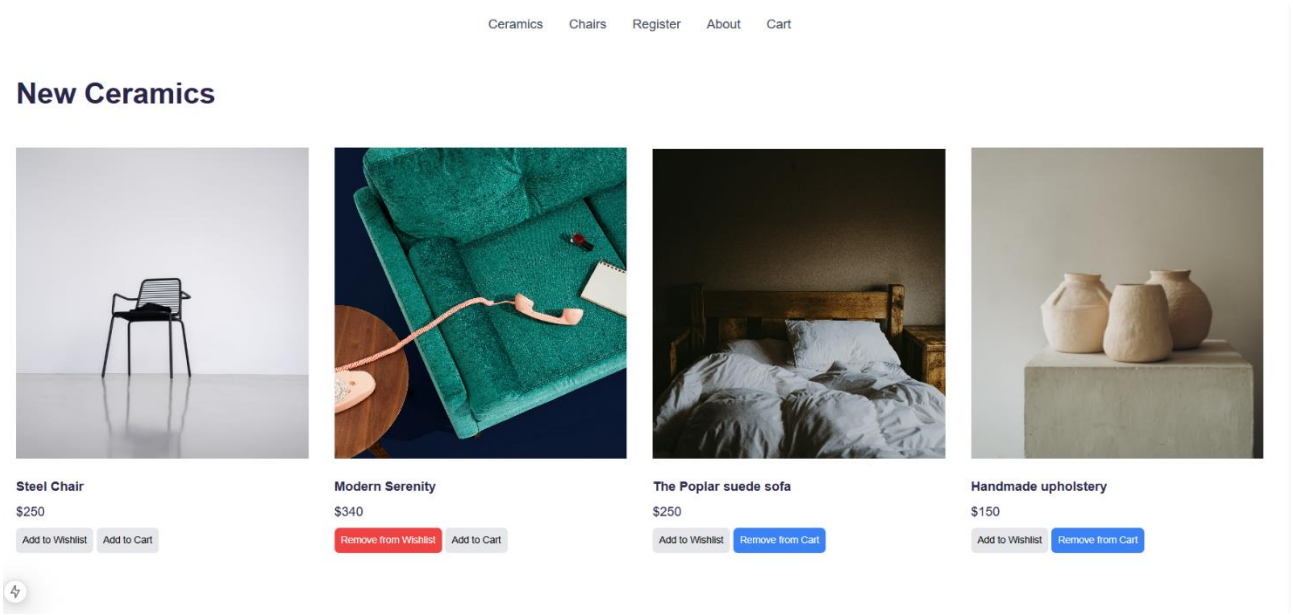
Hackathon Day – 04

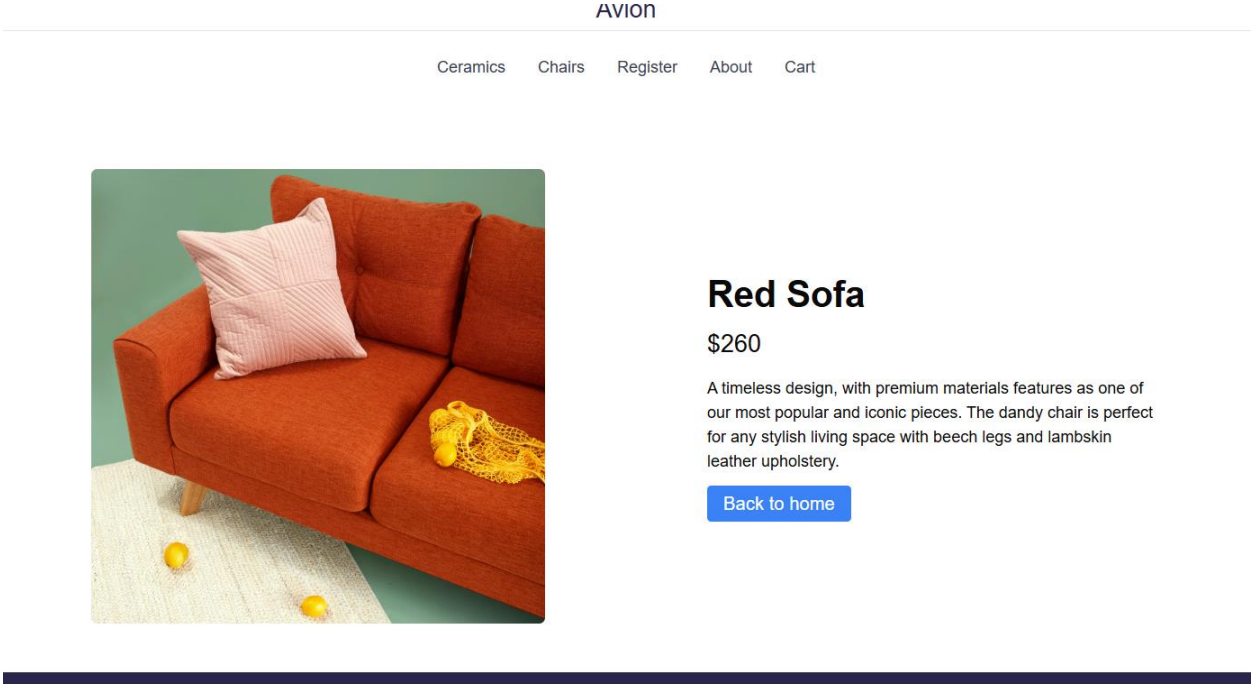
Building Dynamic Components for my General E – Commerce Marketplace.

Screenshots of UI which is showcasing my work

1- Product listing dynamic page(next JS) 👇



2- Product details page using dynamic routing (Next JS) 👇



Ceramics Chairs Register About Cart

Italian Chair

$419

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

Back to home

3-Product cart page dynamic (Next JS)👇

Avion

Ceramics Chairs Register About Cart

The Poplar suede sofa

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

Price: $250

Remove Checkout

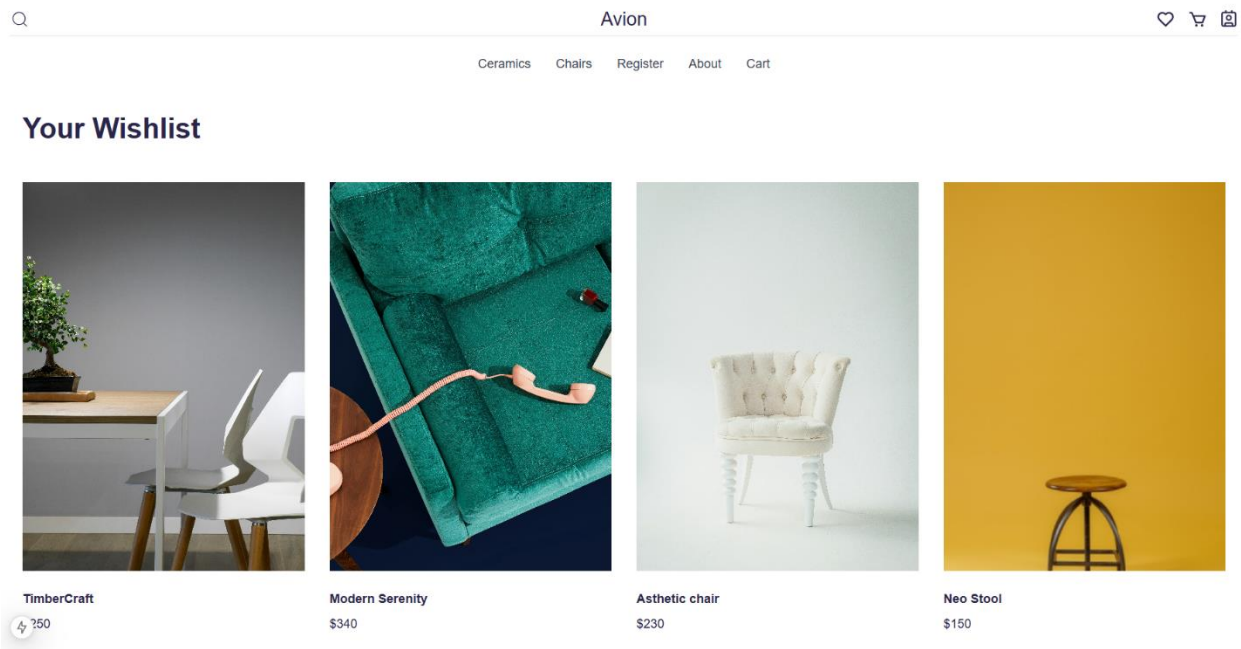
Handmade upholstery

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

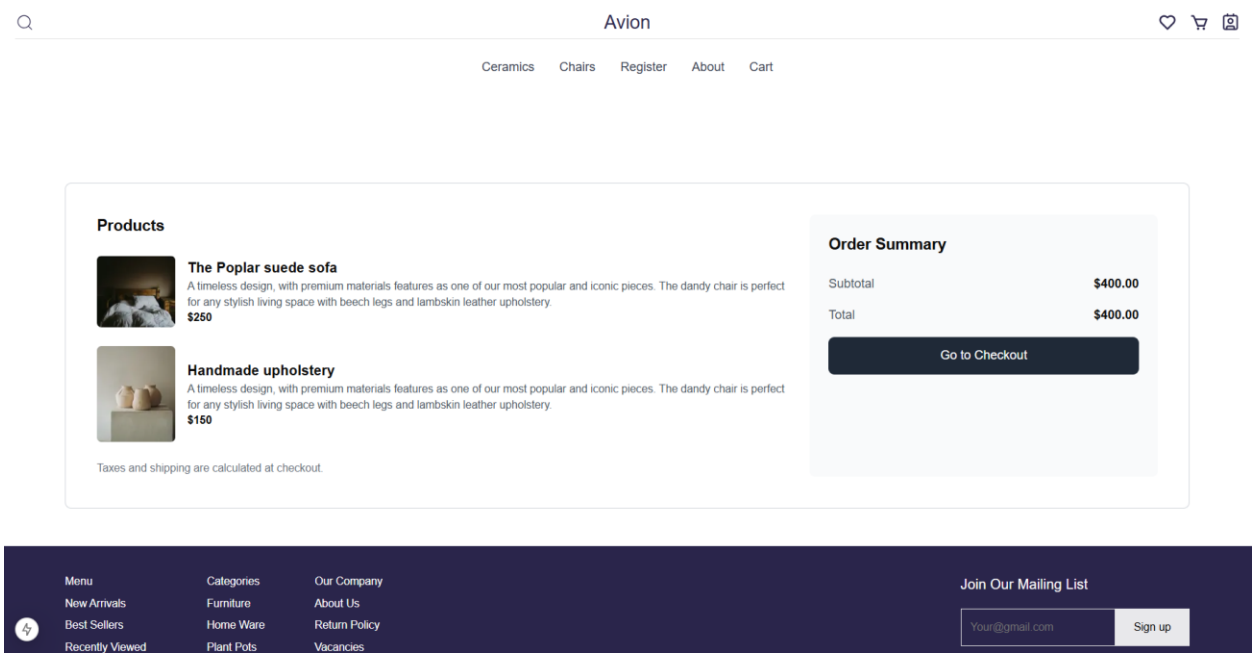
Price: $150

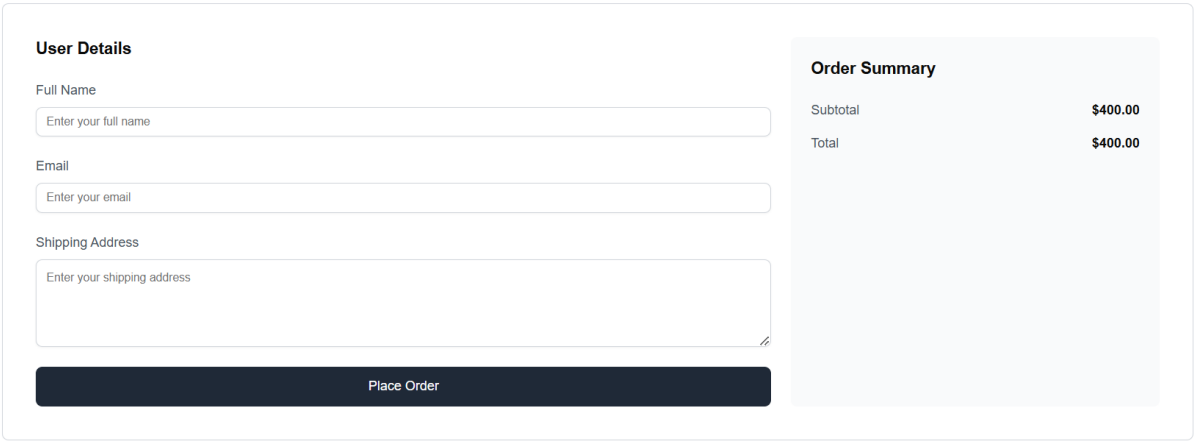
Remove Checkout

4-Product wishlist page dynamic (Next JS)👇



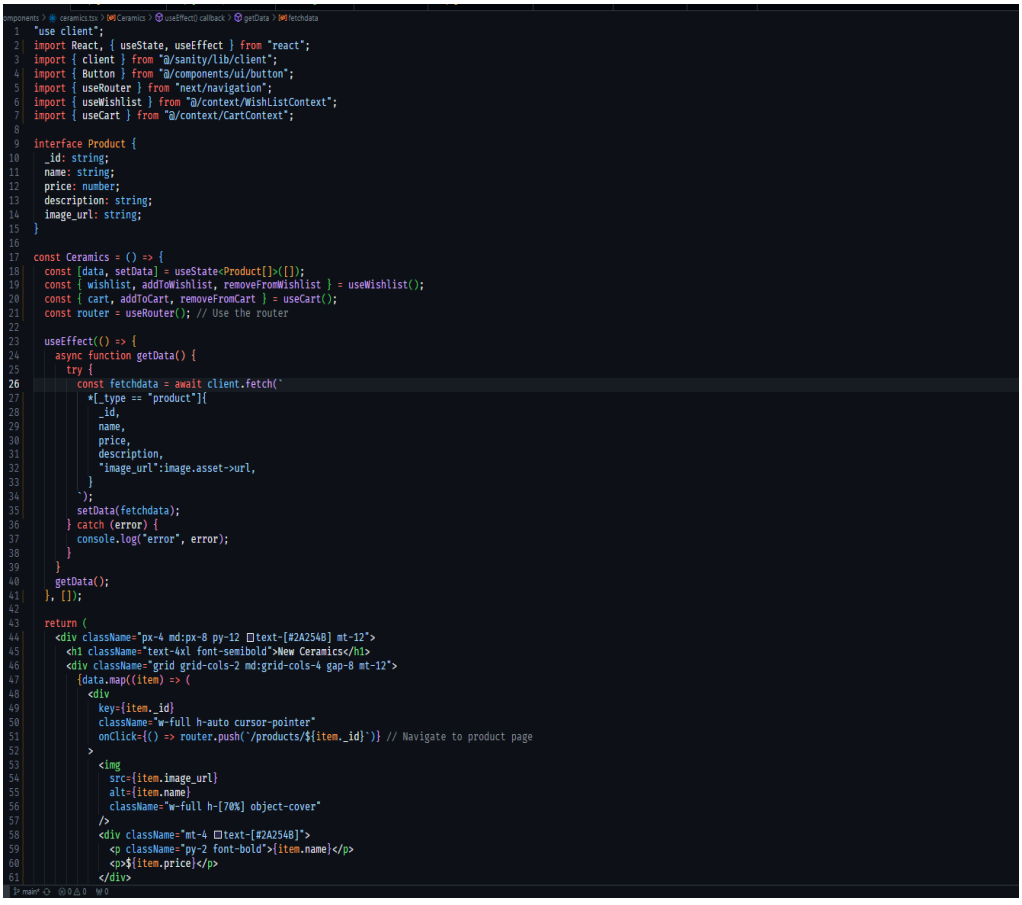
5-Product checkout page dynamic (Next JS)👇





Screenshots of code

1- Product list code 👇

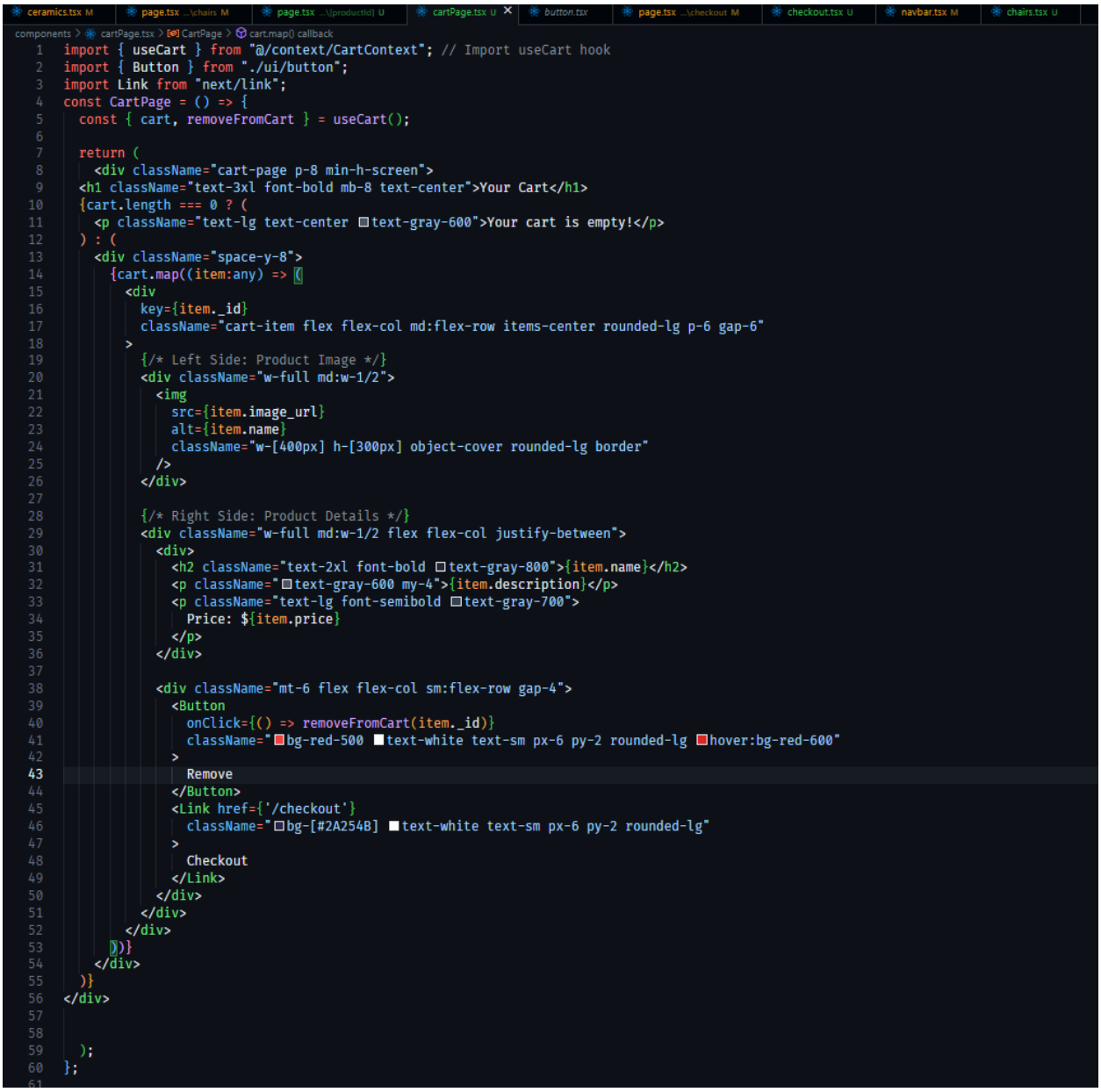


```tsx
"use client";
import React, { useState, useEffect } from "react";
import { client } from "@/sanity/lib/client";
import { Button } from "@/components/ui/button";
import { useRouter } from "next/navigation";
import { useWishlist } from "@/context/WishListContext";
import { useCart } from "@/context/CartContext";

interface Product {
  _id: string;
  name: string;
  price: number;
  description: string;
  image_url: string;
}

const Ceramics = () => {
  const [data, setData] = useState<Product[]>([]);
  const { wishlist, addToWishlist, removeFromWishlist } = useWishlist();
  const { cart, addToCart, removeFromCart } = useCart();
  const router = useRouter(); // Use the router

  useEffect(() => {
    async function getData() {
      try {
        const fetchdata = await client.fetch(`
          *[_type == "product"]{
            _id,
            name,
            price,
            description,
            "image_url":image.asset->url,
          }
        `);
        setData(fetchdata);
      } catch (error) {
        console.log("error", error);
      }
    }
    getData();
  }, []);

  return (
    <div className="px-4 md:px-8 py-12 text-[#2A254B] mt-12">
      <h1 className="text-4xl font-semibold">New Ceramics</h1>
      <div className="grid grid-cols-2 md:grid-cols-4 gap-8 mt-12">
        {data.map((item) => (
          <div
            key={item._id}
            className="w-full h-auto cursor-pointer"
            onClick={() => router.push(`/products/${item._id}`)} // Navigate to product page
          >
            <img
              src={item.image_url}
              alt={item.name}
              className="w-full h-[70%] object-cover"
            />
            <div className="mt-4 text-[#2A254B]">
              <p className="py-2 font-bold">{item.name}</p>
              <p>${item.price}</p>
            </div>
```

2- Product details code 👇

```tsx
import React from "react";
import { client } from "@/sanity/lib/client";
import Link from "next/link";

interface Product {
  _id: string;
  name: string;
  price: number;
  description: string;
  image_url: string;
}

const ProductsDescription = async ({ params }: { params: { productId: string } }) => {
  const productId = params.productId;

  // Fetch product details based on the productId
  const product: Product = await client.fetch(
    `*[_type == "product" && _id == $productId][0]{
      _id,
      name,
      price,
      description,
      "image_url":image.asset->url,
    }`,
    { productId }
  );

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

  return (
    <div className="p-8 flex flex-col items-center justify-center gap-9 md:flex-row md:gap-40 ">

      <div>
      <img
        src={product.image_url}
        alt={product.name}
        width={200}
        height={200}
        className="my-4 w-full max-w-md object-cover rounded-md"
      />
      </div>

     <div className="max-w-md">
     <h1 className="text-3xl md:text-4xl font-bold">{product.name}</h1>
      <p className="md:text-2xl text-lg my-3">${product.price}</p>
      <p className="my-4 text-md leading-6">{product.description}</p>
      <Link href={'/'} className="mt-4 bg-blue-500 text-white px-4 py-2 rounded text-sm md:text-lg">
        Back to home
      </Link>
     </div>

    </div>
  );
};

export default ProductsDescription;
```

3- Add to cart code👇



```tsx
import { useCart } from "@/context/CartContext"; // Import useCart hook
import { Button } from "./ui/button";
import Link from "next/link";
const CartPage = () => {
  const { cart, removeFromCart } = useCart();

  return (
    <div className="cart-page p-8 min-h-screen">
  <h1 className="text-3xl font-bold mb-8 text-center">Your Cart</h1>
  {cart.length === 0 ? (
    <p className="text-lg text-center text-gray-600">Your cart is empty!</p>
  ) : (
    <div className="space-y-8">
      {cart.map((item:any) => (
        <div
          key={item._id}
          className="cart-item flex flex-col md:flex-row items-center rounded-lg p-6 gap-6"
        >
          {/* Left Side: Product Image */}
          <div className="w-full md:w-1/2">
            <img
              src={item.image_url}
              alt={item.name}
              className="w-[400px] h-[300px] object-cover rounded-lg border"
            />
          </div>

          {/* Right Side: Product Details */}
          <div className="w-full md:w-1/2 flex flex-col justify-between">
            <div>
              <h2 className="text-2xl font-bold text-gray-800">{item.name}</h2>
              <p className="text-gray-600 my-4">{item.description}</p>
              <p className="text-lg font-semibold text-gray-700">
                Price: ${item.price}
              </p>
            </div>

            <div className="mt-6 flex flex-col sm:flex-row gap-4">
              <Button
                onClick={() => removeFromCart(item._id)}
                className="bg-red-500 text-white text-sm px-6 py-2 rounded-lg hover:bg-red-600"
              >
                Remove
              </Button>
              <Link href={'/checkout'}
                className="bg-[#2A254B] text-white text-sm px-6 py-2 rounded-lg"
              >
                Checkout
              </Link>
            </div>
          </div>
        </div>
      ))}
    </div>
  )}
</div>


  );
};
```

4- Add to wishlist code👇

components > wishlist.tsx > ...

```tsx
"use client";
import React from "react";

interface Product {
  _id: string;
  name: string;
  price: number;
  image_url: string;
}

interface WishlistProps {
  wishlist?: Product[]; // Make wishlist optional
}

const Wishlist = ({ wishlist = [] }: WishlistProps) => {
  return (
    <div className="px-4 md:px-8 py-12 text-[#2A254B] mt-12">
      <h1 className="text-4xl font-semibold">Your Wishlist</h1>

      <div className="grid grid-cols-2 md:grid-cols-4 gap-8 mt-12">
        {wishlist.length > 0 ? (
          wishlist.map((item) => (
            <div key={item._id} className="w-full h-auto">
              <img
                src={item.image_url}
                alt={item.name}
                className="w-full h-[80%] object-cover"
              />
              <div className="mt-4 text-[#2A254B]">
                <p className="py-2 font-bold">{item.name}</p>
                <p>${item.price}</p>
              </div>
            </div>
          ))
        ) : (
          <p className="text-gray-500">Your wishlist is empty</p>
        )}
      </div>
    </div>
  );
};

export default Wishlist;

```

5-Checkout component part 1👇

components > checkout.tsx > ...

```tsx
'use client';

import React, { useState } from "react";
import Image from "next/image";
import { useCart } from "@/context/CartContext";
import { Input } from "./ui/input";
import { Textarea } from "./ui/textarea";
const Checkout = () => {
  const { cart } = useCart(); // Access cart from context
  const [isCheckout, setIsCheckout] = useState(false); // Toggle between product list and form

  // Calculate subtotal
  const subtotal = cart.reduce((sum, item) => sum + item.price, 0);

  // Render user details form
  const renderCheckoutForm = () => (
    <div className="w-full lg:w-2/3">
      <h1 className="text-xl font-semibold mb-6">User Details</h1>
      <form className="space-y-6">
        <div>
          <label htmlFor="name" className="block text-gray-600 mb-2">
            Full Name
          </label>
          <Input
            type="text"
            id="name"
            className="w-full border border-gray-300 rounded-lg p-3"
            placeholder="Enter your full name"
          />
        </div>
        <div>
          <label htmlFor="email" className="block text-gray-600 mb-2">
            Email
          </label>
          <Input
            type="email"
            id="email"
            className="w-full border border-gray-300 rounded-lg p-3"
            placeholder="Enter your email"
          />
        </div>
        <div>
          <label htmlFor="address" className="block text-gray-600 mb-2">
            Shipping Address
          </label>
          <Textarea
            id="address"
            className="w-full border border-gray-300 rounded-lg p-3"
            placeholder="Enter your shipping address"
            rows={4}
          />
        </div>
        <button
          type="submit"
          className="w-full bg-gray-800 text-white py-3 rounded-lg"
        >
          Place Order
        </button>
      </form>
    </div>
  );
```

Part 2👇

```tsx
const Checkout = () => {
  return (
    <section>
      <div className="py-12 mt-12">
        <div className="w-[90%] mx-auto border border-gray-300 rounded-lg p-6 lg:p-10">
          <div className="flex flex-col lg:flex-row justify-between gap-6">
            {/* Product List or Checkout Form */}
            {!isCheckout ? (
              <div className="w-full lg:w-2/3">
                <h1 className="text-xl font-semibold mb-6">Products</h1>

                {cart.map((item:any) => (
                  <div key={item._id} className="flex items-center gap-4 mb-6">
                    <Image
                      src={item.image_url}
                      width={100}
                      height={100}
                      alt={item.name}
                      className="rounded-md"
                    />
                    <div>
                      <h1 className="text-lg font-semibold">{item.name}</h1>
                      <p className="text-gray-600 text-sm">{item.description}</p>
                      <h1 className="text-sm font-bold">${item.price}</h1>
                    </div>
                  </div>
                ))}

                <p className="text-sm text-gray-500">
                  Taxes and shipping are calculated at checkout.
                </p>
              </div>
            ) : (
              renderCheckoutForm()
            )}

            {/* Summary Section */}
            <div className="w-full lg:w-1/3 bg-gray-50 rounded-lg p-6">
              <h1 className="text-xl font-semibold mb-6">Order Summary</h1>

              <div className="flex justify-between mb-4">
                <p className="text-gray-600">Subtotal</p>
                <p className="font-bold">${subtotal.toFixed(2)}</p>
              </div>

              <div className="flex justify-between mb-4">
                <p className="text-gray-600">Total</p>
                <p className="font-bold">${subtotal.toFixed(2)}</p>
              </div>

              {/* Checkout Button */}
              {!isCheckout && (
                <button
                  className="w-full bg-gray-800 text-white py-3 rounded-lg"
                  onClick={() => setIsCheckout(true)} // Toggle to checkout form
                >
                  Go to Checkout
                </button>
              )}
            </div>
          </div>
```

Day 4 Report of 7-Day Hackathon

Objective

On Day 4 of the hackathon, the focus was on building and making key components dynamic for the e-commerce platform. The components developed include:

1. Dynamic Product Listing
2. Add to Cart Functionality
3. Add to Wishlist Functionality
4. Product Details Page
5. Checkout Page
6. Search Filter Integration

And more…

Development Steps

1. Building Dynamic Product Listing

- Implemented a dynamic product listing page using data fetched from the Sanity CMS.
- Designed the UI grid layout with Tailwind CSS for responsive behavior across devices.
- Added useRouter/Link from Next.js to enable navigation to individual product details pages.
- Each product card dynamically renders the product image, name, price, and action buttons (add to cart, wishlist).

2. Add to Cart Functionality

- Utilized a custom CartContext for managing cart state globally.
- Developed "Add to Cart" and "Remove from Cart" buttons to allow users to manage their selections dynamically.
- Integrated logic to update the cart in real-time and show notifications on adding/removing products.

3. Add to Wishlist Functionality

- Built a WishlistContext to store items users wish to save for later.
- Added a toggle button to switch items between the wishlist and the main product listing.
- Ensured the wishlist persists using local storage.

4. Product Details Page

- Used dynamic routing in Next.js to create unique pages for each product based on their ID.
- Fetched detailed product data (name, price, description, and images) from Sanity CMS.
- Styled the page to ensure it highlights the product information with a clean and user-friendly design.

5. Checkout Page

- Created a summary section for all items in the cart.
- Dynamically calculated the total cost of items, including subtotal and taxes.

- Added a "Proceed to Checkout" button, which leads to a user detail form for order submission.

6. Search Filter Integration (in progress).

Challenges and Solutions

Challenge 1: Managing Global States for Cart and Wishlist

- **Issue:** Handling global states for cart and wishlist while ensuring updates reflect across all pages.
- **Solution:** Used React Context API to create centralized states for both cart and wishlist. This allowed seamless updates and access across components.

Challenge 2: Dynamic Routing for Product Details

- **Issue:** Dynamically generating pages for individual products using product IDs.
- **Solution:** Leveraged Next.js dynamic routing ([productId]/page.tsx) and fetched data from Sanity CMS based on the route parameter.

Best Practices Followed

1. **Code Reusability:**
    - Created reusable components for product cards, buttons, and input fields to maintain consistency and reduce code redundancy.

2. **Responsive Design:**
    - Used Tailwind CSS to ensure all components were mobile-friendly and optimized for various screen sizes.
3. **Separation of Concerns:**
    - Kept data fetching logic in a separate service layer to make components cleaner and easier to manage.
4. **Error Handling:**
    - Added proper error handling for API calls to show user-friendly messages in case of failures.
5. **User Experience (UX):**
    - Implemented loading indicators and notifications for actions like adding to the cart or wishlist.

6. Conclusion

Day 4 was a productive and challenging day focused on making the core components of the e-commerce platform dynamic. By addressing the challenges effectively and adhering to best practices, I was able to build a user-friendly and functional interface. The groundwork laid today will enhance scalability and user satisfaction as we move forward in the hackathon.

THE END