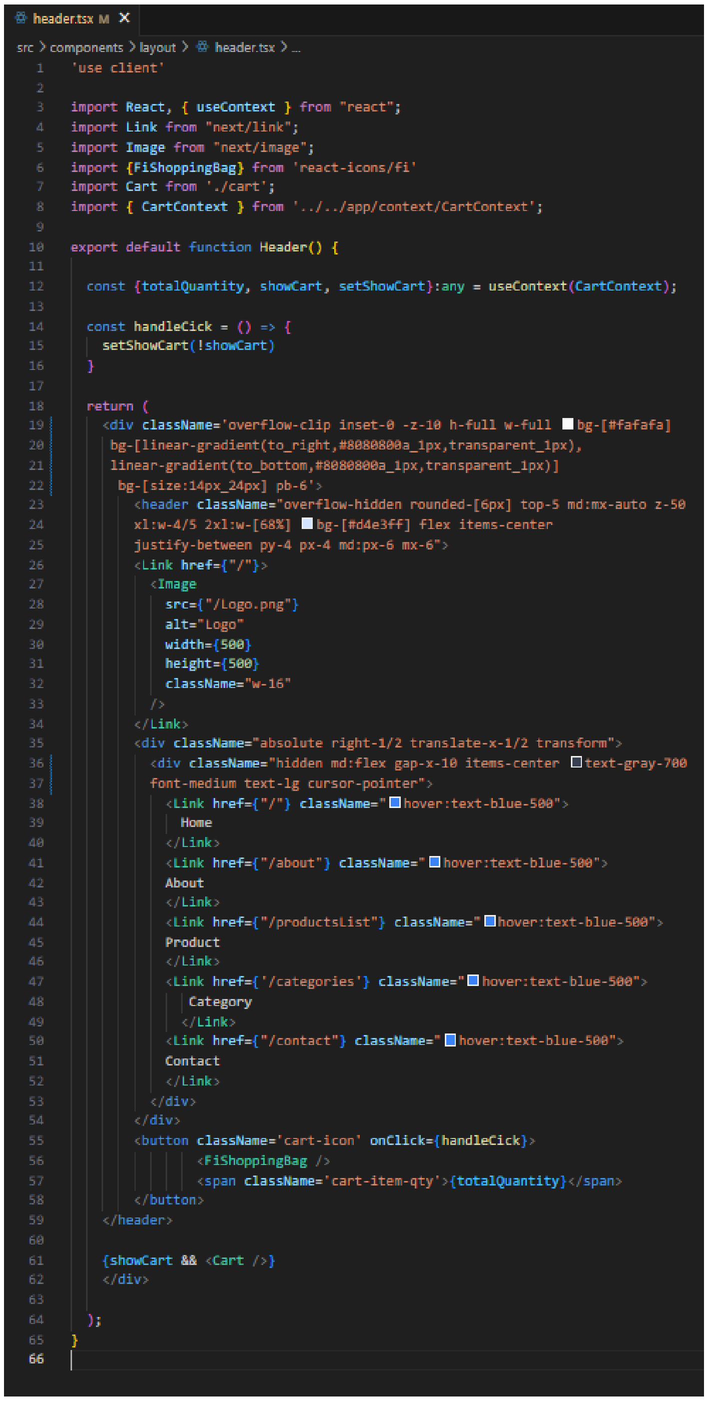
Day-4: “Dynamic Frontend Components For MarketPlace (Beadedwear)”

In this document I have described the libraries used for the implementation of my frontend components

Components:

1. Header
2. Footer
3. Contact Form
4. Cart
5. Category related Product List
6. Loader

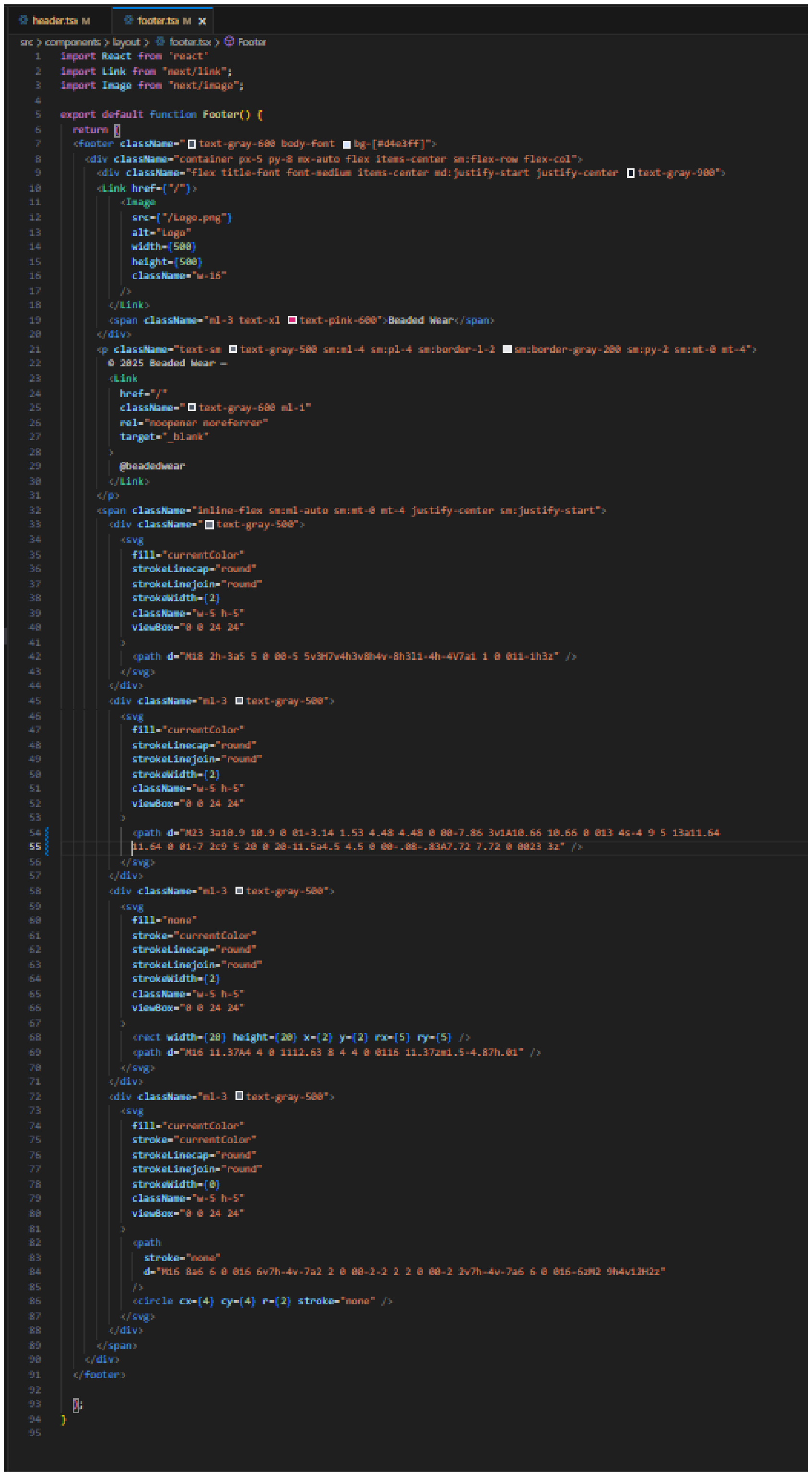
- **Header:**


header.tsx M

src > components > layout > header.tsx > ...

```tsx
'use client'

import React, { useContext } from "react";
import Link from "next/link";
import Image from "next/image";
import {FiShoppingBag} from 'react-icons/fi'
import Cart from './cart';
import { CartContext } from '../../app/context/CartContext';

export default function Header() {

  const {totalQuantity, showCart, setShowCart}:any = useContext(CartContext);

  const handleCick = () => {
    setShowCart(!showCart)
  }

  return (
    <div className='overflow-clip inset-0 -z-10 h-full w-full bg-[#fafafa]
     bg-[linear-gradient(to_right,#8080800a_1px,transparent_1px),
     linear-gradient(to_bottom,#8080800a_1px,transparent_1px)]
      bg-[size:14px_24px] pb-6'>
        <header className="overflow-hidden rounded-[6px] top-5 md:mx-auto z-50
        xl:w-4/5 2xl:w-[68%] bg-[#d4e3ff] flex items-center
        justify-between py-4 px-4 md:px-6 mx-6">
        <Link href={"/"}>
          <Image
            src={"/Logo.png"}
            alt="Logo"
            width={500}
            height={500}
            className="w-16"
          />
        </Link>
        <div className="absolute right-1/2 translate-x-1/2 transform">
          <div className="hidden md:flex gap-x-10 items-center text-gray-700
          font-medium text-lg cursor-pointer">
            <Link href={"/"} className="hover:text-blue-500">
              Home
            </Link>
            <Link href={"/about"} className="hover:text-blue-500">
            About
            </Link>
            <Link href={"/productsList"} className="hover:text-blue-500">
            Product
            </Link>
            <Link href={'/categories'} className="hover:text-blue-500">
               Category
              </Link>
            <Link href={"/contact"} className="hover:text-blue-500">
            Contact
            </Link>
          </div>
        </div>
        <button className='cart-icon' onClick={handleCick}>
                <FiShoppingBag />
                <span className='cart-item-qty'>{totalQuantity}</span>
        </button>
    </header>

    {showCart && <Cart />}
    </div>

  );
}

```

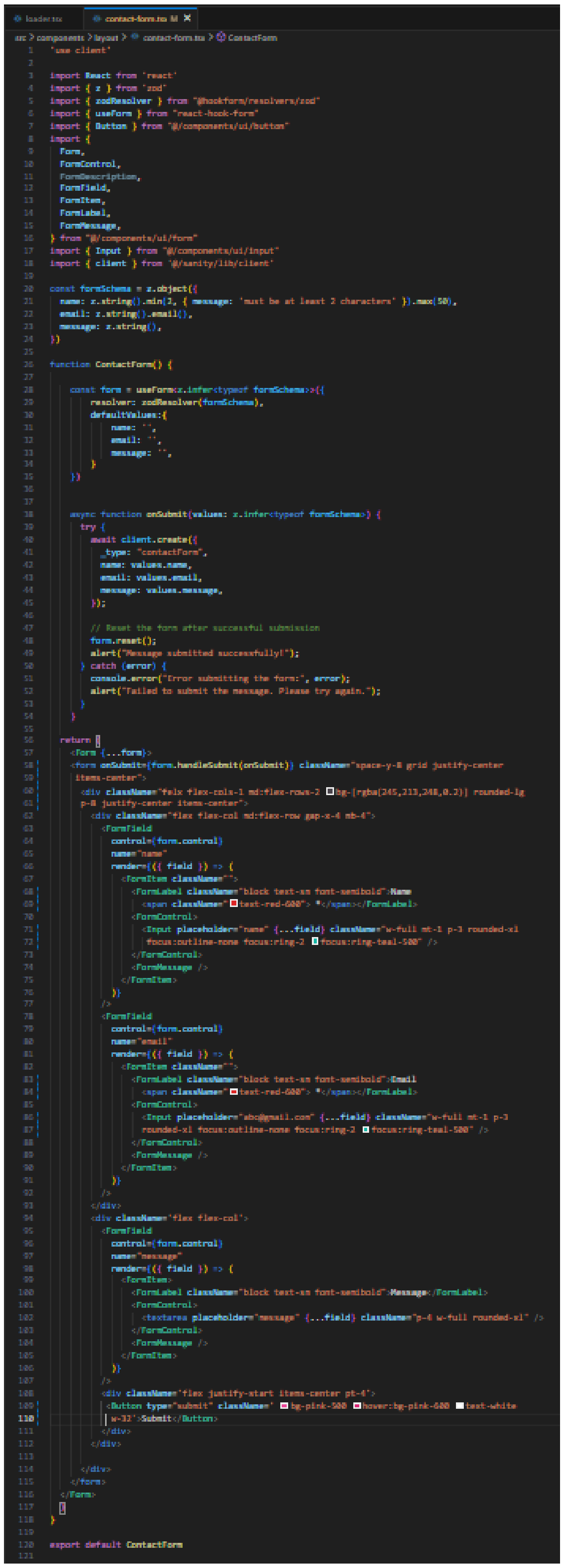
- **Footer:**



```tsx
import React from "react"
import Link from "next/link";
import Image from "next/image";

export default function Footer() {
  return (
  <footer className="text-gray-600 body-font bg-[#d4e3ff]">
    <div className="container px-5 py-8 mx-auto flex items-center sm:flex-row flex-col">
      <div className="flex title-font font-medium items-center md:justify-start justify-center text-gray-900">
      <Link href={"/"}>
          <Image
            src={"/Logo.png"}
            alt="Logo"
            width={500}
            height={500}
            className="w-16"
          />
        </Link>
        <span className="ml-3 text-xl text-pink-600">Beaded Wear</span>
      </div>
      <p className="text-sm text-gray-500 sm:ml-4 sm:pl-4 sm:border-l-2 sm:border-gray-200 sm:py-2 sm:mt-0 mt-4">
        © 2025 Beaded Wear —
        <Link
          href="/"
          className="text-gray-600 ml-1"
          rel="noopener noreferrer"
          target="_blank"
        >
          @beadedwear
        </Link>
      </p>
      <span className="inline-flex sm:ml-auto sm:mt-0 mt-4 justify-center sm:justify-start">
        <div className="text-gray-500">
          <svg
            fill="currentColor"
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={2}
            className="w-5 h-5"
            viewBox="0 0 24 24"
          >
            <path d="M18 2h-3a5 5 0 00-5 5v3H7v4h3v8h4v-8h3l1-4h-4V7a1 1 0 011-1h3z" />
          </svg>
        </div>
        <div className="ml-3 text-gray-500">
          <svg
            fill="currentColor"
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={2}
            className="w-5 h-5"
            viewBox="0 0 24 24"
          >
            <path d="M23 3a10.9 10.9 0 01-3.14 1.53 4.48 4.48 0 00-7.86 3v1A10.66 10.66 0 013 4s-4 9 5 13a11.64
            11.64 0 01-7 2c9 5 20 0 20-11.5a4.5 4.5 0 00-.08-.83A7.72 7.72 0 0023 3z" />
          </svg>
        </div>
        <div className="ml-3 text-gray-500">
          <svg
            fill="none"
            stroke="currentColor"
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={2}
            className="w-5 h-5"
            viewBox="0 0 24 24"
          >
            <rect width={20} height={20} x={2} y={2} rx={5} ry={5} />
            <path d="M16 11.37A4 4 0 1112.63 8 4 4 0 0116 11.37zm1.5-4.87h.01" />
          </svg>
        </div>
        <div className="ml-3 text-gray-500">
          <svg
            fill="currentColor"
            stroke="currentColor"
            strokeLinecap="round"
            strokeLinejoin="round"
            strokeWidth={0}
            className="w-5 h-5"
            viewBox="0 0 24 24"
          >
            <path
              stroke="none"
              d="M16 8a6 6 0 016 6v7h-4v-7a2 2 0 00-2-2 2 2 0 00-2 2v7h-4v-7a6 6 0 016-6zM2 9h4v12H2z"
            />
            <circle cx={4} cy={4} r={2} stroke="none" />
          </svg>
        </div>
      </span>
    </div>
  </footer>

  );
}
```

- **Conatct From:**

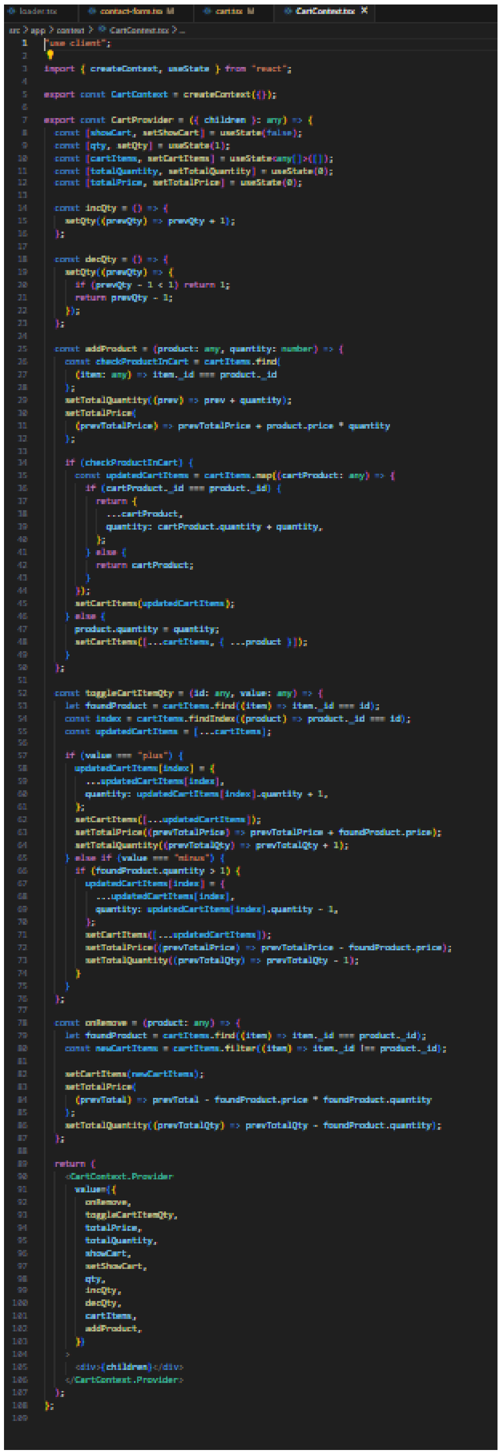
Used zod and react-hook-form for validations,



```tsx
'use client'

import React from 'react'
import { z } from 'zod'
import { zodResolver } from "@hookform/resolvers/zod"
import { useForm } from "react-hook-form"
import { Button } from "@/components/ui/button"
import {
  Form,
  FormControl,
  FormDescription,
  FormField,
  FormItem,
  FormLabel,
  FormMessage,
} from "@/components/ui/form"
import { Input } from "@/components/ui/input"
import { client } from '@/sanity/lib/client'

const formSchema = z.object({
  name: z.string().min(2, { message: 'must be at least 2 characters' }).max(50),
  email: z.string().email(),
  message: z.string(),
})

function ContactForm() {

    const form = useForm<z.infer<typeof formSchema>>({
        resolver: zodResolver(formSchema),
        defaultValues:{
            name: '',
            email: '',
            message: '',
        }
    })


    async function onSubmit(values: z.infer<typeof formSchema>) {
      try {
        await client.create({
          _type: "contactForm",
          name: values.name,
          email: values.email,
          message: values.message,
        });

        // Reset the form after successful submission
        form.reset();
        alert("Message submitted successfully!");
      } catch (error) {
        console.error("Error submitting the form:", error);
        alert("Failed to submit the message. Please try again.");
      }
    }

  return (
    <Form {...form}>
    <form onSubmit={form.handleSubmit(onSubmit)} className="space-y-8 grid justify-center
     items-center">
      <div className="felx flex-cols-1 md:flex-rows-2 bg-[rgba(245,213,248,0.2)] rounded-lg
      p-8 justify-center items-center">
        <div className="flex flex-col md:flex-row gap-x-4 mb-4">
          <FormField
            control={form.control}
            name="name"
            render={({ field }) => (
              <FormItem className="">
                <FormLabel className="block text-sm font-semibold">Name
                  <span className="text-red-600"> *</span></FormLabel>
                <FormControl>
                  <Input placeholder="name" {...field} className="w-full mt-1 p-3 rounded-xl
                   focus:outline-none focus:ring-2 focus:ring-teal-500" />
                </FormControl>
                <FormMessage />
              </FormItem>
            )}
          />
          <FormField
            control={form.control}
            name="email"
            render={({ field }) => (
              <FormItem className="">
                <FormLabel className="block text-sm font-semibold">Email
                  <span className="text-red-600"> *</span></FormLabel>
                <FormControl>
                  <Input placeholder="abc@gmail.com" {...field} className="w-full mt-1 p-3
                  rounded-xl focus:outline-none focus:ring-2 focus:ring-teal-500" />
                </FormControl>
                <FormMessage />
              </FormItem>
            )}
          />
        </div>
        <div className='flex flex-col'>
          <FormField
            control={form.control}
            name="message"
            render={({ field }) => (
              <FormItem>
                <FormLabel className="block text-sm font-semibold">Message</FormLabel>
                <FormControl>
                  <textarea placeholder="message" {...field} className="p-4 w-full rounded-xl" />
                </FormControl>
                <FormMessage />
              </FormItem>
            )}
          />
          <div className='flex justify-start items-center pt-4'>
            <Button type="submit" className=' bg-pink-500 hover:bg-pink-600 text-white
            w-32'>Submit</Button>
          </div>
        </div>

      </div>
    </form>
    </Form>
  )
}

export default ContactForm
```

- **Cart:**

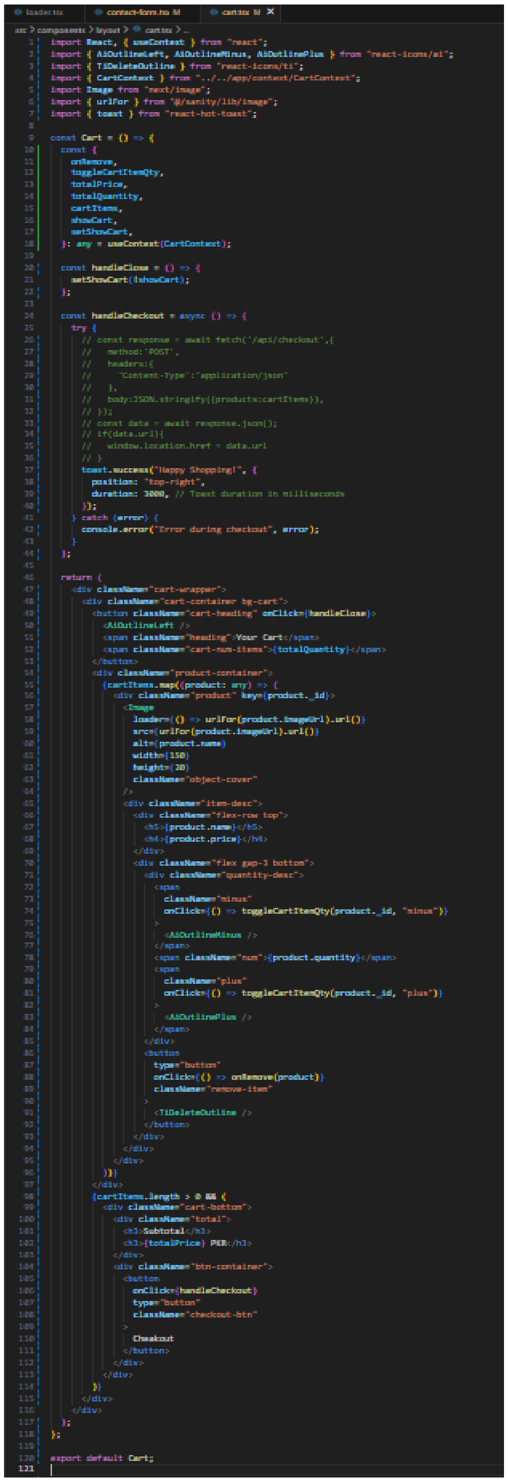
In the Cart I managed the real time changes using react states createContext and useState for quantity, price calculations , add and remove etc



loader.tsx   contact-form.tsx M   cart.tsx M   CartContext.tsx

src > app > context > CartContext.tsx > ...

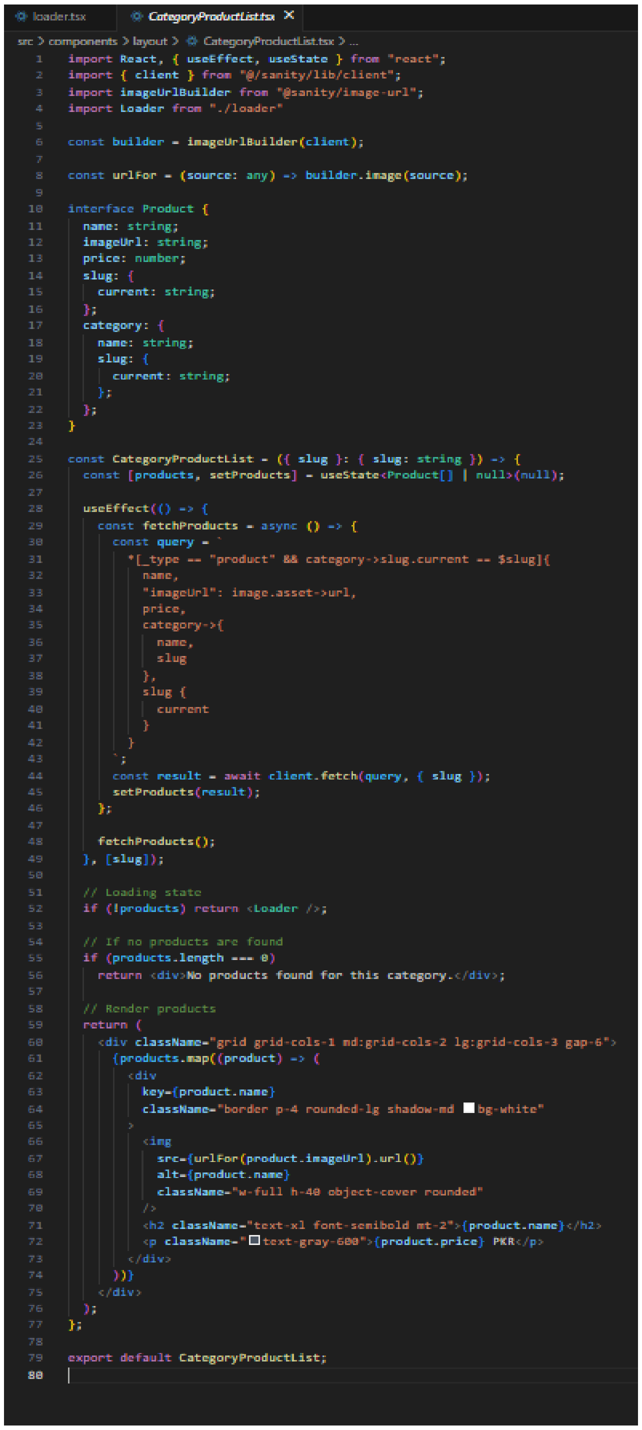
```tsx
"use client";

import { createContext, useState } from "react";

export const CartContext = createContext({});

export const CartProvider = ({ children }: any) => {
  const [showCart, setShowCart] = useState(false);
  const [qty, setQty] = useState(1);
  const [cartItems, setCartItems] = useState<any[]>([]);
  const [totalQuantity, setTotalQuantity] = useState(0);
  const [totalPrice, setTotalPrice] = useState(0);

  const incQty = () => {
    setQty((prevQty) => prevQty + 1);
  };

  const decQty = () => {
    setQty((prevQty) => {
      if (prevQty - 1 < 1) return 1;
      return prevQty - 1;
    });
  };

  const addProduct = (product: any, quantity: number) => {
    const checkProductInCart = cartItems.find(
      (item: any) => item._id === product._id
    );
    setTotalQuantity((prev) => prev + quantity);
    setTotalPrice(
      (prevTotalPrice) => prevTotalPrice + product.price * quantity
    );

    if (checkProductInCart) {
      const updatedCartItems = cartItems.map((cartProduct: any) => {
        if (cartProduct._id === product._id) {
          return {
            ...cartProduct,
            quantity: cartProduct.quantity + quantity,
          };
        } else {
          return cartProduct;
        }
      });
      setCartItems(updatedCartItems);
    } else {
      product.quantity = quantity;
      setCartItems([...cartItems, { ...product }]);
    }
  };

  const toggleCartItemQty = (id: any, value: any) => {
    let foundProduct = cartItems.find((item) => item._id === id);
    const index = cartItems.findIndex((product) => product._id === id);
    const updatedCartItems = [...cartItems];

    if (value === "plus") {
      updatedCartItems[index] = {
        ...updatedCartItems[index],
        quantity: updatedCartItems[index].quantity + 1,
      };
      setCartItems([...updatedCartItems]);
      setTotalPrice((prevTotalPrice) => prevTotalPrice + foundProduct.price);
      setTotalQuantity((prevTotalQty) => prevTotalQty + 1);
    } else if (value === "minus") {
      if (foundProduct.quantity > 1) {
        updatedCartItems[index] = {
          ...updatedCartItems[index],
          quantity: updatedCartItems[index].quantity - 1,
        };
        setCartItems([ ...updatedCartItems]);
        setTotalPrice((prevTotalPrice) => prevTotalPrice - foundProduct.price);
        setTotalQuantity((prevTotalQty) => prevTotalQty - 1);
      }
    }
  };

  const onRemove = (product: any) => {
    let foundProduct = cartItems.find((item) => item._id === product._id);
    const newCartItems = cartItems.filter((item) => item._id !== product._id);

    setCartItems(newCartItems);
    setTotalPrice(
      (prevTotal) => prevTotal - foundProduct.price * foundProduct.quantity
    );
    setTotalQuantity((prevTotalQty) => prevTotalQty - foundProduct.quantity);
  };

  return (
    <CartContext.Provider
      value={{
        onRemove,
        toggleCartItemQty,
        totalPrice,
        totalQuantity,
        showCart,
        setShowCart,
        qty,
        incQty,
        decQty,
        cartItems,
        addProduct,
      }}
    >
      <div>{children}</div>
    </CartContext.Provider>
  );
};
```



```tsx
import React, { useContext } from "react";
import { AiOutlineLeft, AiOutlineMinus, AiOutlinePlus } from "react-icons/ai";
import { TiDeleteOutline } from "react-icons/ti";
import { CartContext } from "../../app/context/CartContext";
import Image from "next/image";
import { urlFor } from "@/sanity/lib/image";
import { toast } from "react-hot-toast";

const Cart = () => {
  const {
    onRemove,
    toggleCartItemQty,
    totalPrice,
    totalQuantity,
    cartItems,
    showCart,
    setShowCart,
  }: any = useContext(CartContext);

  const handleClose = () => {
    setShowCart(!showCart);
  };

  const handleCheckout = async () => {
    try {
      // const response = await fetch('/api/checkout',{
      //   method:'POST',
      //   headers:{
      //     "Content-Type":"application/json"
      //   },
      //   body:JSON.stringify({products:cartItems}),
      // });
      // const data = await response.json();
      // if(data.url){
      //   window.location.href = data.url
      // }
      toast.success("Happy Shopping!", {
        position: "top-right",
        duration: 3000, // Toast duration in milliseconds
      });
    } catch (error) {
      console.error("Error during checkout", error);
    }
  };

  return (
    <div className="cart-wrapper">
      <div className="cart-container bg-cart">
        <button className="cart-heading" onClick={handleClose}>
          <AiOutlineLeft />
          <span className="heading">Your Cart</span>
          <span className="cart-num-items">{totalQuantity}</span>
        </button>
        <div className="product-container">
          {cartItems.map((product: any) => (
            <div className="product" key={product._id}>
              <Image
                loader={() => urlFor(product.imageUrl).url()}
                src={urlFor(product.imageUrl).url()}
                alt={product.name}
                width={150}
                height={20}
                className="object-cover"
              />
              <div className="item-desc">
                <div className="flex-row top">
                  <h5>{product.name}</h5>
                  <h4>{product.price}</h4>
                </div>
                <div className="flex gap-3 bottom">
                  <div className="quantity-desc">
                    <span
                      className="minus"
                      onClick={() => toggleCartItemQty(product._id, "minus")}
                    >
                      <AiOutlineMinus />
                    </span>
                    <span className="num">{product.quantity}</span>
                    <span
                      className="plus"
                      onClick={() => toggleCartItemQty(product._id, "plus")}
                    >
                      <AiOutlinePlus />
                    </span>
                  </div>
                  <button
                    type="button"
                    onClick={() => onRemove(product)}
                    className="remove-item"
                  >
                    <TiDeleteOutline />
                  </button>
                </div>
              </div>
            </div>
          ))}
        </div>
        {cartItems.length > 0 && (
          <div className="cart-bottom">
            <div className="total">
              <h3>Subtotal</h3>
              <h3>{totalPrice} PKR</h3>
            </div>
            <div className="btn-container">
              <button
                onClick={handleCheckout}
                type="button"
                className="checkout-btn"
              >
                Checkout
              </button>
            </div>
          </div>
        )}
      </div>
    </div>
  );
};

export default Cart;
```

- **Category Product:**

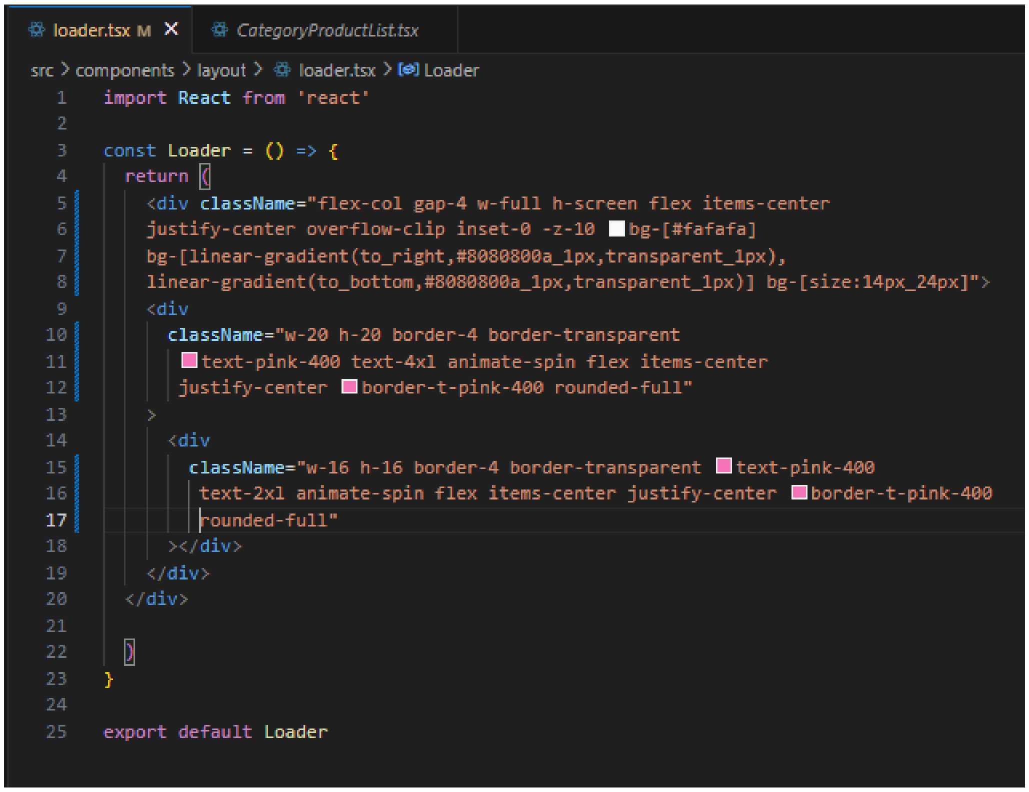
Made dynamic categories page and fetched products for the desired category from sanity with state management using react hooks (useState and useEffect),



```tsx
import React, { useEffect, useState } from "react";
import { client } from "@/sanity/lib/client";
import imageUrlBuilder from "@sanity/image-url";
import Loader from "./loader"

const builder = imageUrlBuilder(client);

const urlFor = (source: any) => builder.image(source);

interface Product {
  name: string;
  imageUrl: string;
  price: number;
  slug: {
    current: string;
  };
  category: {
    name: string;
    slug: {
      current: string;
    };
  };
}

const CategoryProductList = ({ slug }: { slug: string }) => {
  const [products, setProducts] = useState<Product[] | null>(null);

  useEffect(() => {
    const fetchProducts = async () => {
      const query = `
        *[_type == "product" && category->slug.current == $slug]{
          name,
          "imageUrl": image.asset->url,
          price,
          category->{
            name,
            slug
          },
          slug {
            current
          }
        }
      `;
      const result = await client.fetch(query, { slug });
      setProducts(result);
    };

    fetchProducts();
  }, [slug]);

  // Loading state
  if (!products) return <Loader />;

  // If no products are found
  if (products.length === 0)
    return <div>No products found for this category.</div>;

  // Render products
  return (
    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
      {products.map((product) => (
        <div
          key={product.name}
          className="border p-4 rounded-lg shadow-md bg-white"
        >
          <img
            src={urlFor(product.imageUrl).url()}
            alt={product.name}
            className="w-full h-40 object-cover rounded"
          />
          <h2 className="text-xl font-semibold mt-2">{product.name}</h2>
          <p className="text-gray-600">{product.price} PKR</p>
        </div>
      ))}
    </div>
  );
};

export default CategoryProductList;
```

- **Loader:**

Made a customize Loader Component for interactive user interface,



```tsx
import React from 'react'

const Loader = () => {
  return (
    <div className="flex-col gap-4 w-full h-screen flex items-center
    justify-center overflow-clip inset-0 -z-10 bg-[#fafafa]
    bg-[linear-gradient(to_right,#8080800a_1px,transparent_1px),
    linear-gradient(to_bottom,#8080800a_1px,transparent_1px)] bg-[size:14px_24px]">
    <div
      className="w-20 h-20 border-4 border-transparent
       text-pink-400 text-4xl animate-spin flex items-center
       justify-center border-t-pink-400 rounded-full"
    >
      <div
        className="w-16 h-16 border-4 border-transparent text-pink-400
         text-2xl animate-spin flex items-center justify-center border-t-pink-400
         rounded-full"
      ></div>
    </div>
  </div>

  )
}

export default Loader
```

Challenges And Solutions:

It was hard to update the cart dynamically using props as there were multiple things being updated at the same time like change in the total price, add and subtract quatity from cart directly moreover when to show the cart ..etc

Here I do my research and came across to a react function createContext(), In React the createContext() function is used to create a Context that enables sharing values (like data or functions) between components without having to pass props down manually at every level. This is especially useful for managing global state or shared values.

Document by Ilsa Ubaid(**senior student GIAIC**)