DAY 4 - DYNAMIC FRONT-END COMPONENTS

SHOP.CO

OVERVIEW:

- **On Day 4, I implemented key features to enhance the functionality of my e-commerce website, Shop.co, ensuring a dynamic and user-friendly experience.**
- **Below are the details of the work completed:**

FEATURES IMPLEMENTED

01 • Product Listing Page

02 • Product Detail Page

03 • Add to Cart Functionality with Redux

04 • Pagination with ShadCN UI

TECHNICAL WORKFLOW

Product Listing:

- Fetched dynamic data from Sanity CMS to populate the product grid.
- Integrated category filters, a search bar, and pagination to allow users to browse and locate products seamlessly.

Product Detail Page:

- Implemented individual product detail pages with accurate dynamic routing using Next.Js.
- Each detail page renders data dynamically from Sanity CMS.

Category Page:

- Dynamically fetched products based on their categories using Sanity CMS.
- Rendered category-specific product listings with pagination for better user navigation.

TECHNICAL WORKFLOW

Add to Cart Workflow:

- Developed a slice using Redux Toolkit to manage cart state.
- Configured store.ts to include persistence middleware for saving cart data in local storage.
- Connected the toast notification component to trigger upon successful cart actions.

Responsive Design:

- Thoroughly tested and adjusted the layout for different screen sizes, ensuring consistent functionality and aesthetics.

Toastify Notifications:

- Created a client-rendered component to manage toast notifications efficiently.
- Added success messages to confirm user actions, such as adding items to the cart.

BEST PRACTICES FOLLOWED

- Ensured code modularity by separating concerns into dedicated folders and files.
- Followed clean code principles, making the project easy to maintain and extend.
- Implemented efficient state management using Redux Toolkit and redux-persist.
- Tested the application thoroughly to ensure responsiveness and functionality across devices.

Product Listing Page(Dynamic Routing)

```tsx
"use client"
import { Button } from "@/components/ui/button";
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";
import Link from "next/link";
import { FaStar } from "react-icons/fa";
import { useState, useEffect } from "react";

interface Iproducts {
  imageUrl: string;
  discountPercent: number;
  isNew: boolean;
  name: string;
  description: string;
  price: number;
  _id: string;
}

// Star icons array
const star = [
  <FaStar key={1} />,
  <FaStar key={2} />,
  <FaStar key={3} />,
  <FaStar key={4} />,
  <FaStar key={5} />,
];

export default function Products() {
  const [products, setProducts] = useState<Iproducts[]>([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    // Fetch products with error handling
    const fetchProducts = async () => {
      try {
        setLoading(true);
        setError(null);
        const fetchedProducts: Iproducts[] = await client.fetch(
          `*[_type == 'products']{
            "imageUrl": image.asset->url,
            category,
            discountPercent,
            isNew,
            name,
            description,
            price,
            _id
          }[0...4]`
        );
        setProducts(fetchedProducts);
      } catch (err: any) {
        setError("Failed to load products. Please try again later.");
        console.error("Error fetching products:", err);
      } finally {
        setLoading(false);
      }
    };
```

```tsx
fetchProducts();
  }, []);

  if (loading) {
    return (
      <div className="flex justify-center items-center h-screen">
        <p>Loading products...</p>
      </div>
    );
  }

  if (error) {
    return (
      <div className="flex justify-center items-center h-screen">
        <p className="text-red-500 font-bold">{error}</p>
      </div>
    );
  }

  return (
    <div className="w-full h-full mt-10 max-w-screen-xl mx-auto">
      <h1 className="text-3xl md:text-4xl font-bold text-center">TOP SELLING</h1>
      <div className="relative mt-10 overflow-x-auto flex space-x-5 px-8">
        {products.map((data) => (
          <div key={data._id} className="flex-shrink-0">
            <Link href={`/product/${data._id}`}>
              <div className="w-[200px] md:w-[283px] h-[200px] md:h-[290px] bg-[#F0EEED] rounded-[20px]">
                {data.imageUrl ? (
                  <Image
                    src={urlFor(data.imageUrl).url()}
                    alt={data.name}
                    className="w-full h-full rounded-[20px]"
                    width={100}
                    height={100}
                  />
                ) : (
                  <div className="w-full h-full flex justify-center items-center bg-gray-300 rounded-[20px]">
                    <p>No Image</p>
                  </div>
                )}
              </div>
            </Link>
            <div className="pl-2">
              <p className="text-lg mt-2 font-bold">{data.name}</p>
              <div className="flex text-yellow-400">
                {star.map((icon, index) => (
                  <span key={index}>{icon}</span>
                ))}
              </div>
              <p className="font-bold mt-1">
                ${data.price.toFixed(2)}
                {data.discountPercent ? (
                  <span className="text-gray-400 font-bold line-through ml-2">
                    {data.discountPercent}%
                  </span>
                ) : null}
              </p>
            </div>
          </div>
        ))}
      </div>
      <div className="flex justify-center items-start mt-5">
        <Link href="/casual">
          <Button
            variant={"outline"}
            className="sm:mt-0 w-[80%] sm:w-[200px] rounded-[20px]"
          >
            View all
          </Button>
        </Link>
      </div>
    </div>
  );
}
```

Redux Store

```
import { combineReducers, configureStore } from '@reduxjs/toolkit'
import  cartSlice  from './features/cart'
import storage  from 'redux-persist/lib/storage';
import persistReducer from 'redux-persist/lib/persistReducer';

const persistconfiq = {
  key:"root",
  version:1,
  storage,
}
const reducer = combineReducers({
  cart:cartSlice
})
const persistedReducer = persistReducer(persistconfiq,reducer)

export const store = configureStore({
  reducer:persistedReducer,
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware({serializableCheck:false}),
})

// Infer the `RootState`,  `AppDispatch`, and `AppStore` types from the store itself
export type RootState = ReturnType<typeof store.getState>
// Inferred type: {posts: PostsState, comments: CommentsState, users: UsersState}
export type AppDispatch = typeof store.dispatch
export type AppStore = typeof store
```

Redux Store

Redux Hooks

```
1  import { useDispatch, useSelector } from 'react-redux'
2  import type { AppDispatch, RootState } from './store'
3
4  // Use throughout your app instead of plain `useDispatch` and `useSelector`
5  export const useAppDispatch = useDispatch.withTypes<AppDispatch>()
6  export const useAppSelector = useSelector.withTypes<RootState>()
```

CartSlice

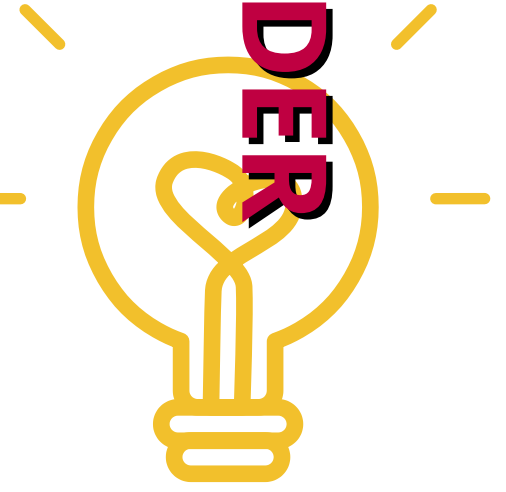
```
import { createSlice, PayloadAction } from '@reduxjs/toolkit'

// Define the initial state using that type
export const cartSlice = createSlice({
  name: 'products',
  // `createSlice` will infer the state type from the `initialState` argument
  initialState:[],
  reducers: {
    // add to cart functionalty
    add(state:any,action){
      let uuid = Math.floor(1000+Math.random()*9000)
      let newobj = {...action.payload,uuid}
      state.push(newobj)
     },
     // delete from cart
     remove(state:any,{payload}){
       return  state.filter((val:any)=> val.uuid !== payload)
     },
     //  addition of item
    addition(state:any,action){
      let obj = state.find(
        (val:any)=>
          val.id == action.payload.id &&
          val.color == action.payload.color &&
          val.size == action.payload.size
      );
      if(obj){
```

```
  ++obj.qty;
        let newState = state.filter((val:any)=> val.id !== obj.id);
        state = [...newState,obj];
        return
      }
    },
     //   subraction of item
     subraction(state:any,action){
      let obj = state.find(
        (val:any)=>
          val.id == action.payload.id &&
          val.color == action.payload.color &&
          val.size == action.payload.size
      );
      if(obj !== undefined){
        --obj.qty;
        let newState = state.filter((val:any)=> val.uuid !== obj.uuid);
        state = [...newState,obj];
        return;
      }
      // end
  }}
})

export const {add, remove,subraction,addition} = cartSlice.actions

export default cartSlice.reducer
```

Provider

```tsx
import { combineReducers, configureStore } from '@reduxjs/toolkit'
import  cartSlice  from './features/cart'
import storage  from 'redux-persist/lib/storage';
import persistReducer from 'redux-persist/lib/persistReducer';

const persistconfiq = {
  key:"root",
  version:1,
  storage,
}
const reducer = combineReducers({
  cart:cartSlice
})
const persistedReducer = persistReducer(persistconfiq,reducer)

export const store = configureStore({
  reducer:persistedReducer,
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware({serializableCheck:false}),
})

// Infer the `RootState`,  `AppDispatch`, and `AppStore` types from the store itself
export type RootState = ReturnType<typeof store.getState>
// Inferred type: {posts: PostsState, comments: CommentsState, users: UsersState}
export type AppDispatch = typeof store.dispatch
export type AppStore = typeof store
```

PROVIDER



Pagination & Toastify

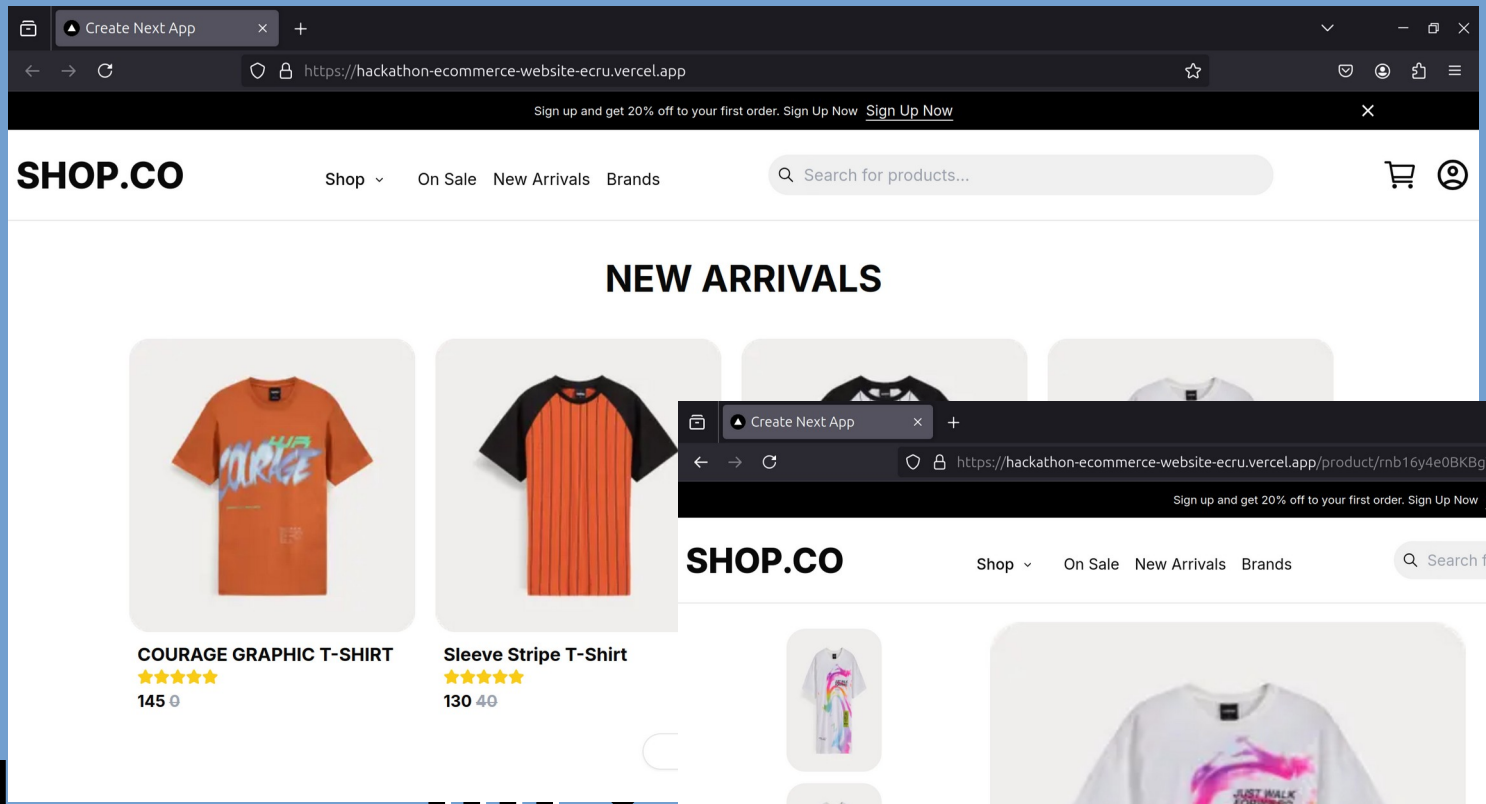
TOASTIFY

```tsx
"use client"
import React from 'react';
import { useDispatch } from 'react-redux';
import { Bounce, ToastContainer, toast } from 'react-toastify';
import "react-toastify/ReactToastify.css";
import { add } from '../Redux/features/cart';
import { Button } from '@/components/ui/button';


function Toastify({cartItem}:any) {

const dispatch = useDispatch()

const handleadd = (cartItem:any)=>{
    dispatch(add(cartItem))
}


    const notify = () =>
    toast.success('Product added Successfully!', {
      position: "bottom-right",
      autoClose: 5000,
      hideProgressBar: false,
      closeOnClick: false,
      pauseOnHover: true,
      draggable: true,
      progress: undefined,
      theme: "light",
      transition: Bounce,
      });
  return (
    <>
    <div  onClick={()=>handleadd(cartItem)}>
         <Button  onClick={notify} className="bg-black text-white lg:w-[300px]"
           >Add to Cart</Button>
    </div>
     <ToastContainer
     position="bottom-right"
     autoClose={5000}
     hideProgressBar={false}
     newestOnTop={false}
     closeOnClick={false}
     rtl={false}
     pauseOnFocusLoss
     draggable
     pauseOnHover
     theme="light"
     transition={Bounce}
     />
     </>

  )
}

export default Toastify
```

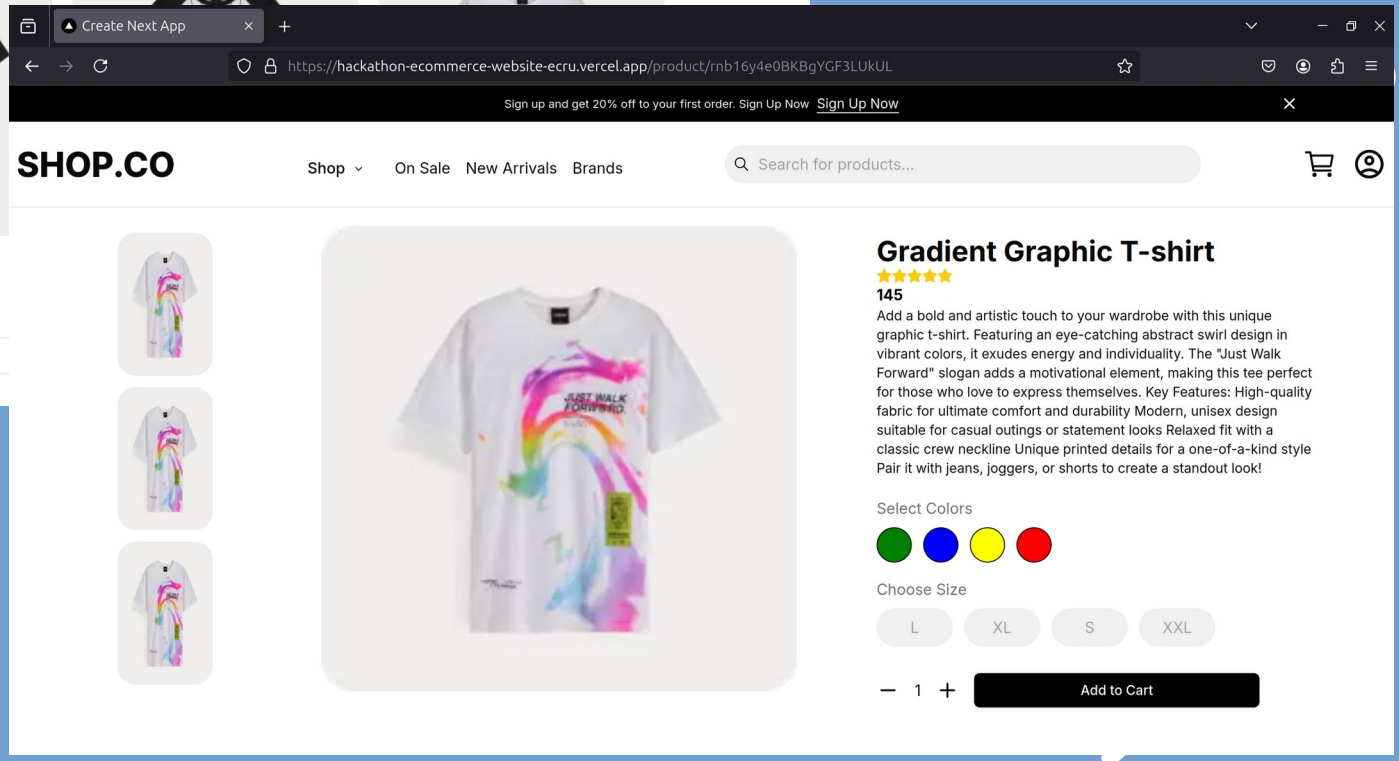
Pagination

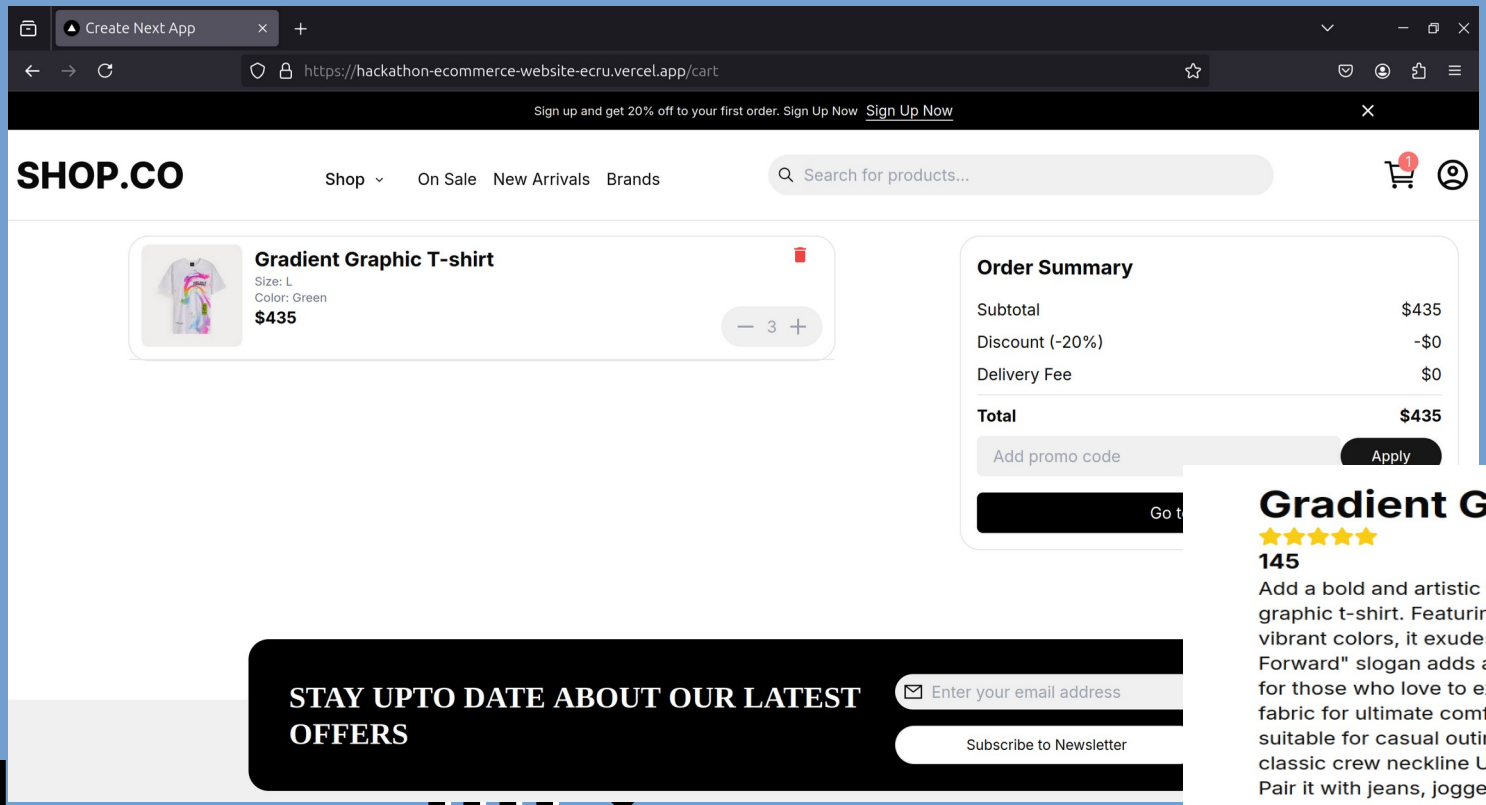
```tsx
import {
    Pagination,
    PaginationContent,
    PaginationEllipsis,
    PaginationItem,
    PaginationLink,
    PaginationNext,
    PaginationPrevious,
  } from "@/components/ui/pagination"
import React from 'react'

function Paginationpage() {
  return (
    <div className="w-full mt-4">
 <Pagination>
  <PaginationContent className="w-full space-x-4 flex justify-center lg:ml-7">
    <PaginationItem>
      <PaginationPrevious href="/" />
    </PaginationItem>

    <PaginationItem className="hidden lg:block">
      <PaginationLink href="/">Home</PaginationLink>
    </PaginationItem>
    <PaginationItem className="hidden lg:block">
      <PaginationEllipsis />
    </PaginationItem>

    <PaginationItem className="hidden lg:block">
      <PaginationLink href="/sell">Top Sell</PaginationLink>
    </PaginationItem>
    <PaginationItem className="hidden lg:block">
      <PaginationEllipsis />
    </PaginationItem>

    <PaginationItem>
      <PaginationLink href="/brand">Brands</PaginationLink>
    </PaginationItem>
    <PaginationItem>
      <PaginationEllipsis />
    </PaginationItem>

    <PaginationItem>
      <PaginationNext href="/cart" />
    </PaginationItem>
  </PaginationContent>
</Pagination>

    </div>
  )
}

export default Paginationpage
```

PRODUCTS LISTING

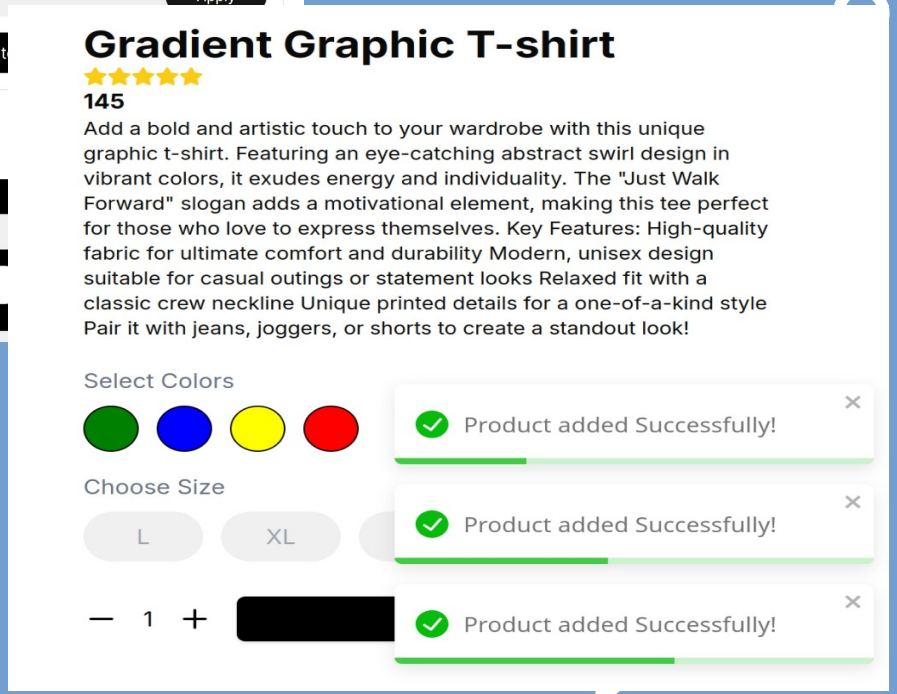


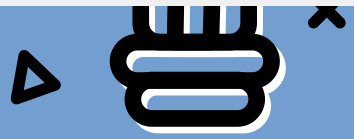
PRODUCT DETAILS





CART PAGE





TOASTIFY NOTIFICATION

THANK YOU