DAY 4 - DYNAMIC FRONTEND 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

Product Listing Page:

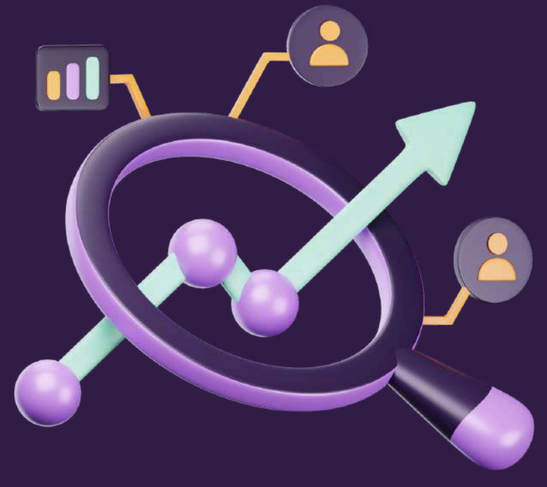
Product Detail Page:

Add to Cart Functionality with Redux:

Pagination with ShadCN UI:

Responsive Design with Tailwind css

TECHNICAL WORKFLOW



Product Listing Page Workflow

- 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.

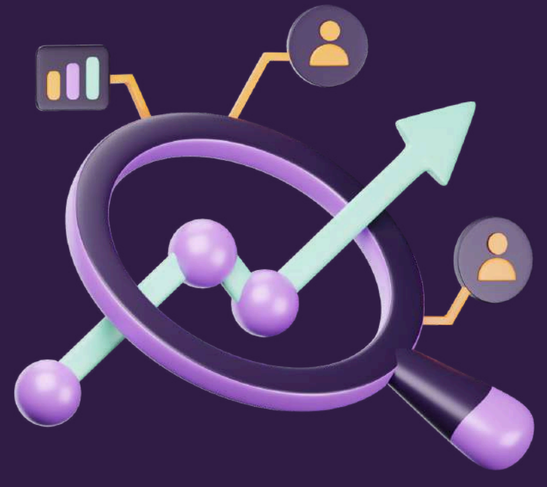
Category Page:

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

Product Detail Page:

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

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

```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 { useEffect, useState } from "react";

// Adding key prop in star array
let star = [
    <FaStar key={1} />,
    <FaStar key={2} />,
    <FaStar key={3} />,
    <FaStar key={4} />,
    <FaStar key={5} />,
];

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

export default function Product() {
    const [products, setProducts] = useState<Iproducts[]>([]);

    useEffect(() => {
        const fetchProducts = async () => {
            try {
                const fetchedProducts = await client.fetch(
                    `*[_type == 'products' && category == 'tshirt']{
                        "image": image.asset->url,
                        category,
                        discountPercent,
                        isNew,
                        name,
                        description,
                        price,
                        _id
                    }[0...4]`
                );
                setProducts(fetchedProducts);
            } catch (error) {
                console.error("Error fetching products:", error);
            }
        };

        fetchProducts();
    }, []);

    return (
        <>
            <div className="w-full h-full mt-10 lg:mt-36 max-w-screen-xl mx-auto">
                <h1 className="text-3xl md:text-4xl font-bold text-center">NEW ARRIVALS</h1>
                <div className="relative mt-10 overflow-x-auto flex space-x-5 px-8">
                    {products.length > 0 ? (
                        products.map((data) => {
                            return (
                                <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.image && (
                                                <Image
                                                    src={urlFor(data.image).url()}
                                                    alt={data.name}
                                                    className="w-full h-full rounded-[20px]"
                                                    width={100}
                                                    height={100}
                                                />
                                            )}
                                        </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}{" "}
                                            <span className="text-gray-400 font-bold line-through">
                                                {data.discountPercent}
                                            </span>
                                        </p>
                                    </div>
                                </div>
                            );
                        })
                    ) : (
                        <p className="text-center text-gray-500 w-full">
                            No products available at the moment. Please try again later.
                        </p>
                    )}
                </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>
        </>
    );
}
```

```tsx
"use client";
import Image from "next/image";
import { FaStar } from "react-icons/fa";
import { Minus, Plus } from "lucide-react";
import { useEffect, useState } from "react";
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
import CustomerTestimonials from "@/components/AllReviews";
import Top_sell from "@/components/arrivals";
import { BreadcrumbCollapsed } from "@/components/Breadcrupm";
import { useDispatch } from "react-redux";
import Toastify from "@/app/cart/toastify";

// Adding key prop in star array
let star = [
  <FaStar key={1} />,
  <FaStar key={2} />,
  <FaStar key={3} />,
  <FaStar key={4} />,
  <FaStar key={5} />,
];

interface Iproducts {
  image: string[];
  discountPercent: number;
  isNew: boolean;
  name: string;
  description: string;
  price: number;
  _id: string;
  colors: string[];
  sizes: string[];
}

export default function SlugPage({ params }: { params: { id: string } }) {
  const [product, setProduct] = useState<Iproducts | null>(null);
  const [cartItem, setCartItem] = useState<any>(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(false);
   const disispatch = useDispatch()
  useEffect(() => {
    const fetchProduct = async () => {
      try {
        setLoading(true);
        const products: Iproducts[] = await client.fetch(
          `*[_type == 'products']{
            "image": image.asset->url,
            category,
            discountPercent,
            isNew,
            name,
            description,
            price,
            _id,
            colors,
            sizes
          }`
        );

        const slug = products.find((item) => item._id === params.id);

        if (!slug) {
          setError(true);
        } else {
          setProduct(slug);
          setCartItem({
            id: slug._id,
            name: slug.name,
            image: slug.image,
            price: slug.price,
            size: slug.sizes[0],
            color: slug.colors[0],
            qty: 1,
            discount: slug.discountPercent,
          });
        }
      } catch (err) {
        setError(true);
      } finally {
        setLoading(false);
      }
    };

    fetchProduct();
  }, [params.id]);


  if (loading) {
    return <h1 className="text-center mt-28 font-bold">Loading...</h1>;
  }

  if (error || !product) {
    return <h1 className="mt-28 text-center font-bold">Product not found</h1>;
  }


  return (
    <>
    <div className="mt-28 md:mt-36">
      <BreadcrumbCollapsed/>
      <div className="flex h-full items-center flex-col md:flex-row justify-center sm:justify-evenly  sm:p-0 max-w-screen-2xl mx-auto">
        {/* Left */}
        <div className="flex space-x-4 md:space-x-0  md:space-y-3 p-5 md:flex-col justify-between items-center md:w-[200px]  order-2 md:order-1">
          {product.image &&
            <Image
              key={product._id}
              src={urlFor(product.image).url()}
              className="w-[100px]  h-[100px] md:h-[150px] lg:mt-3 rounded-[20px]"
              alt={product.name}
              width={100}
              height={100}
            />
          }
          {product.image &&
```

Product Detail Page

Dynamic Routing

```tsx

  if (loading) {
    return <h1 className="text-center mt-28 font-bold">Loading...</h1>;
  }

  if (error || !product) {
    return <h1 className="mt-28 text-center font-bold">Product not found</h1>;
  }


  return (
    <>
    <div className="mt-28 md:mt-36">
      <BreadcrumbCollapsed/>
      <div className="flex h-full items-center flex-col md:flex-row justify-center sm:justify-evenly  sm:p-0 max-w-screen-2xl mx-auto">
        {/* Left */}
        <div className="flex space-x-4 md:space-x-0  md:space-y-3 p-5 md:flex-col justify-between items-center md:w-[200px]  order-2 md:order-1">
          {product.image &&
            <Image
              key={product._id}
              src={urlFor(product.image).url()}
              className="w-[100px]  h-[100px] md:h-[150px] lg:mt-3 rounded-[20px]"
              alt={product.name}
              width={100}
              height={100}
            />
          }
          {product.image &&
            <Image
              key={product._id}
              src={urlFor(product.image).url()}
              className="w-[100px]  h-[100px] md:h-[150px] lg:mt-3 rounded-[20px]"
              alt={product.name}
              width={100}
              height={100}
            />
          }
          {product.image &&
            <Image
              key={product._id}
              src={urlFor(product.image).url()}
              className="w-[100px]  h-[100px] md:h-[150px] lg:mt-3 rounded-[20px]"
              alt={product.name}
              width={100}
              height={100}
            />
          }
        </div>
        {/* Mid */}
        <div className="w-[90%] pb-3  h-[260px] lg:w-[500px] md:h-[500px] mt-5 lg:mt-0 order-1 md::order-2">
        {product.image &&
            <Image
              key={product._id}
              src={urlFor(product.image).url()}
              className=" w-full h-full sm:mt-3 rounded-[20px]"
              alt={product.name}
              width={100}
              height={100}
            />
          }
        </div>
        {/* Right */}
        <div className="w-full p-5 lg:w-[500px] lg:h-[500px]  order-3">
          <h1 className="text-2xl lg:text-3xl font-bold">{cartItem.name}</h1>
          <div className="flex text-yellow-400">{star}</div>
          <div className="flex items-center space-x-2">
            <p className="font-bold">{cartItem.price * cartItem.qty}</p>
            {cartItem.discount > 0 && (
              <span className="text-gray-400 line-through">
                {(cartItem.price - (cartItem.price * cartItem.discount) / 100) *
                  cartItem.qty}
              </span>
            )}
            {cartItem.discount > 0 && (
              <span className="bg-red-400 rounded-[10px]">{`-${cartItem.discount}% `}</span>
            )}
          </div>
          <p className="text-sm">{product.description}</p>
          {/* Select Color */}
          <div className="mt-5">
            <p className="text-gray-500">Select Colors</p>
            <div className="flex space-x-3 mt-2">
              {product.colors.map((color, i) => (
                <button
                  key={i}
                  onClick={() => setCartItem({ ...cartItem, color })}
                  className="w-[37px] h-[37px] border border-black active:outline rounded-full flex justify-center items-center"
                  style={{ backgroundColor: color }}
                ></button>
              ))}
            </div>
          </div>
          {/* Choose Size */}
          <div className="mt-4">
            <p className="text-gray-500">Choose Size</p>
            <div className="flex space-x-3 mt-2">
              {product.sizes.map((size, i) => (
                <button
                  key={i}
                  onClick={() => setCartItem({ ...cartItem, size })}
                  className="w-[80px] h-[40px] flex justify-center items-center active:outline rounded-[62px] bg-[#F0F0F0] text-gray-400"
                >
                  {size}
                </button>
              ))}
            </div>
          </div>
          {/* Quantity & Add to Cart */}
          <div className="flex justify-start items-center mt-7 space-x-4">
            <button
              onClick={() =>
                setCartItem({
                  ...cartItem,
                  qty: cartItem.qty <= 1 ? 1 : --cartItem.qty,
                })
              }
            >
              <Minus />
            </button>
            <span>{cartItem.qty}</span>
            <button
              onClick={() =>
                setCartItem({ ...cartItem, qty: ++cartItem.qty })
              }
            >
              <Plus />
            </button>
            {/* <Button onClick={()=>handleadd(cartItem)} className="bg-black text-white w-[300px]">Add to Cart</Button> */}
            <Toastify cartItem = {cartItem}/>
          </div>
        </div>
      </div>
    </div>
      <CustomerTestimonials/>
      <Top_sell/>
    </>
  );
}
```

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
```

Hooks

```tsx
import { useDispatch, useSelector } from 'react-redux'
import type { AppDispatch, RootState } from './store'

// Use throughout your app instead of plain `useDispatch` and `useSelector`
export const useAppDispatch = useDispatch.withTypes<AppDispatch>()
export const useAppSelector = useSelector.withTypes<RootState>()
```

Cartslice

```tsx
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` a
  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

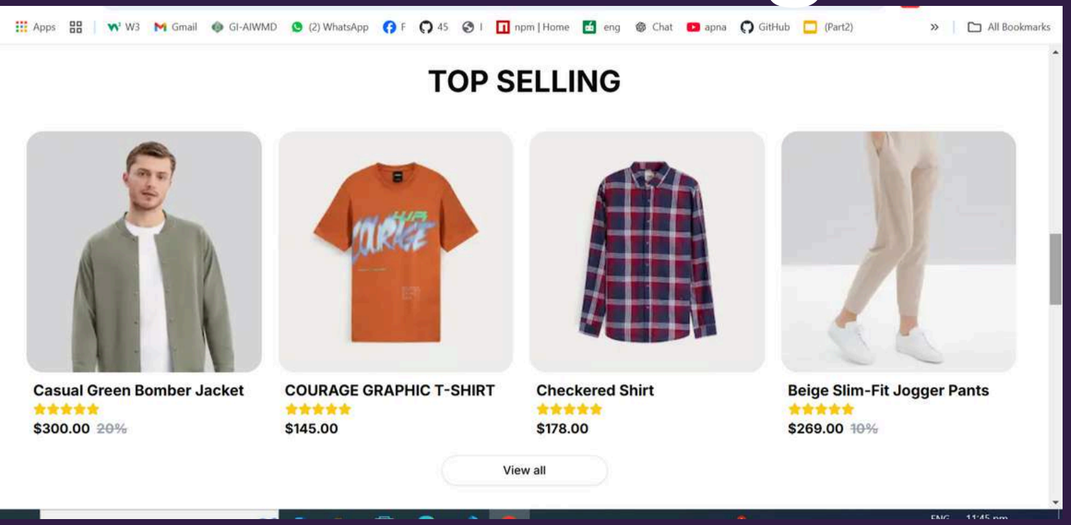
```tsx
"use client"
import { Provider } from "react-redux";
import { store } from "../app/Redux/store";
import { PersistGate } from "redux-persist/integration/react";
import { persistStore } from "redux-persist";


function Providers({ children,}: Readonly<{children: React.ReactNode;}>) {
  let persistore = persistStore(store)
  return (
    <Provider store={store}>
      <PersistGate persistor={persistore}>
        {children}
        </PersistGate>
    </Provider>
  )
}

export default Providers
```

Pagination & Toastify

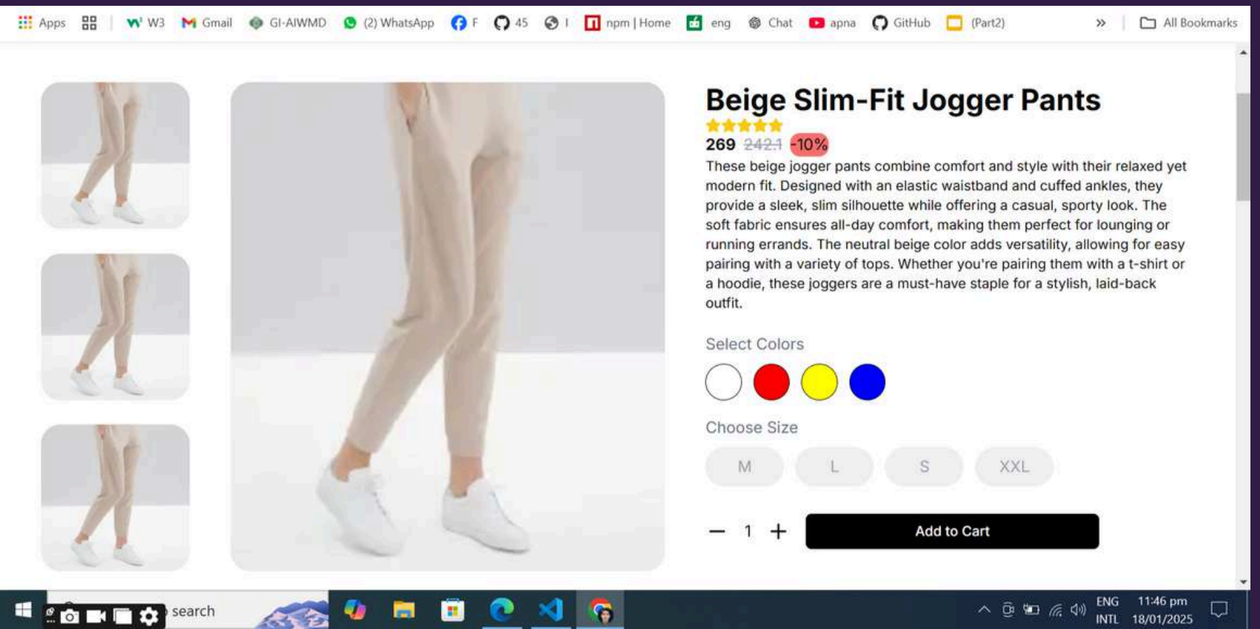
```tsx
import {
    Pagination,
    PaginationContent,
    PaginationEllipsis,
    PaginationItem,
    PaginationLink,
    PaginationNext,
    PaginationPrevious,
  } from "@/components/ui/pagination"
import React from 'react'

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
```

```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 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
```

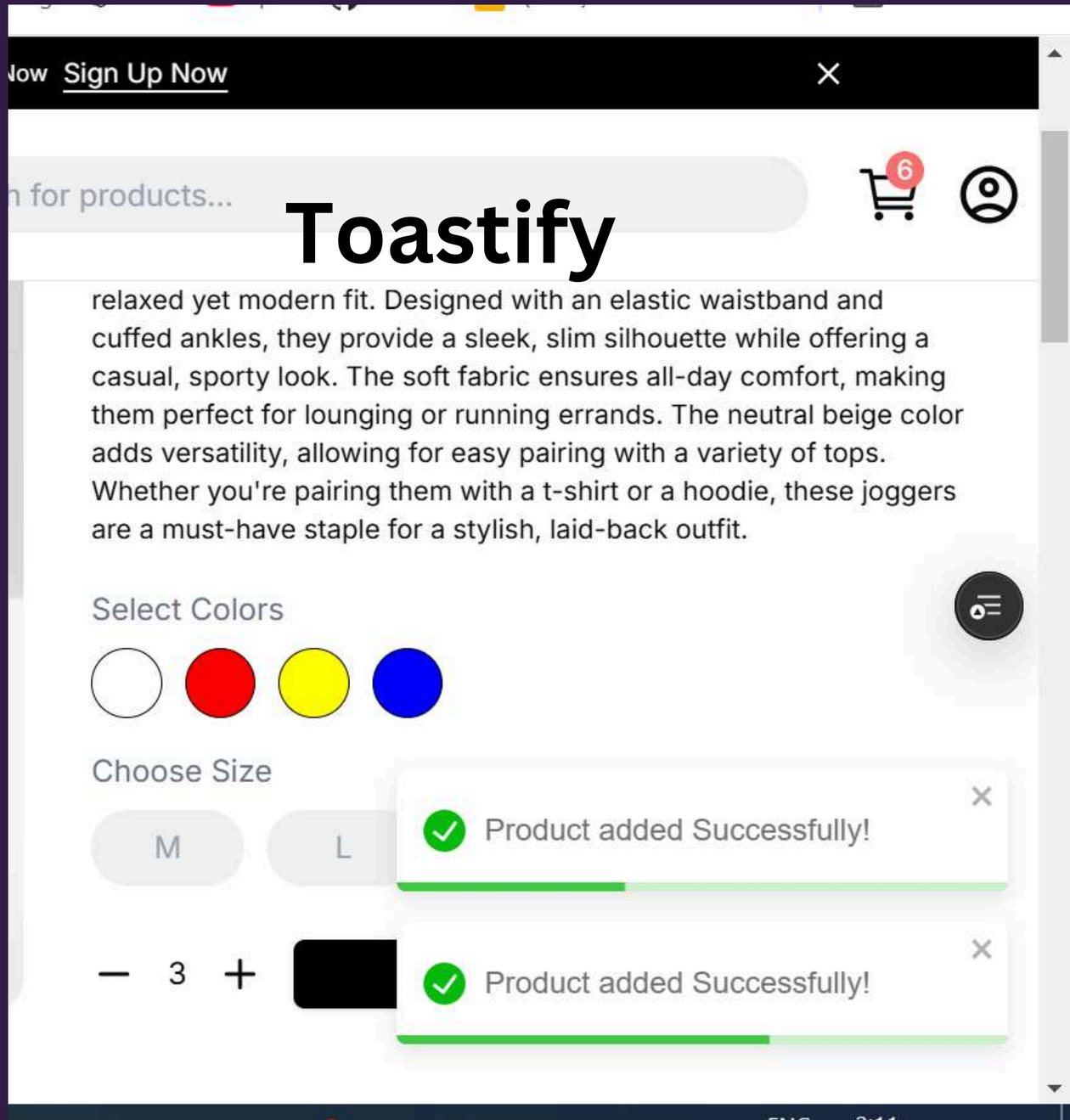
Product Listing



Detail Page



Toastify



Cart Page

hackathon-cyan-one.vercel.app/cart

Apps | W3 | Gmail | GI-AIWMD | (2) WhatsApp | F | 45 | I | npm | Home | eng | Chat | apna | GitHub | (Part2) | All Bookmar

Sign up and get 20% off to your first order. Sign Up Now Sign Up Now

SHOP.CO

Shop On Sale New Arrivals Brands

Search for products...

1

Home > Shop > Mens T-shirts >

Casual Green Bomber Jacket

Size:XL

Color:Yellow

$960

— 4 +

Order Summary

Subtotal	$960
Discount (-20%)	-$0
Delivery Fee	$0
Total	**$960**

Add promo code Apply

Go to Checkout

THANK YOU