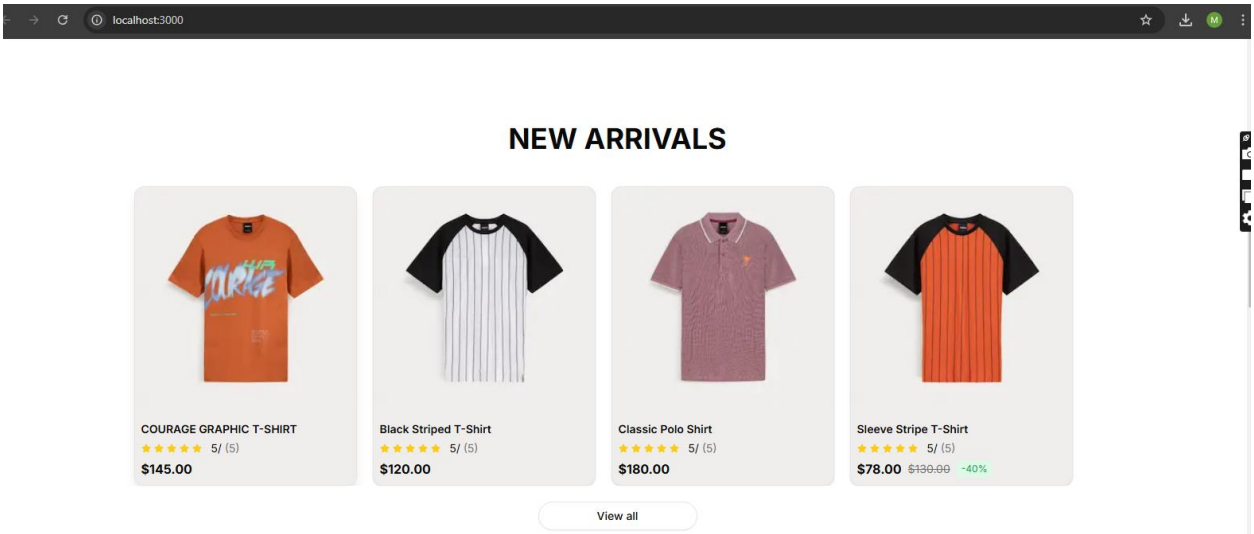
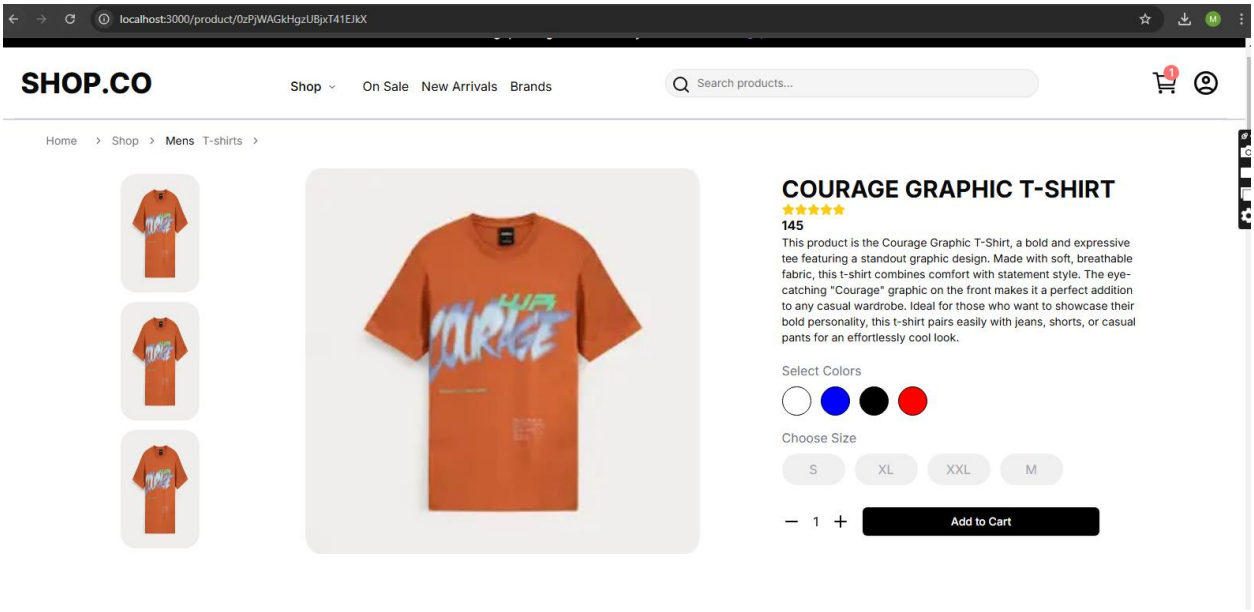
Day 4 Dynamic Frontend Components [SHOP.CO]

This project focuses on solve the problem of online shopping for men and giving them the best choices for shirts , t-shirts , pants and hoddies while implementing dynamic frontend components for a marketplace application. Key features include product listing, dynamic routing for product details, a functional search bar, and category filters.

Product listing of dynamic

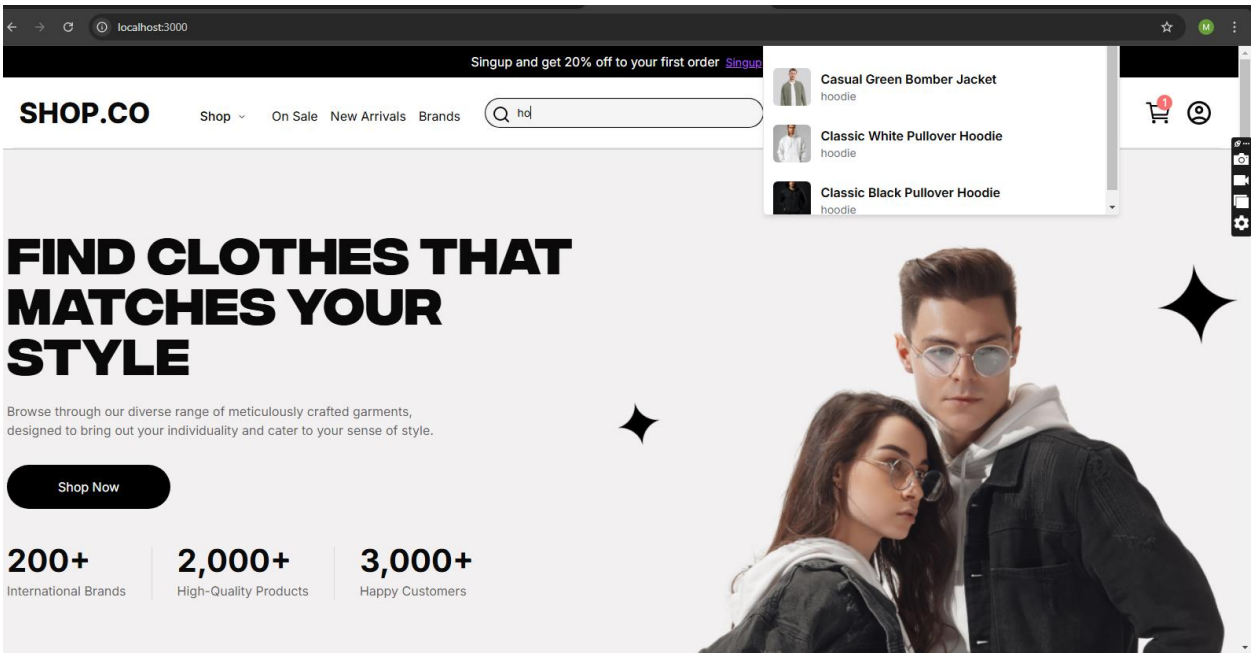


Indivisual product detail with dynamic routing

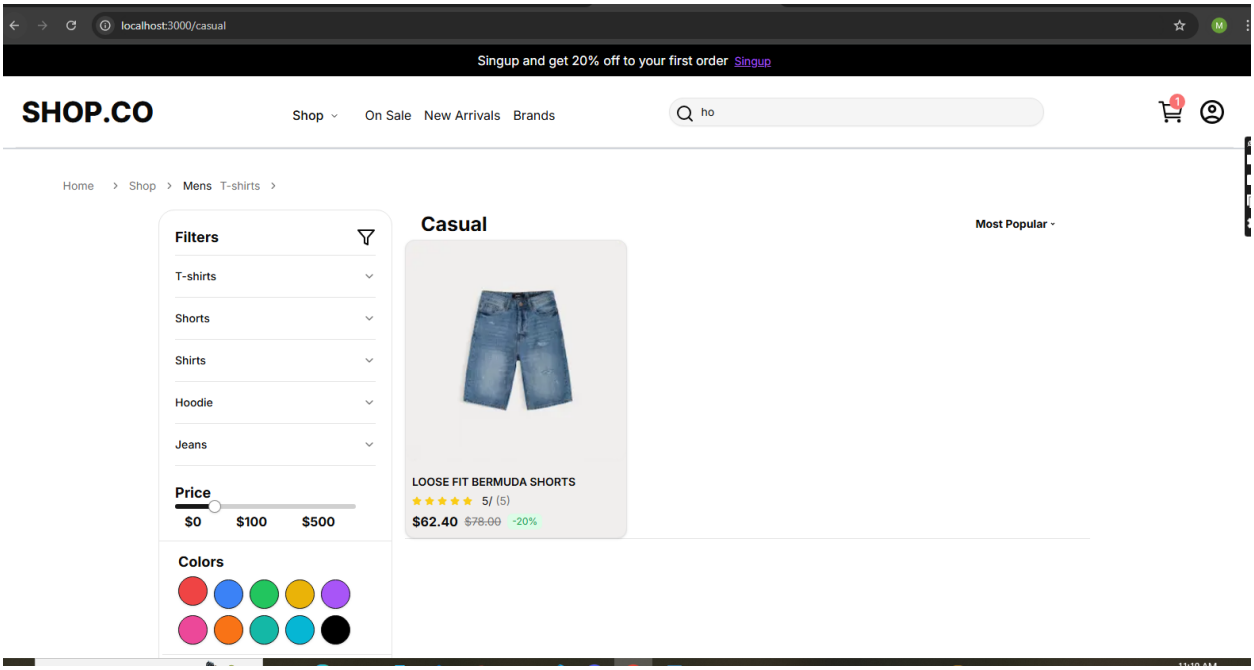


Day 4 Dynamic Frontend Components [SHOP.CO]

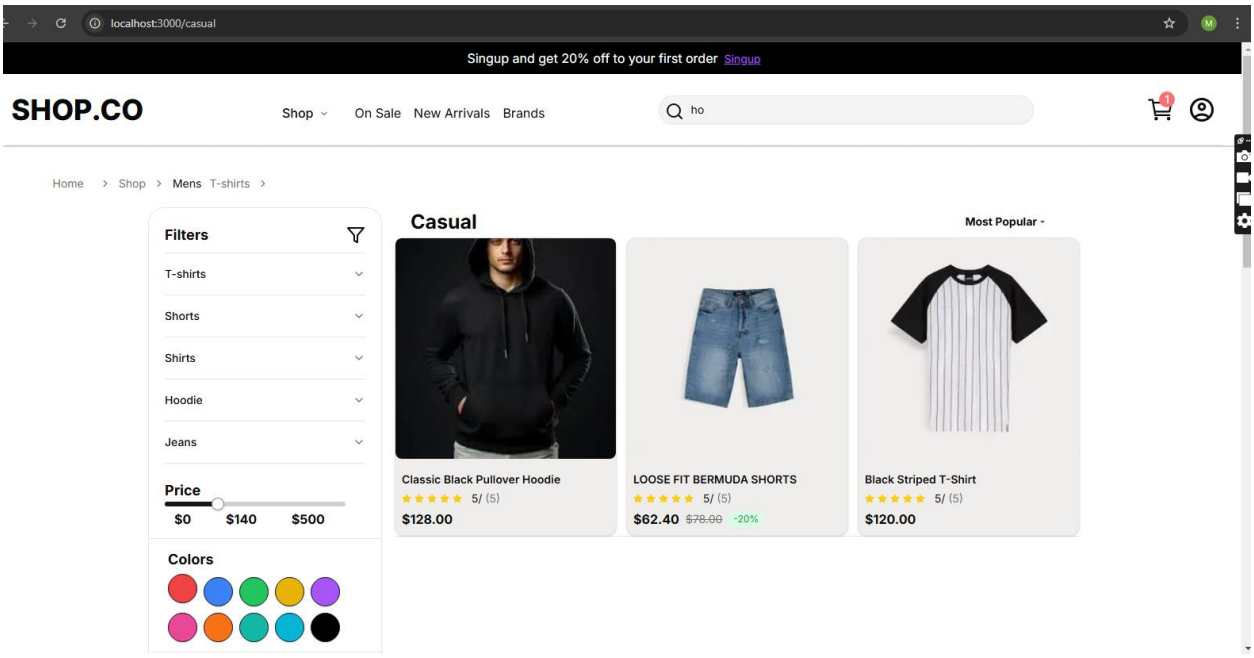
Search bar

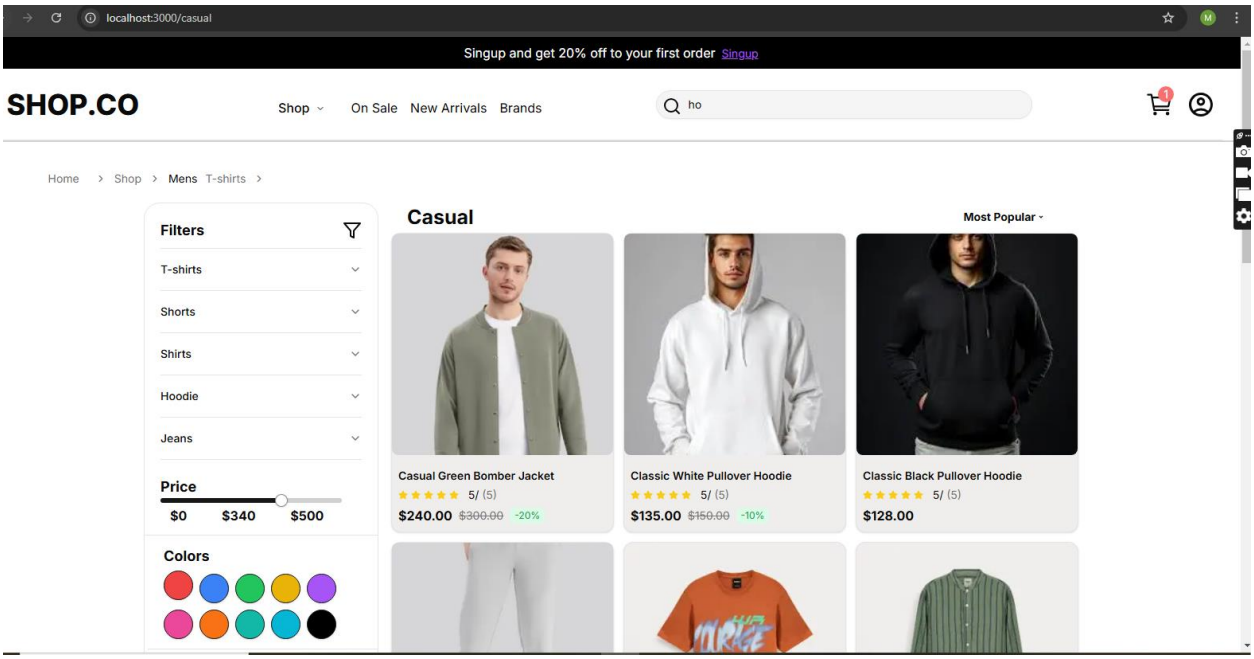


Price filter



Day 4 Dynamic Frontend Components [SHOP.CO]

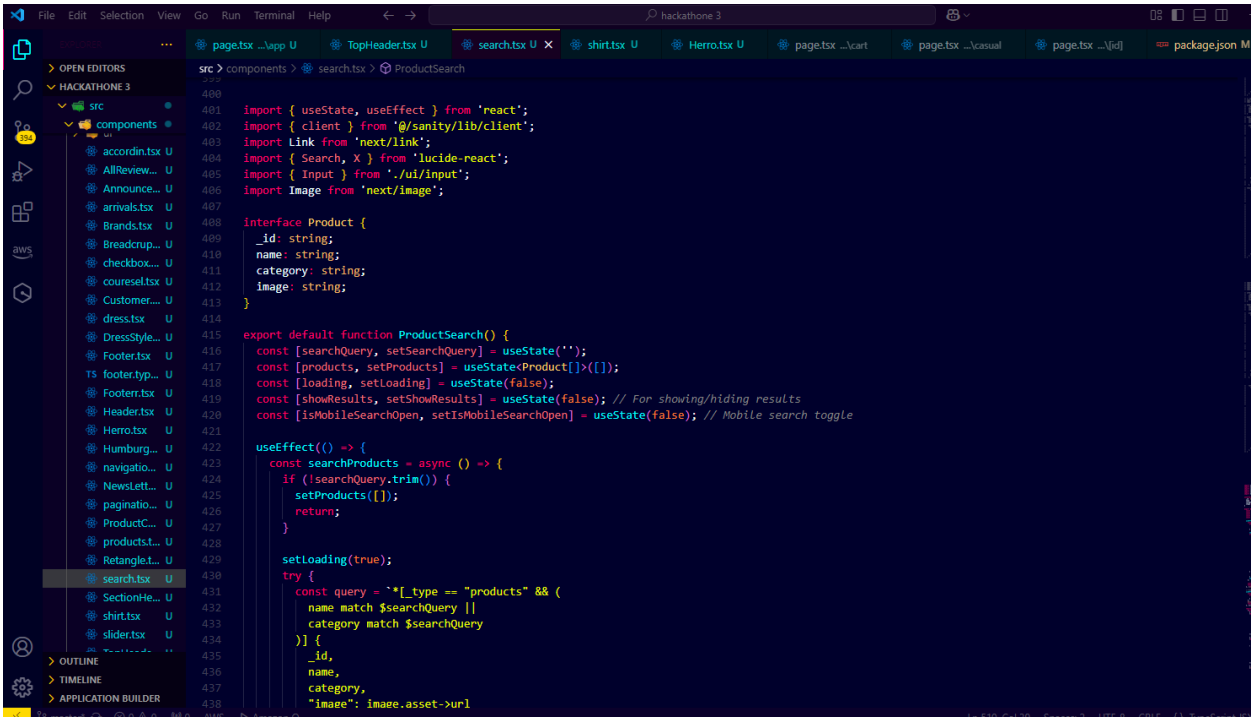




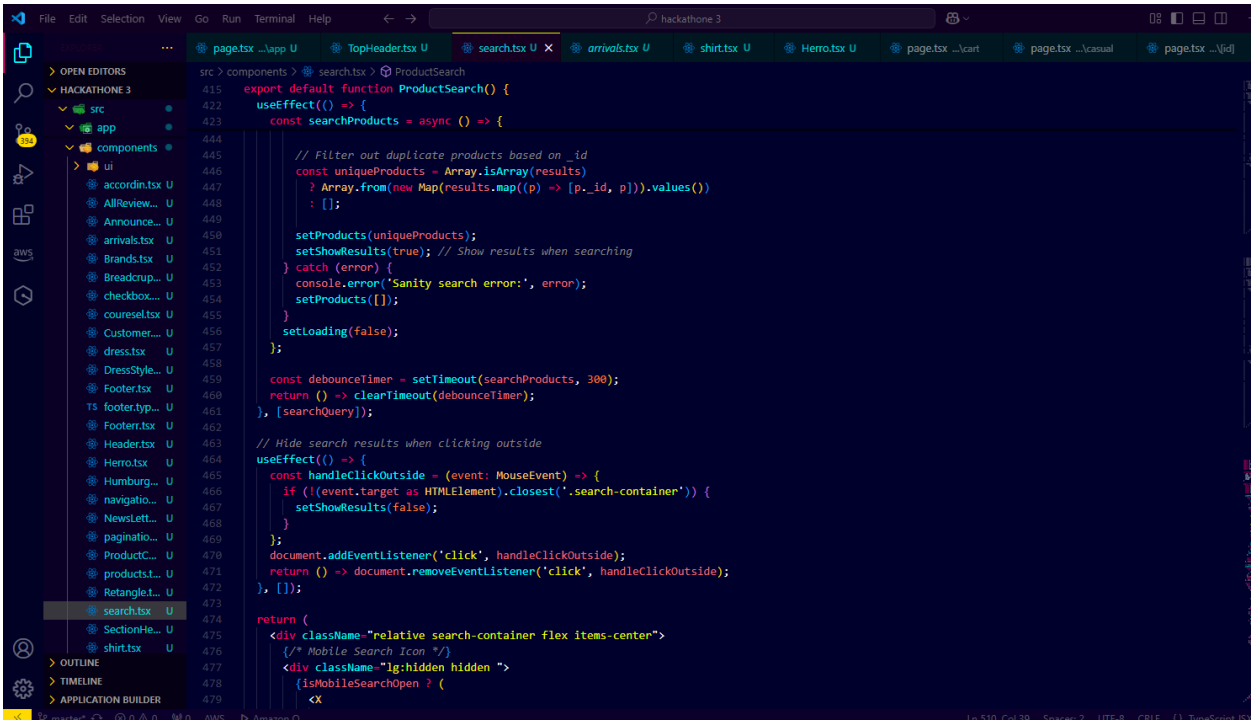
Code Snippets

Search Bar

Day 4 Dynamic Frontend Components [SHOP.CO]

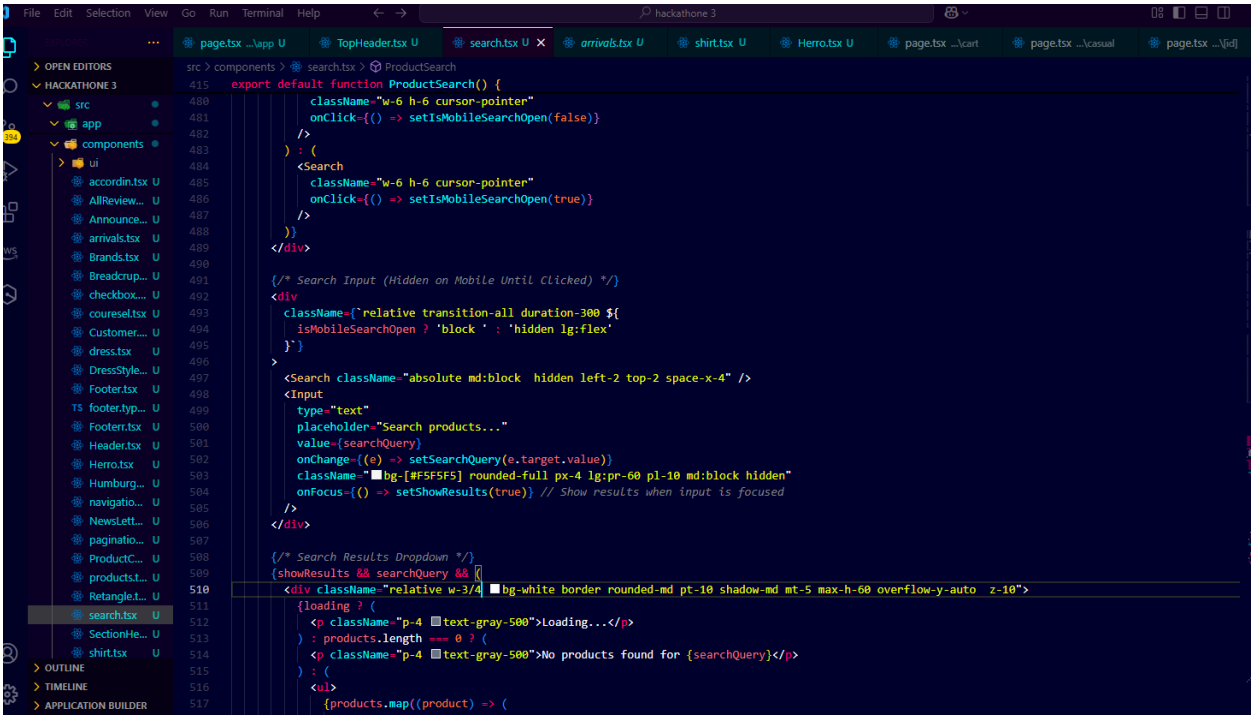


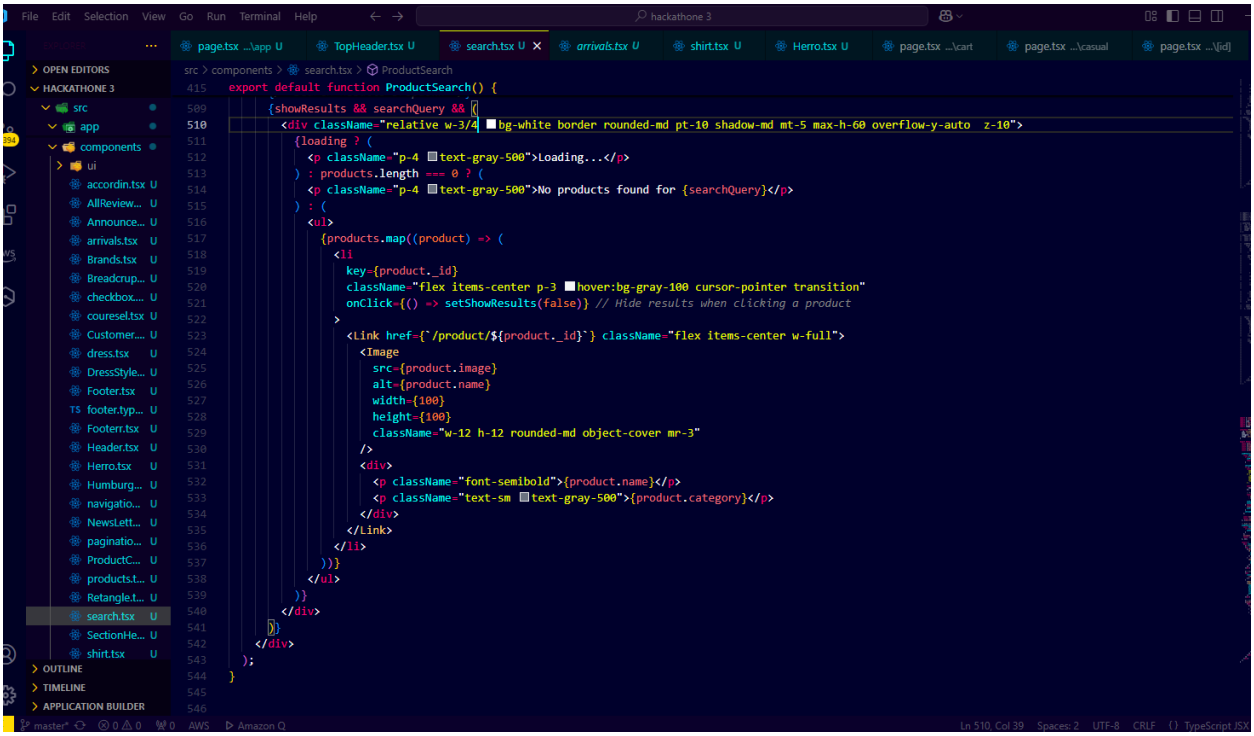
```tsx
import { useState, useEffect } from 'react';
import { client } from '@/sanity/lib/client';
import Link from 'next/link';
import { Search, X } from 'lucide-react';
import { Input } from './ui/input';
import Image from 'next/image';

interface Product {
  _id: string;
  name: string;
  category: string;
  image: string;
}

export default function ProductSearch() {
  const [searchQuery, setSearchQuery] = useState('');
  const [products, setProducts] = useState<Product[]>([]);
  const [loading, setLoading] = useState(false);
  const [showResults, setShowResults] = useState(false); // For showing/hiding results
  const [isMobileSearchOpen, setIsMobileSearchOpen] = useState(false); // Mobile search toggle

  useEffect(() => {
    const searchProducts = async () => {
      if (!searchQuery.trim()) {
        setProducts([]);
        return;
      }

      setLoading(true);
      try {
        const query = `*[_type == "products" && (
          name match $searchQuery ||
          category match $searchQuery
        )] {
          _id,
          name,
          category,
          "image": image.asset->url
```



```tsx
export default function ProductSearch() {
  useEffect(() => {
    const searchProducts = async () => {

      // Filter out duplicate products based on _id
      const uniqueProducts = Array.isArray(results)
        ? Array.from(new Map(results.map((p) => [p._id, p])).values())
        : [];

      setProducts(uniqueProducts);
      setShowResults(true); // Show results when searching
    } catch (error) {
      console.error('Sanity search error:', error);
      setProducts([]);
    }
    setLoading(false);
  };

  const debounceTimer = setTimeout(searchProducts, 300);
  return () => clearTimeout(debounceTimer);
}, [searchQuery]);

// Hide search results when clicking outside
useEffect(() => {
  const handleClickOutside = (event: MouseEvent) => {
    if (!(event.target as HTMLElement).closest('.search-container')) {
      setShowResults(false);
    }
  };
  document.addEventListener('click', handleClickOutside);
  return () => document.removeEventListener('click', handleClickOutside);
}, []);

return (
  <div className="relative search-container flex items-center">
    {/* Mobile Search Icon */}
    <div className="lg:hidden hidden ">
      {isMobileSearchOpen ? (
        <X
```

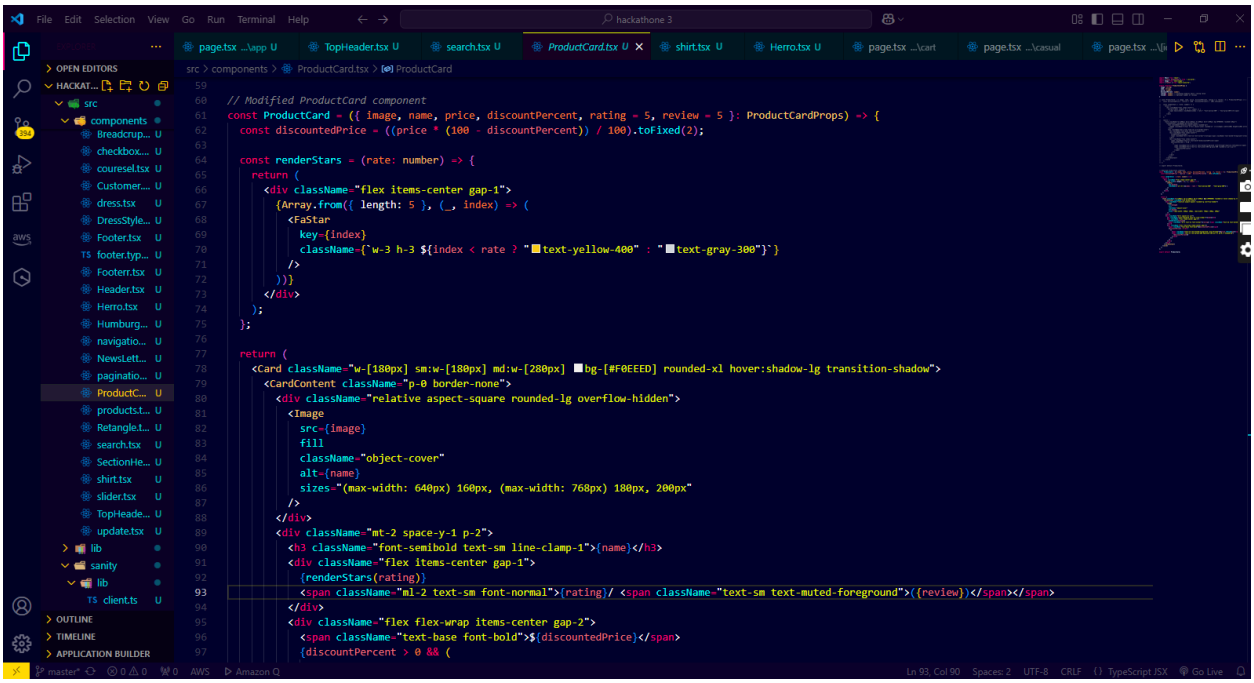
Day 4 Dynamic Frontend Components [SHOP.CO]



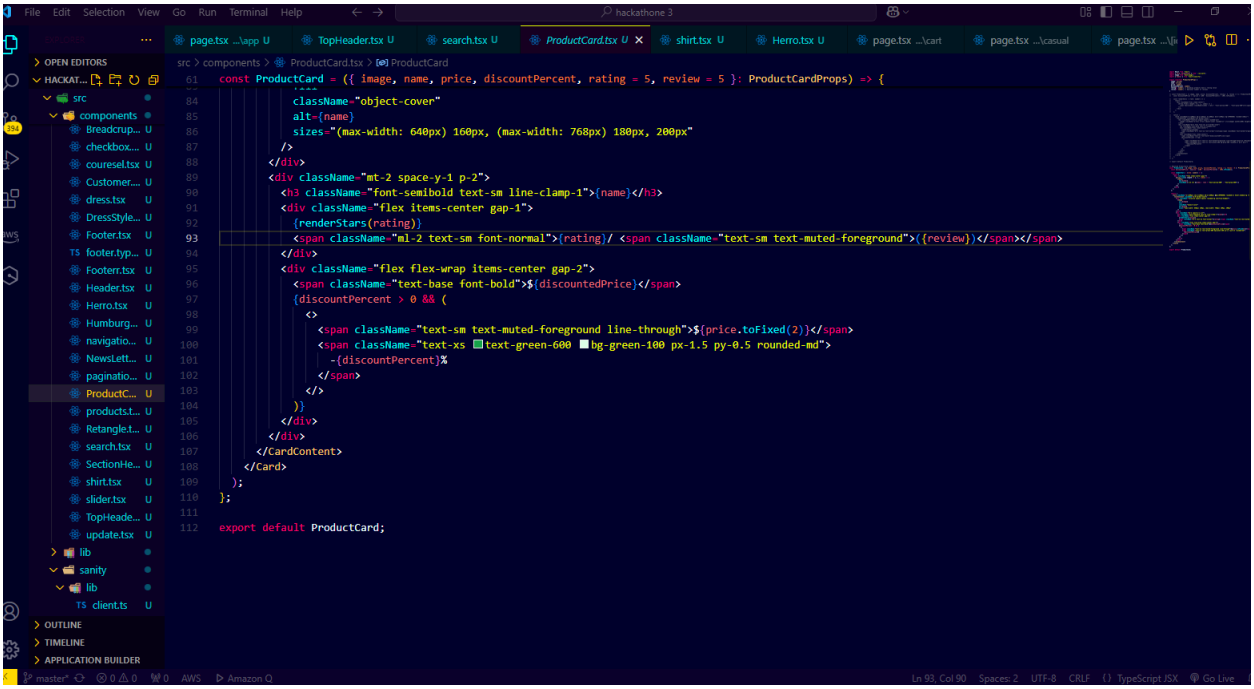


Product Card

Day 4 Dynamic Frontend Components [SHOP.CO]



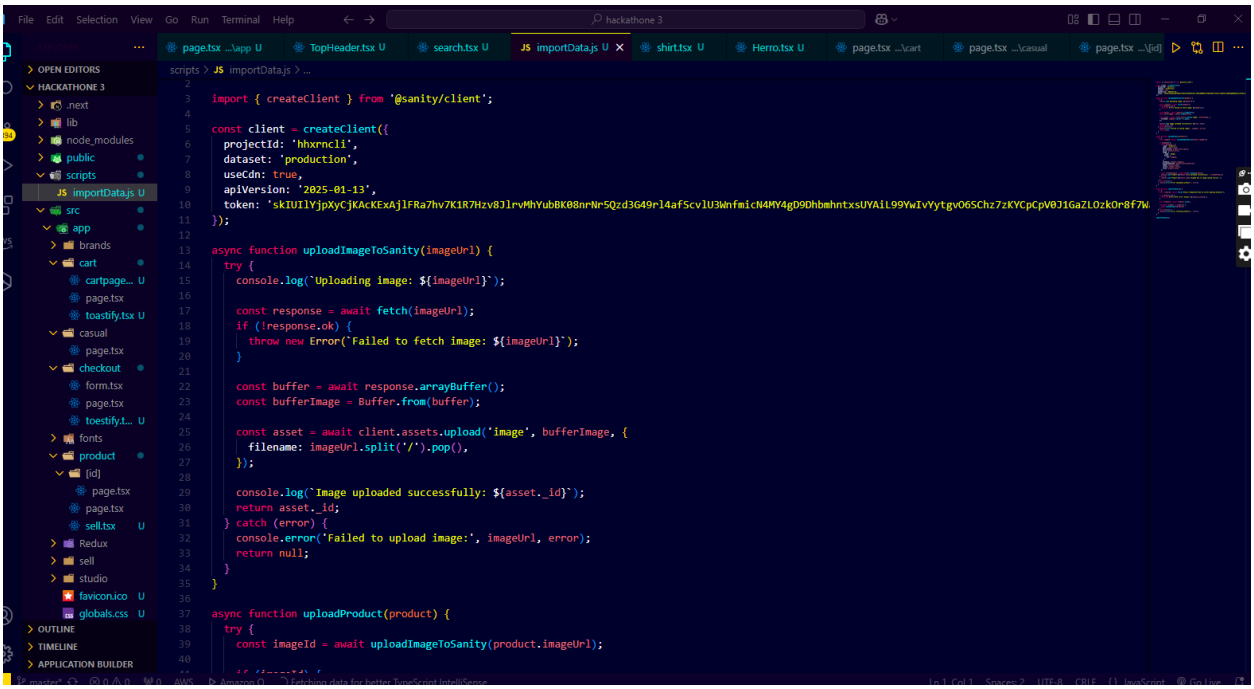
```tsx
// Modified ProductCard component
const ProductCard = ({ image, name, price, discountPercent, rating = 5, review = 5 }: ProductCardProps) => {
  const discountedPrice = ((price * (100 - discountPercent)) / 100).toFixed(2);

  const renderStars = (rate: number) => {
    return (
      <div className="flex items-center gap-1">
        {Array.from({ length: 5 }, (_, index) => (
          <FaStar
            key={index}
            className={`w-3 h-3 ${index < rate ? "text-yellow-400" : "text-gray-300"}`}
          />
        ))}
      </div>
    );
  };

  return (
    <Card className="w-[180px] sm:w-[180px] md:w-[280px] bg-[#F0EEED] rounded-xl hover:shadow-lg transition-shadow">
      <CardContent className="p-0 border-none">
        <div className="relative aspect-square rounded-lg overflow-hidden">
          <Image
            src={image}
            fill
            className="object-cover"
            alt={name}
            sizes="(max-width: 640px) 160px, (max-width: 768px) 180px, 200px"
          />
        </div>
        <div className="mt-2 space-y-1 p-2">
          <h3 className="font-semibold text-sm line-clamp-1">{name}</h3>
          <div className="flex items-center gap-1">
            {renderStars(rating)}
            <span className="ml-2 text-sm font-normal">{rating}/ <span className="text-sm text-muted-foreground">({review})</span></span>
          </div>
          <div className="flex flex-wrap items-center gap-2">
            <span className="text-base font-bold">${discountedPrice}</span>
            {discountPercent > 0 && (
```



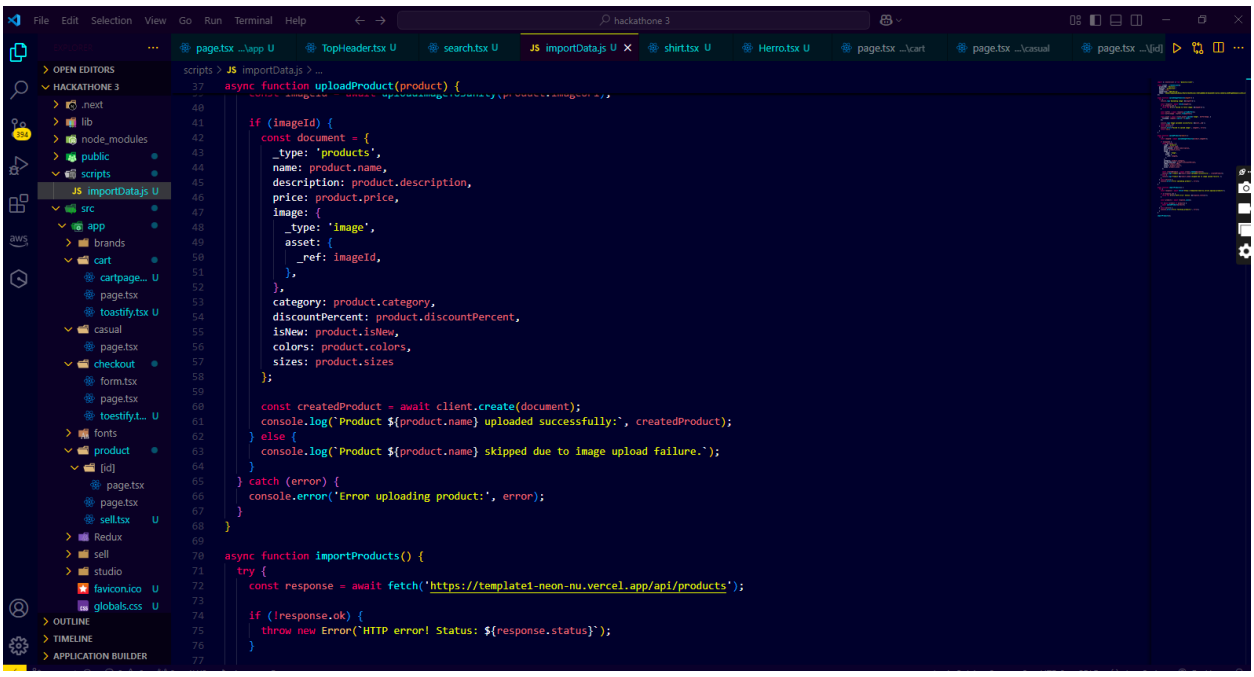
```tsx
const ProductCard = ({ image, name, price, discountPercent, rating = 5, review = 5 }: ProductCardProps) => {
            className="object-cover"
            alt={name}
            sizes="(max-width: 640px) 160px, (max-width: 768px) 180px, 200px"
          />
        </div>
        <div className="mt-2 space-y-1 p-2">
          <h3 className="font-semibold text-sm line-clamp-1">{name}</h3>
          <div className="flex items-center gap-1">
            {renderStars(rating)}
            <span className="ml-2 text-sm font-normal">{rating}/ <span className="text-sm text-muted-foreground">({review})</span></span>
          </div>
          <div className="flex flex-wrap items-center gap-2">
            <span className="text-base font-bold">${discountedPrice}</span>
            {discountPercent > 0 && (
              <>
                <span className="text-sm text-muted-foreground line-through">${price.toFixed(2)}</span>
                <span className="text-xs text-green-600 bg-green-100 px-1.5 py-0.5 rounded-md">
                  -{discountPercent}%
                </span>
              </>
            )}
          </div>
        </div>
      </CardContent>
    </Card>
  );
};

export default ProductCard;
```

Day 4 Dynamic Frontend Components [SHOP.CO]

Scripts



```
import { createClient } from '@sanity/client';

const client = createClient({
  projectId: 'hhxrncli',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-13',
  token: 'skIUIlYjpXyCjKAcKExAjlFRa7hv7K1R7Hzv8JlrvMhYubBK08nrNr5Qzd3G49rl4afScvlU3WnfmicN4MY4gD9Dhbmhntx...',
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      filename: imageUrl.split('/').pop(),
    });

    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

async function uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);
```



```
async function uploadProduct(product) {

    if (imageId) {
      const document = {
        _type: 'products',
        name: product.name,
        description: product.description,
        price: product.price,
        image: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        category: product.category,
        discountPercent: product.discountPercent,
        isNew: product.isNew,
        colors: product.colors,
        sizes: product.sizes
      };

      const createdProduct = await client.create(document);
      console.log(`Product ${product.name} uploaded successfully:`, createdProduct);
    } else {
      console.log(`Product ${product.name} skipped due to image upload failure.`);
    }
  } catch (error) {
    console.error('Error uploading product:', error);
  }
}

async function importProducts() {
  try {
    const response = await fetch('https://template1-neon-nu.vercel.app/api/products');

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
```

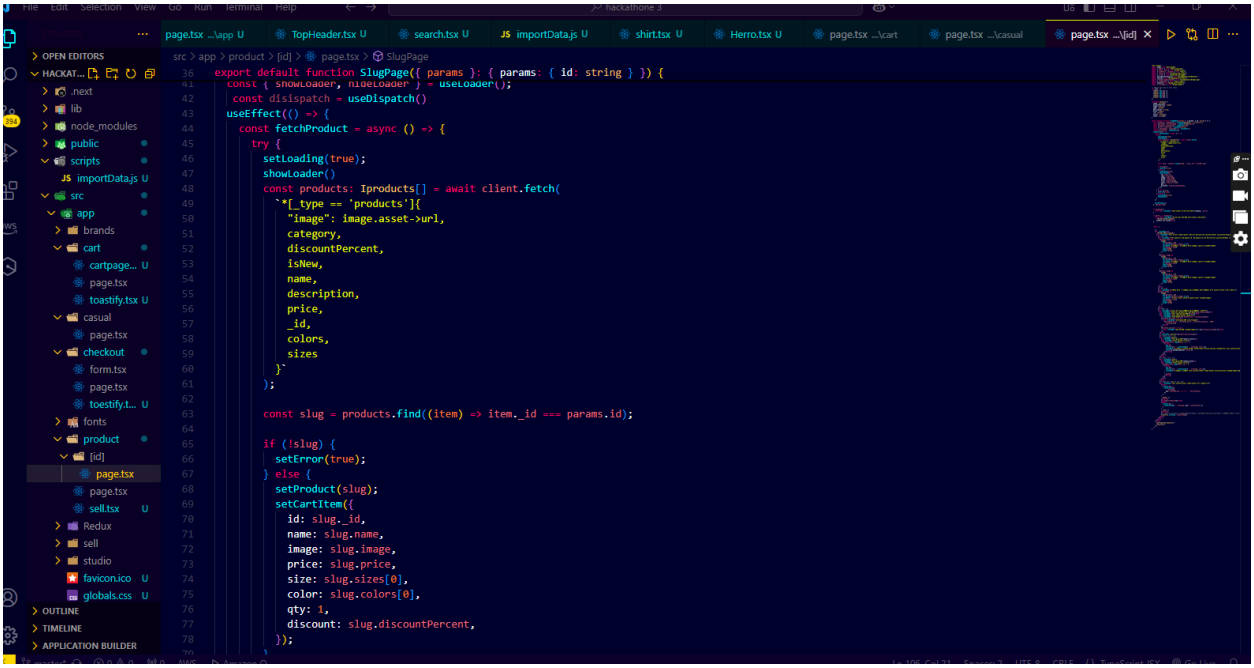
Day 4 Dynamic Frontend Components [SHOP.CO]

```
async function uploadProduct(product) {
    }
  } catch (error) {
    console.error('Error uploading product:', error);
  }
}

async function importProducts() {
  try {
    const response = await fetch('https://template1-neon-nu.vercel.app/api/products');

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error('Error fetching products:', error);
  }
}

importProducts();
```

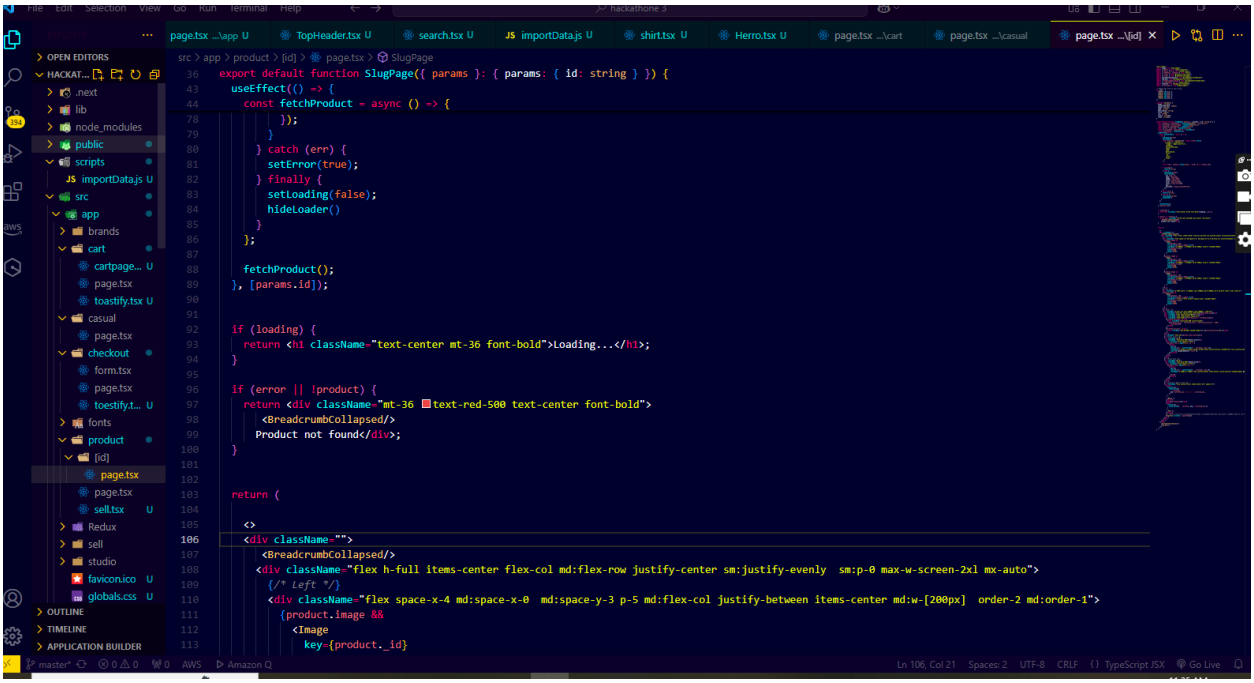
Dynamic Routing & Api Integration

```
"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";
import { useLoader } from "@/app/Loader";

// 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);
```

Day 4 Dynamic Frontend Components [SHOP.CO]

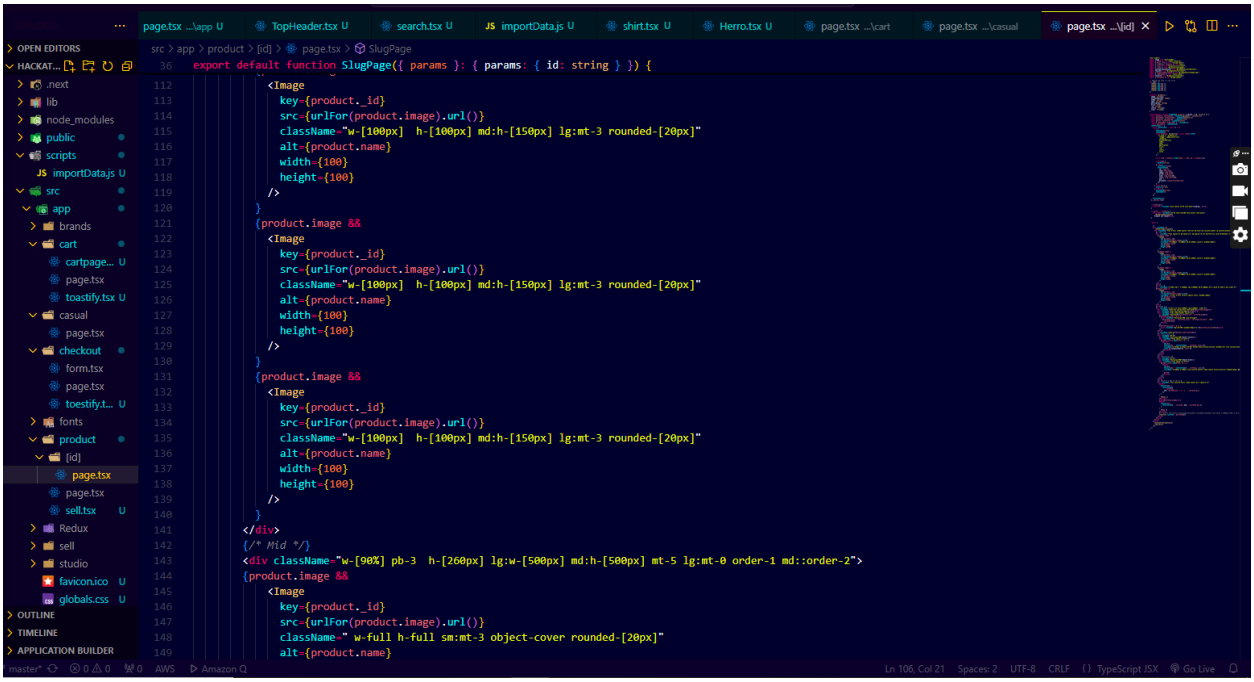


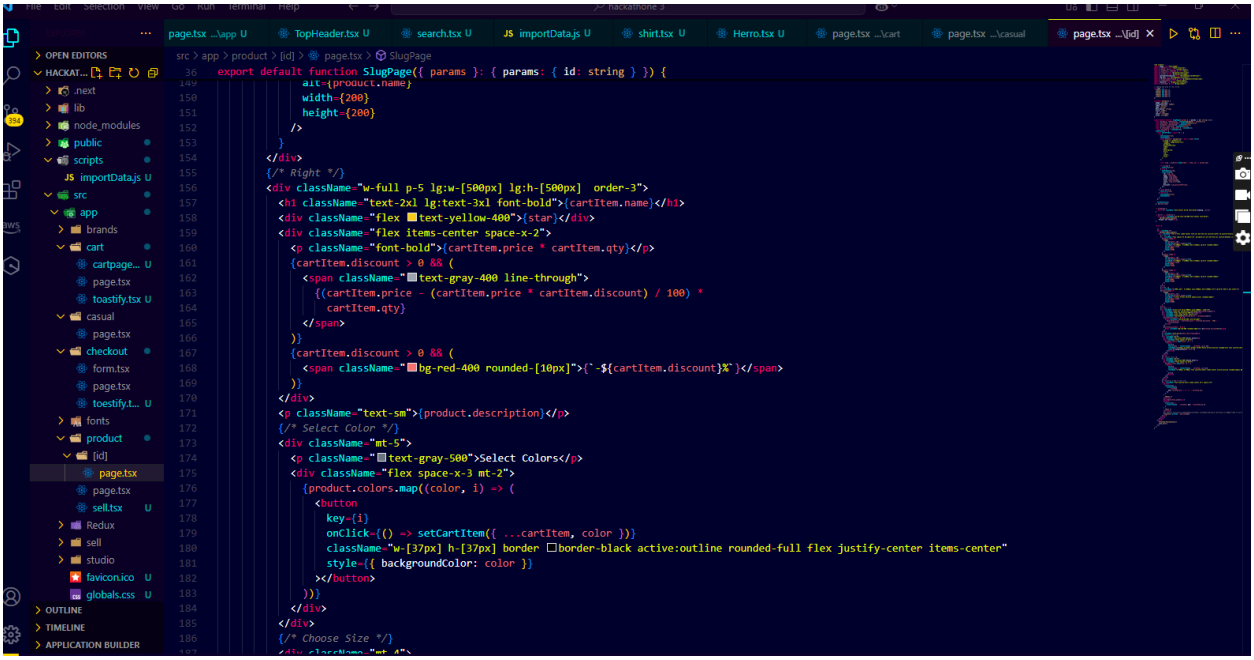
```
export default function SlugPage({ params }: { params: { id: string } }) {
  const { showLoader, hideLoader } = useLoader();
  const disispatch = useDispatch()
  useEffect(() => {
    const fetchProduct = async () => {
      try {
        setLoading(true);
        showLoader()
        const products: 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,
          });
```



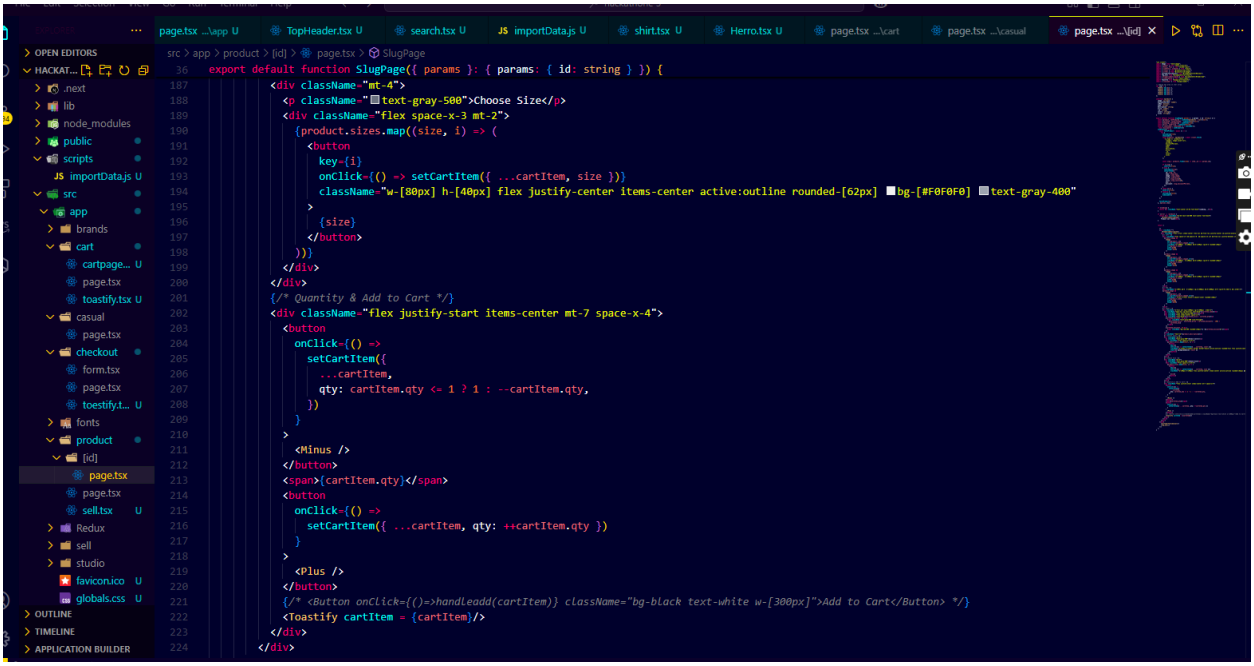
```
export default function SlugPage({ params }: { params: { id: string } }) {
  useEffect(() => {
    const fetchProduct = async () => {
          });
        }
      } catch (err) {
        setError(true);
      } finally {
        setLoading(false);
        hideLoader()
      }
    };

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


  if (loading) {
    return <h1 className="text-center mt-36 font-bold">Loading...</h1>;
  }

  if (error || !product) {
    return <div className="mt-36 text-red-500 text-center font-bold">
      <BreadcrumbCollapsed/>
      Product not found</div>;
  }


  return (

    <>
    <div className="">
      <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}
```

Day 4 Dynamic Frontend Components [SHOP.CO]





Day 4 Dynamic Frontend Components [SHOP.CO]



Technical Report

1. Introduction

This document provides a summary of the development process, including steps taken to build and integrate components, challenges faced, solutions implemented, and best practices followed during the development.

---

2. Development Process

2.1 Steps Taken to Build and Integrate Components

1. **Project Initialization**
    - Set up the Next.js project with TypeScript.
    - Installed necessary dependencies, including TailwindCSS and ShadcnUI for styling.
2. **Component Development**
    - Created reusable UI components such as buttons, modals, and form inputs.
    - Developed dynamic routes for displaying product details.
    - Implemented a global loader to improve user experience during data fetching.

3. **State Management & API Integration**
    - Used React's useState and useEffect for local state management.
    - Integrated APIs using fetch for dynamic content retrieval.
    - Implemented React Query for caching API responses to optimize performance.
4. **Performance Optimization**
    - Enabled Next.js image optimization for faster image loading.
    - Used dynamic imports (next/dynamic) to optimize component loading.
    - Preloaded frequently visited pages using Next.js Link prefetching.

---

3. Challenges Faced & Solutions Implemented

Challenge	**Solution**
Slow page transitions on dynamic routes	Implemented a global loader and prefetching using Next.js Link.
Large API responses causing delay	Used React Query for caching and optimized API requests.
Unoptimized images affecting load speed	Used Next.js Image component for automatic optimization.
UI freezing due to synchronous API calls	Moved heavy computations to background processing using asynchronous operations.

---

4. Best Practices Followed

- **Component Reusability:** Created modular and reusable components for maintainability.
- **State Management:** Used React's built-in state management along with React Query for efficient data handling.
- **Code Optimization:** Followed DRY (Don't Repeat Yourself) and SOLID principles for clean code.
- **Performance Enhancements:**
    - Implemented lazy loading for non-essential components.

    - Used Next.js optimizations like getStaticProps and getServerSideProps where applicable.

- **Error Handling:** Implemented proper error handling for API calls to prevent UI crashes.
- **Accessibility & SEO:** Ensured semantic HTML usage and implemented meta tags for better SEO.

---

5. Conclusion

The project successfully integrates multiple components with an optimized performance approach. By implementing best practices, performance improvements, and a structured development process, the user experience is significantly enhanced. Future improvements could include further optimizations using Next.js middleware and AI-driven caching strategies.