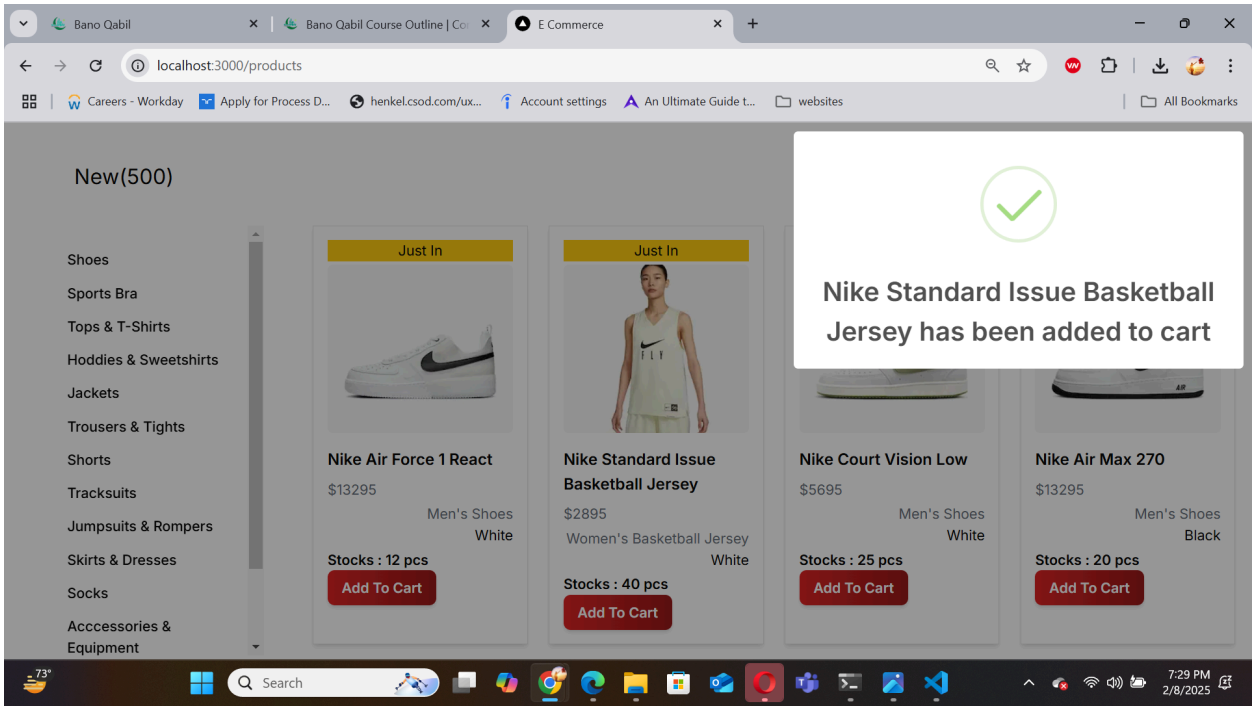
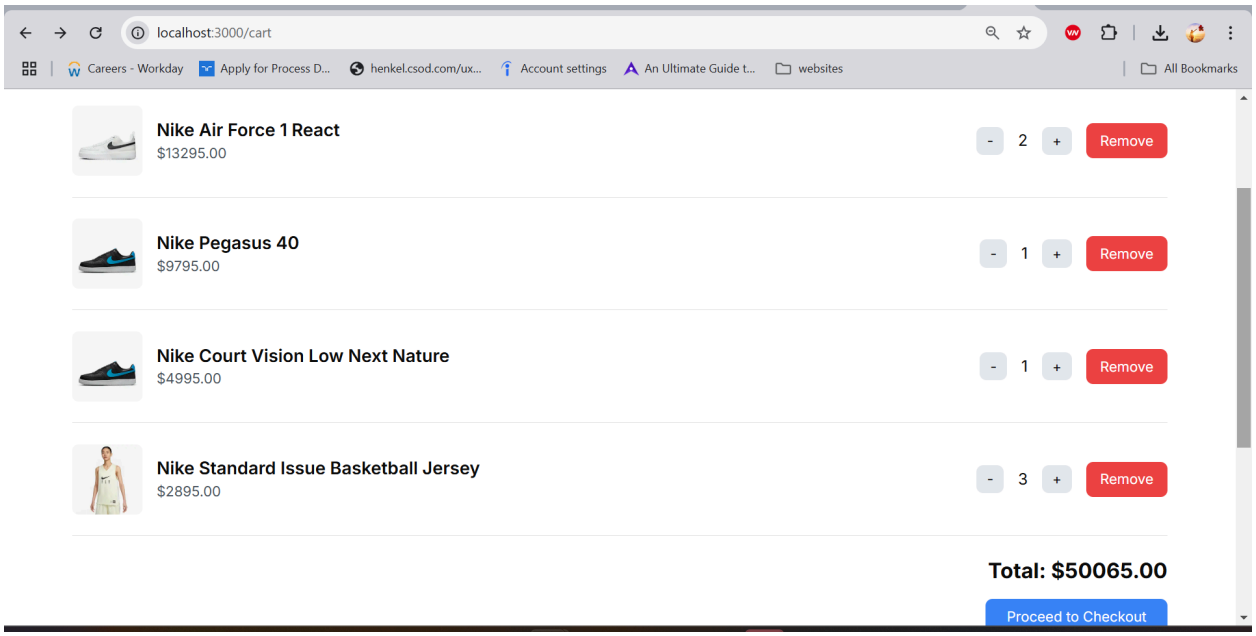
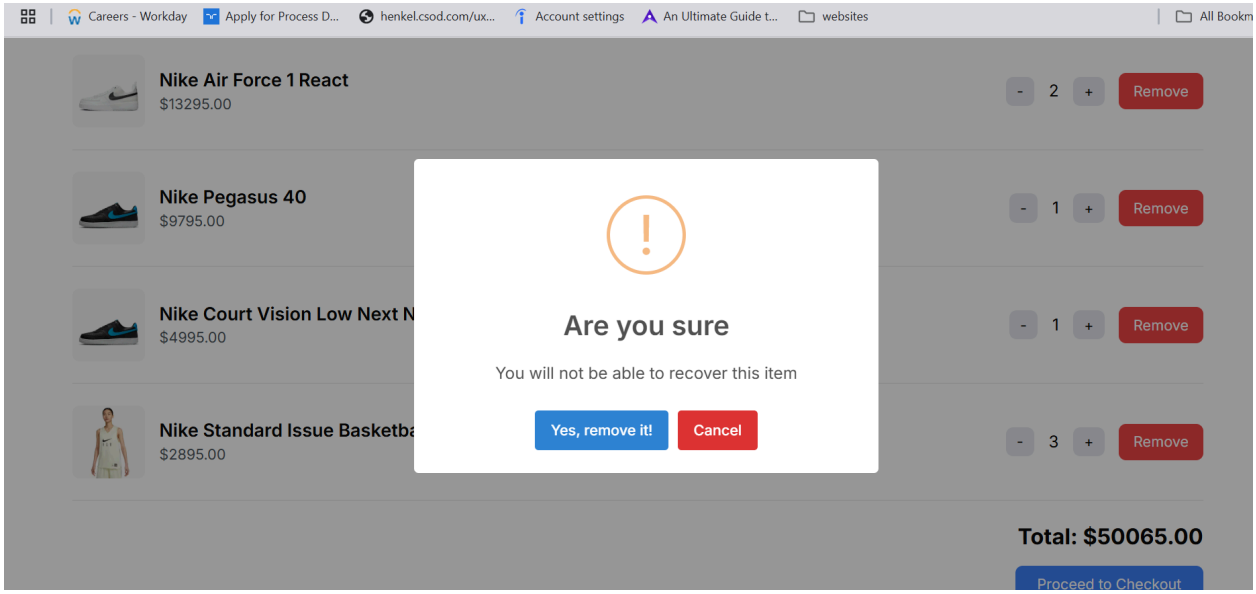
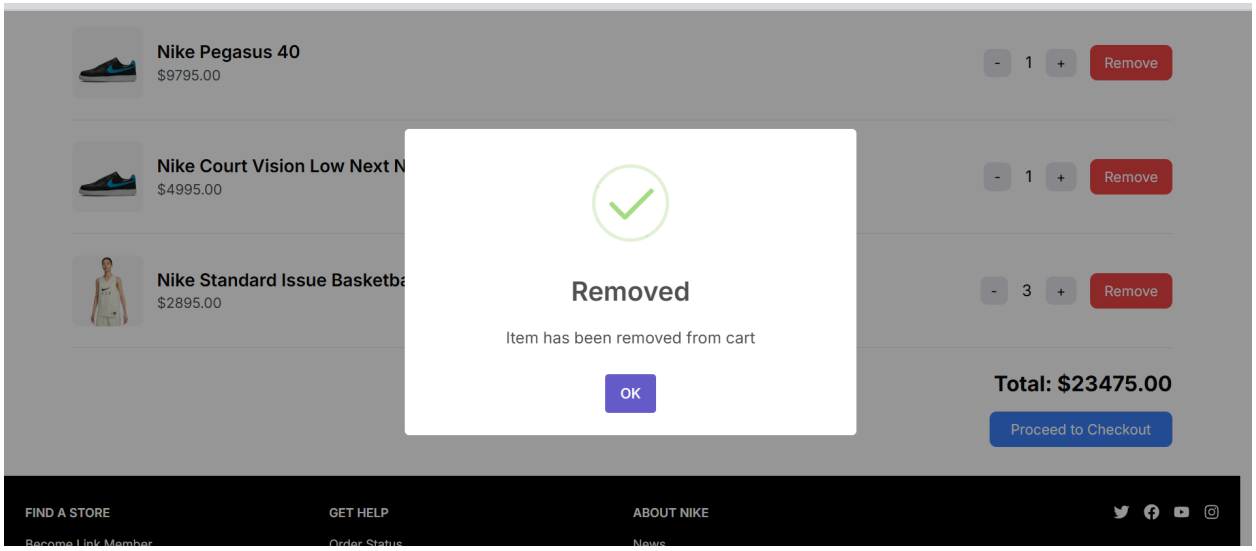
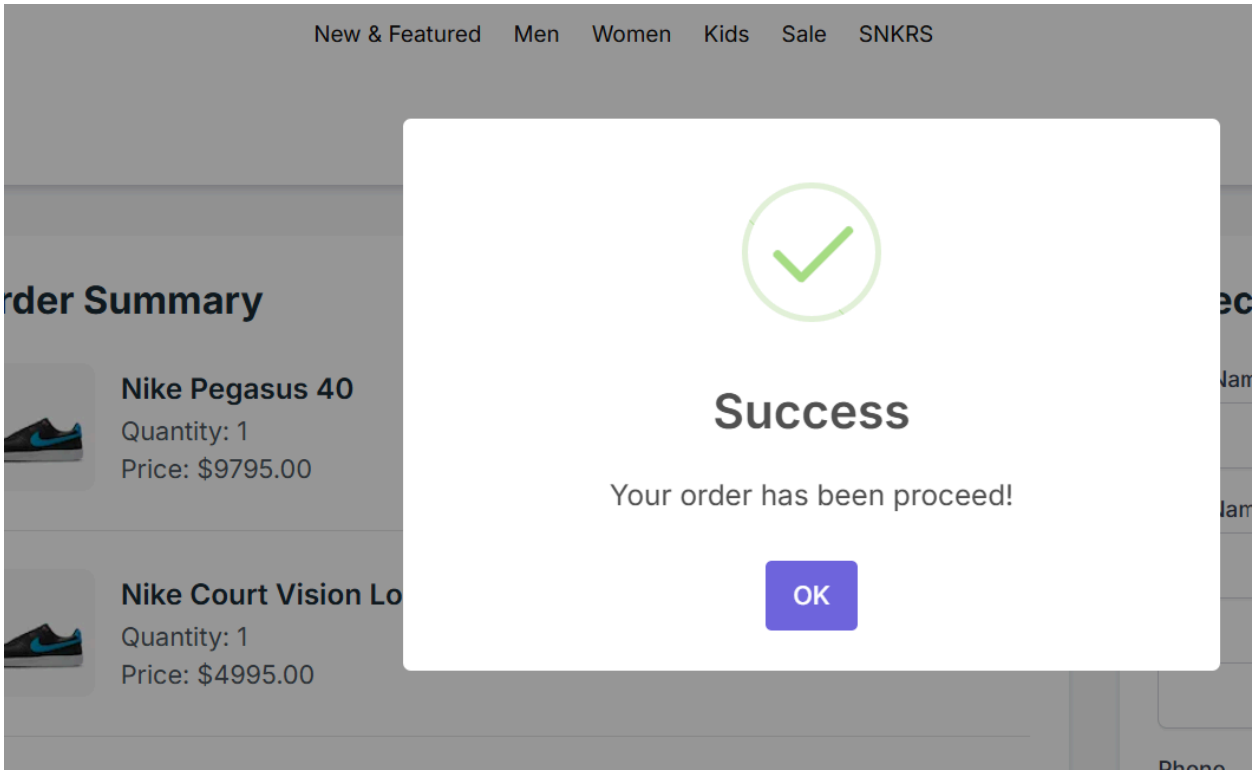

Bano Qabil
Bano Qabil Course Outline | Co
E Commerce
localhost:3000/products
Careers - Workday
Apply for Process D...
henkel.csod.com/ux...
Account settings
An Ultimate Guide t...
websites
All Bookmarks
New(500)
Shoes
Sports Bra
Tops & T-Shirts
Hoddies & Sweetshirts
Jackets
Trousers & Tights
Shorts
Tracksuits
Jumpsuits & Rompers
Skirts & Dresses
Socks
Acccessories & Equipment
Just In
Nike Air Force 1 React
$13295
Men's Shoes
White
Stocks : 12 pcs
Add To Cart
Just In
Nike Standard Issue Basketball Jersey
$2895
Women's Basketball Jersey
White
Stocks : 40 pcs
Add To Cart
Nike Standard Issue Basketball Jersey has been added to cart
Nike Court Vision Low
$5695
Men's Shoes
White
Stocks : 25 pcs
Add To Cart
Nike Air Max 270
$13295
Men's Shoes
Black
Stocks : 20 pcs
Add To Cart
Search
7:29 PM
2/8/2025


localhost:3000/cart
Careers - Workday
Apply for Process D...
henkel.csod.com/ux...
Account settings
An Ultimate Guide t...
websites
All Bookmarks
Nike Air Force 1 React
$13295.00
- 2 + Remove
Nike Pegasus 40
$9795.00
- 1 + Remove
Nike Court Vision Low Next Nature
$4995.00
- 1 + Remove
Nike Standard Issue Basketball Jersey
$2895.00
- 3 + Remove
Total: $50065.00
Proceed to Checkout


Careers - Workday
Apply for Process D...
henkel.csod.com/ux...
Account settings
An Ultimate Guide t...
websites
All Bookm
Nike Air Force 1 React
$13295.00
- 2 + Remove
Nike Pegasus 40
$9795.00
- 1 + Remove
Nike Court Vision Low Next N
$4995.00
- 1 + Remove
Nike Standard Issue Basketba
$2895.00
- 3 + Remove
Are you sure
You will not be able to recover this item
Yes, remove it!
Cancel
Total: $50065.00
Proceed to Checkout


Nike Pegasus 40
$9795.00
- 1 + Remove
Nike Court Vision Low Next N
$4995.00
- 1 + Remove
Nike Standard Issue Basketba
$2895.00
- 3 + Remove
Removed
Item has been removed from cart
OK
Total: $23475.00
Proceed to Checkout
FIND A STORE
GET HELP
ABOUT NIKE
News



Order Summary

Nike Pegasus 40
Quantity: 1
Price: $9795.00

Nike Court Vision Low Next Nature
Quantity: 1
Price: $4995.00

Nike Standard Issue Basketball Jersey
Quantity: 3
Price: $8685.00

Subtotal	$23475.00
Discount	-$0.00

Checkout Details

First Name

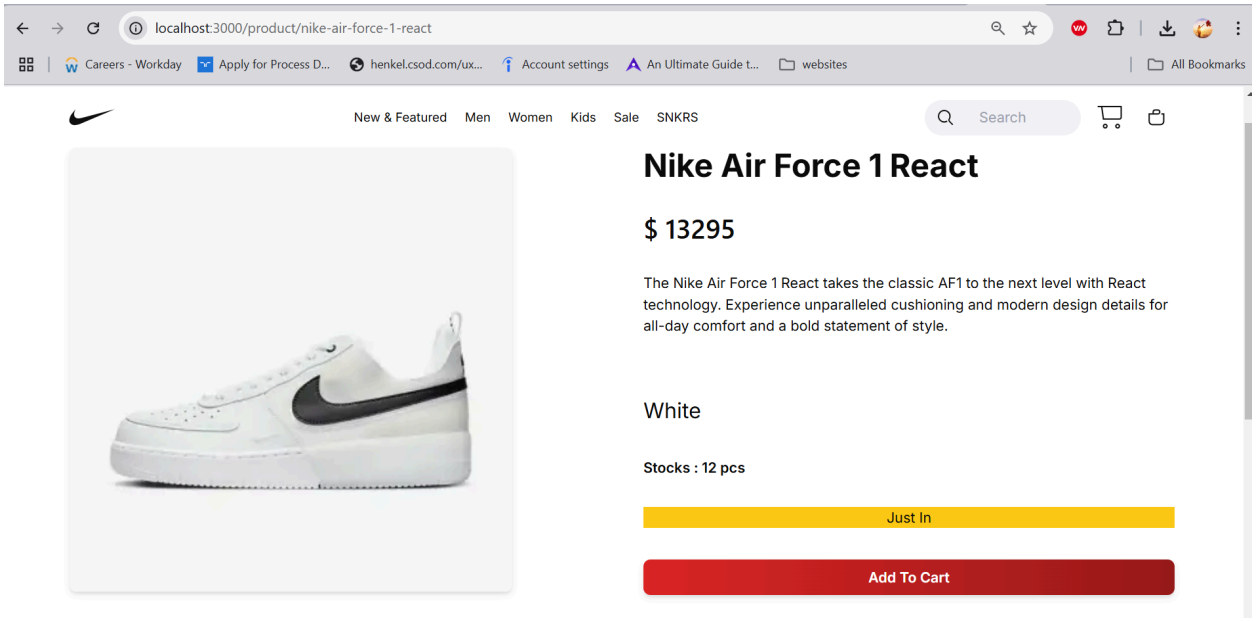
Last Name

Email

Phone

Address

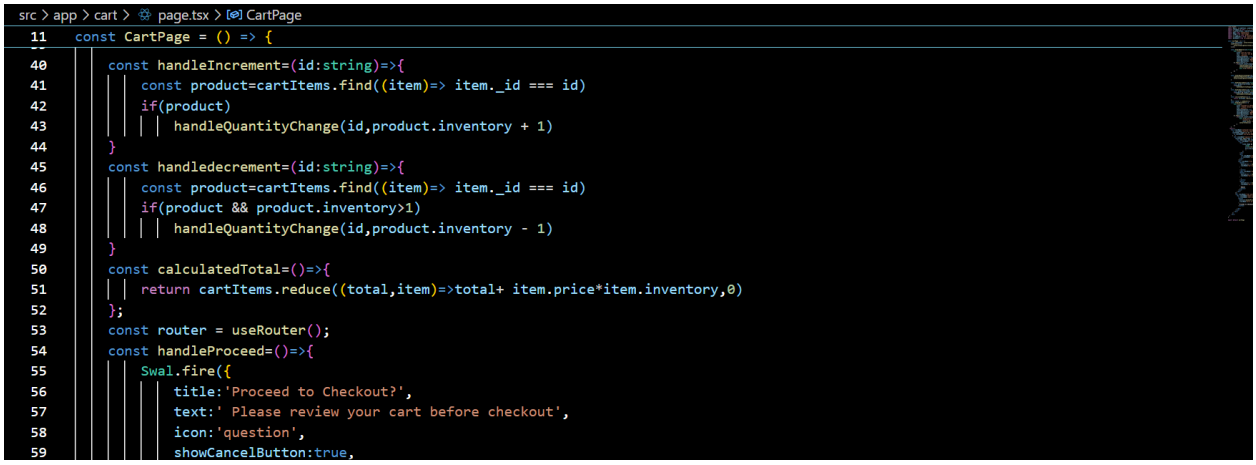
Zip Code



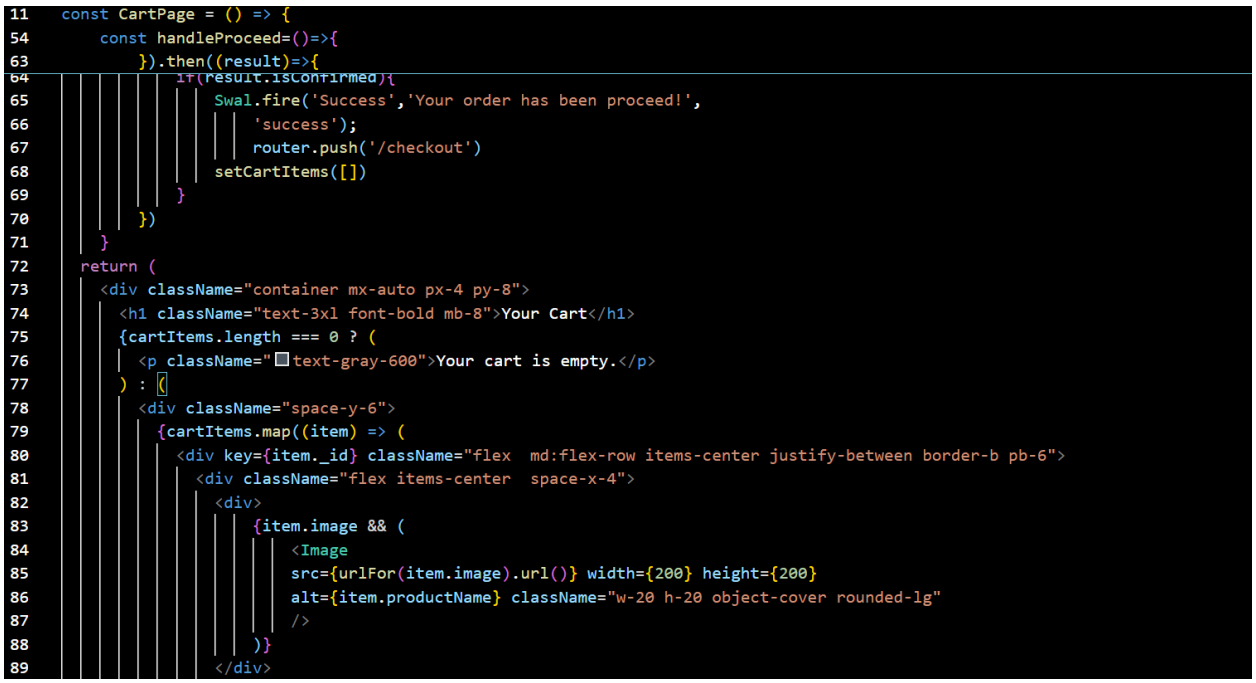

```tsx
'use client'
import React, { useEffect, useState } from 'react'
import { Product } from '../../../Types/products'
import { getCartItems, removeFromCart, updateCartQuantity } from '../actions/actions'
import Swal from 'sweetalert2'
import Image from 'next/image'
import { urlFor } from '@/sanity/lib/image'
import { useRouter } from 'next/navigation'
import AuthGuard from '@/components/AuthGuard'

const CartPage = () => {
    const [cartItems, setCartItems] = useState<Product[]>([])
    useEffect(()=>{
        setCartItems(getCartItems())
    },[]);

    const handleRemove=(id:string)=>{
        Swal.fire({
            title:'Are you sure',
            text: 'You will not be able to recover this item',
            icon: 'warning',
```

src > app > cart > page.tsx > CartPage

```tsx
const CartPage = () => {
    const handleRemove=(id:string)=>{
            cancelButtonColor:'#d33',
            confirmButtonText: 'Yes, remove it!',
        }).then((result)=>{
            if(result.isConfirmed){
                removeFromCart(id)
                setCartItems(getCartItems())
                Swal.fire('Removed', 'Item has been removed from cart','success')
            }
        })
    }

    const handleQuantityChange=(id:string, quantity:number)=>{
        updateCartQuantity(id,quantity);
        setCartItems(getCartItems())
    }

    const handleIncrement=(id:string)=>{
        const product=cartItems.find((item)=> item._id === id)
        if(product)
```



```
src > app > cart > page.tsx > CartPage
11  const CartPage = () => {
40      const handleIncrement=(id:string)=>{
41          const product=cartItems.find((item)=> item._id === id)
42          if(product)
43              handleQuantityChange(id,product.inventory + 1)
44      }
45      const handledecrement=(id:string)=>{
46          const product=cartItems.find((item)=> item._id === id)
47          if(product && product.inventory>1)
48              handleQuantityChange(id,product.inventory - 1)
49      }
50      const calculatedTotal=()=>{
51          return cartItems.reduce((total,item)=>total+ item.price*item.inventory,0)
52      };
53      const router = useRouter();
54      const handleProceed=()=>{
55          Swal.fire({
56              title:'Proceed to Checkout?',
57              text:' Please review your cart before checkout',
58              icon:'question',
59              showCancelButton:true,
```



```
11  const CartPage = () => {
54      const handleProceed=()=>{
63          }).then((result)=>{
64              if(result.isConfirmed){
65                  Swal.fire('Success','Your order has been proceed!',
66                      'success');
67                  router.push('/checkout')
68                  setCartItems([])
69              }
70          })
71      }
72      return (
73          <div className="container mx-auto px-4 py-8">
74              <h1 className="text-3xl font-bold mb-8">Your Cart</h1>
75              {cartItems.length === 0 ? (
76                  <p className="text-gray-600">Your cart is empty.</p>
77              ) : (
78                  <div className="space-y-6">
79                      {cartItems.map((item) => (
80                          <div key={item._id} className="flex  md:flex-row items-center justify-between border-b pb-6">
81                              <div className="flex items-center  space-x-4">
82                                  <div>
83                                      {item.image && (
84                                          <Image
85                                          src={urlFor(item.image).url()} width={200} height={200}
86                                          alt={item.productName} className="w-20 h-20 object-cover rounded-lg"
87                                          />
88                                      )}
89                                  </div>
```

src > app > products > page.tsx > ...

```tsx
"use client";
import Image from "next/image";
import React, { useEffect, useState } from "react";
import { Product } from "../../../Types/products";
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
import Link from "next/link";
import { allProducts } from "@/sanity/lib/queries";
import { addToCart } from "../actions/actions";
import Swal from "sweetalert2";

const Products = () => {
  const [product, setProduct] = useState<Product[]>([]);

  useEffect(() => {
    async function fetchproduct() {
      const fetchedProduct: Product[] = await client.fetch(allProducts);
      setProduct(fetchedProduct);
    }
    fetchproduct();
  }, []);
  const handleAddToCart=(e:React.MouseEvent, product:Product)=>{
    e.preventDefault()
    Swal.fire({
      position : 'top-right',
      icon : "success",
      title: `${product.productName} has been added to cart`,
      showConfirmButton: false,
      timer:2000
```

src > app > product > [slug] > page.tsx > ...

```tsx
'use client'
import { client } from "@/sanity/lib/client";
import { Product } from "../../../../Types/products";
import { groq } from "next-sanity";
import Image from "next/image";
import { urlFor } from "@/sanity/lib/image";
import Swal from "sweetalert2";
import { addToCart } from "@/app/actions/actions";

interface ProductPageProps {
  params: Promise<{ slug: string }>;
}

async function getProduct(slug: string): Promise<Product> {
  return client.fetch(
    groq`*[_type== "product" && slug.current == $slug][0]{
        _id,
        productName,
        _type,
        image,
        price,
        colors,
        status,
        description,
        inventory,
        }`,
    { slug }
  );
}
```

src > app > product > [slug] > page.tsx > ...

```tsx
export default async function ProductPage({ params }: ProductPageProps) {
  const { slug } = await params;
  const product = await getProduct(slug);

  const handleAddToCart=(e:React.MouseEvent, product:Product)=>{
    e.preventDefault()
    Swal.fire({
      position : 'top-right',
      icon : "success",
      title: `${product.productName} has been added to cart`,
      showConfirmButton: false,
      timer:2000,

    })

    addToCart(product)

  }

  return (
    <div className="max-w-7xl mx-auto px-4">
      <div className="grid grid-cols-1 md:grid-cols-2 gap-12">
        <div className="aspect-square">
          {product.image && (
            <Image
              src={urlFor(product.image).url()}
              alt={product.productName}
              width={500}
```

Technical Report: E-Commerce Web Application Development

1. Steps Taken to Build and Integrate Components

- **Project Setup:** Initialized a **Next.js** project with TypeScript and integrated **Sanity CMS** for product management.
- **Product Page Development:**
  - Created an API call using **GROQ queries** to fetch product data from **Sanity CMS**.
  - Displayed product details (image, price, description, inventory, etc.) using **Next.js Image component** for optimized performance.
  - Implemented an **"Add to Cart"** function with **SweetAlert2** notifications.
- **Cart Page Development:**
  - Retrieved cart data from **local storage** and displayed products dynamically.
  - Implemented functions to **increase/decrease item quantity**, **remove items**, and **calculate total cost**.
  - Used **SweetAlert2** for user confirmations and success messages.
  - Integrated **Next.js Router** for navigation to the checkout page.

---

2. Challenges Faced and Solutions Implemented

Challenges	Solutions
State Persistence: Cart items were lost on page reload.	Used **local storage** to persist cart data.
Quantity Management Bugs: Incrementing/decrementing inventory affected product availability.	Implemented functions to check **stock limits** before modifying quantities.
Dynamic Routing Issues: Fetching product details using the slug caused page errors.	Ensured proper **async/await handling** in `getProduct()` to avoid race conditions.
UI Responsiveness: Layout issues on different screen sizes.	Used **Tailwind CSS** for a fully responsive design.
Cart Management Scalability: Local storage limits flexibility for multi-user support.	Future enhancement: **Integrate backend API** for session-based cart storage.

---

3. Best Practices Followed During Development

- **Modular Code Structure:** Separated components for reusability (`ProductPage`, `CartPage`, `AuthGuard`, etc.).
- **Optimized Data Fetching:** Used **GROQ queries** for efficient data retrieval from **Sanity CMS**.
- **Performance Optimization:** Used **Next.js Image component** for better loading speed.
- **User-Friendly Interactions:** Integrated **SweetAlert2** for clear feedback and confirmations.
- **Code Maintainability:** Followed **TypeScript** best practices for type safety and scalability.