Day 4

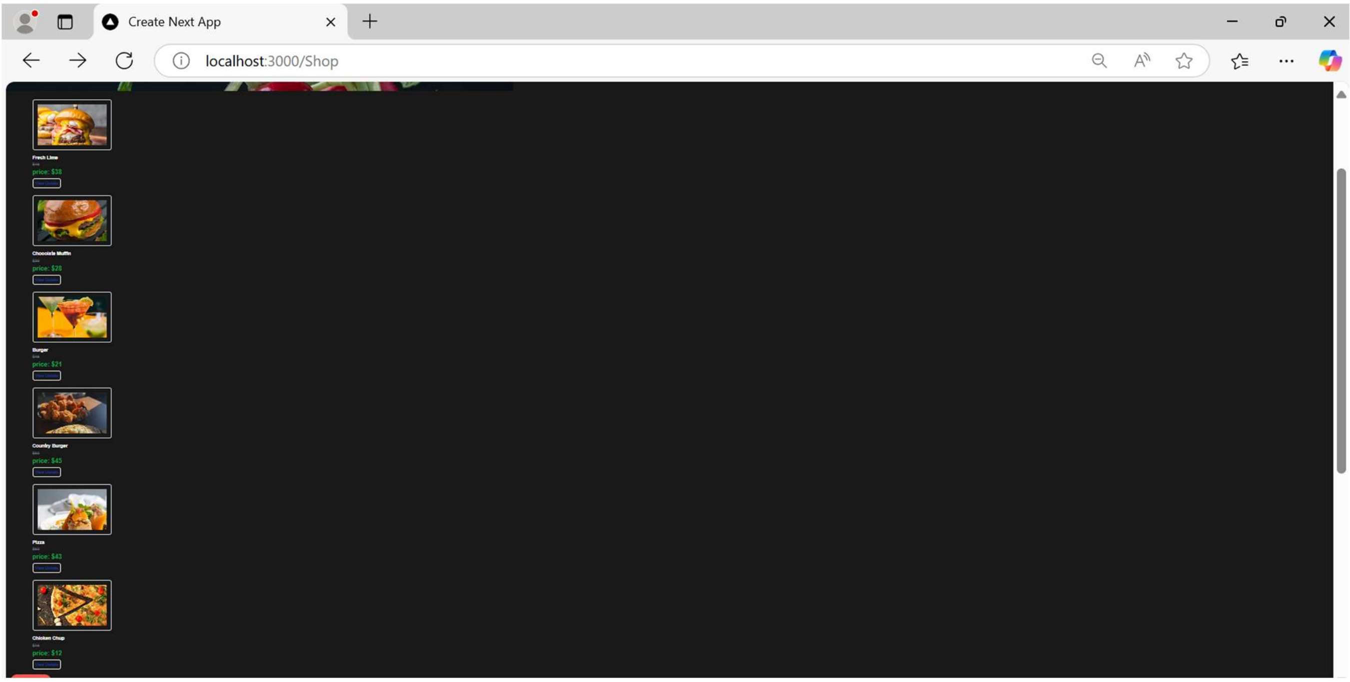
Objective:

The focus of today's task was to develop key frontend components for our marketplace application. The goal was to ensure these components are dynamic, reusable, and responsive, fetching data seamlessly from Sanity CMS or APIs.

Product Listing Component:

- Designed a grid layout to display products dynamically, including details such as:
  - **Name**
  - **Original Price**
  - **Price**
  - **Image**
  - **Product dimensions Link**

- Created a reusable ProductCard component to showcase each product. Data was retrieved using the Sanity CMS API.

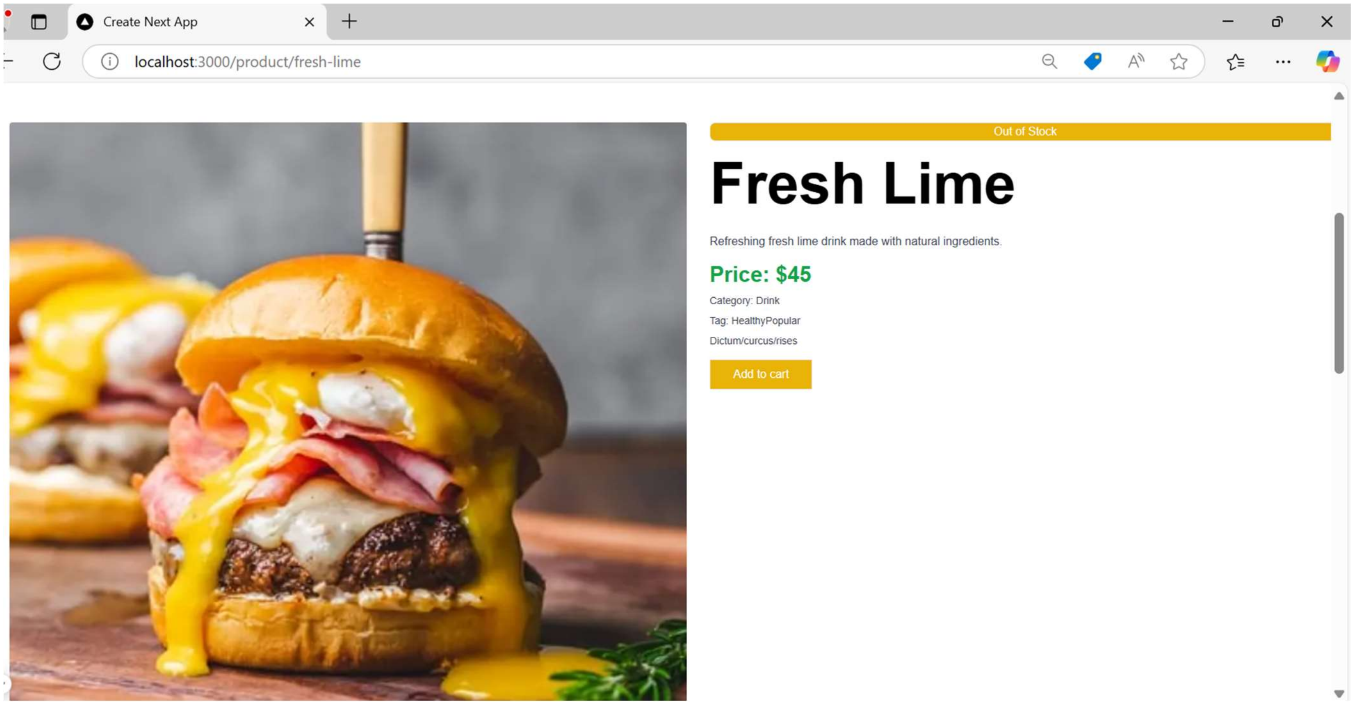


Product Detail Component:

Developed dynamic product pages using Next.js dynamic routing.

These pages include:

- Product description.
- Pricing details.
- Large size product image
- Product name.



Out of Stock

Fresh Lime

Refreshing fresh lime drink made with natural ingredients.

Price: $45

Category: Drink

Tag: HealthyPopular

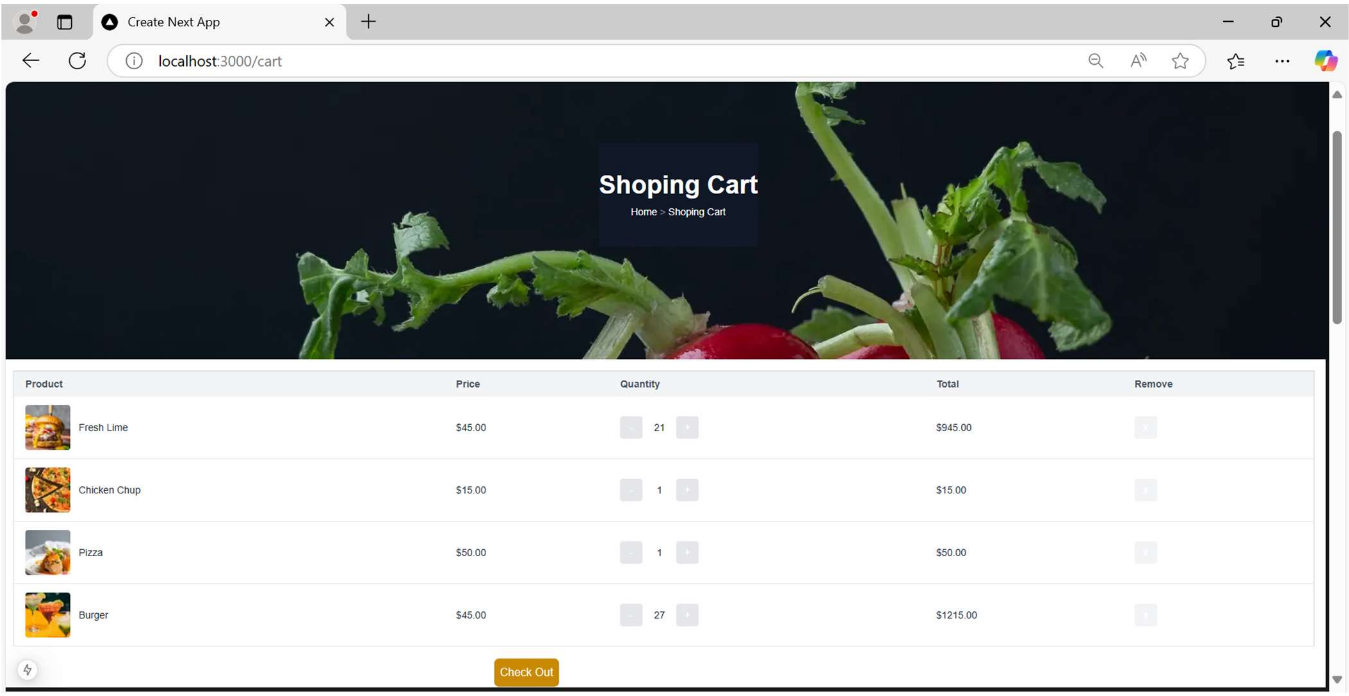
Dictum/curcus/rises

Cart Component:

- Built a functional cart that:

  Displays added items, their quantities, and the total price.

- Used React state to track and manage cart items.



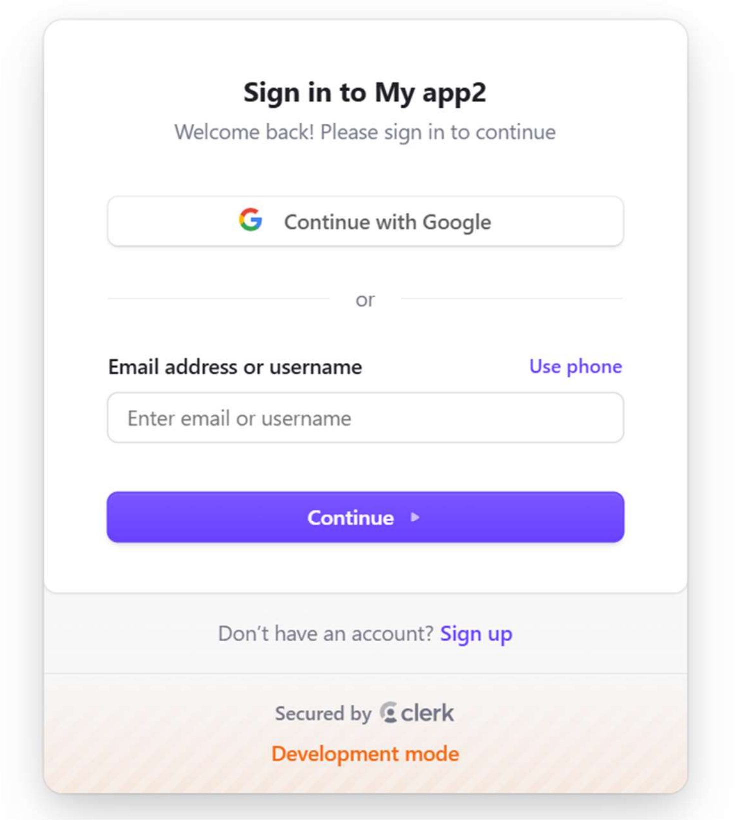
Shoping Cart

Home > Shoping Cart

Product	Price	Quantity	Total	Remove
Fresh Lime	$45.00	- 21 +	$945.00	x
Chicken Chup	$15.00	- 1 +	$15.00	x
Pizza	$50.00	- 1 +	$50.00	x
Burger	$45.00	- 27 +	$1215.00	x

User Authentication Features:

- Developed a **Sign-Up Page** and a **Login Page** for user registration and login functionality using clerk.
- User can sign in through their **google** account.
- Created a **Profile Page** to display user-specific details.



Sign in to My app2

Welcome back! Please sign in to continue

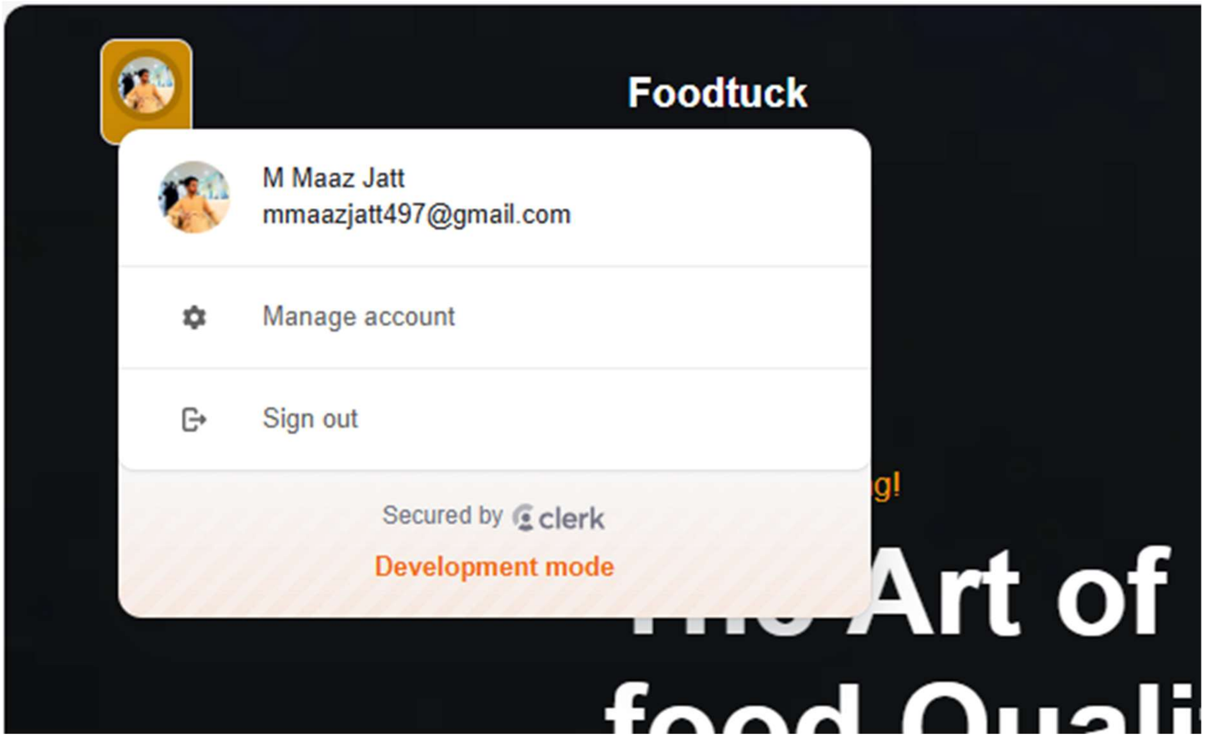
Email address or username

Enter email or username

Don't have an account? Sign up

Secured by clerk

Development mode



M Maaz Jatt
mmaazjatt497@gmail.com

Checkout Flow Component:

I added a checkout flow component to the website to make the purchasing process more efficient.

The component gathers essential user information, including:

- First name
- Last name
- Address
- Cart items

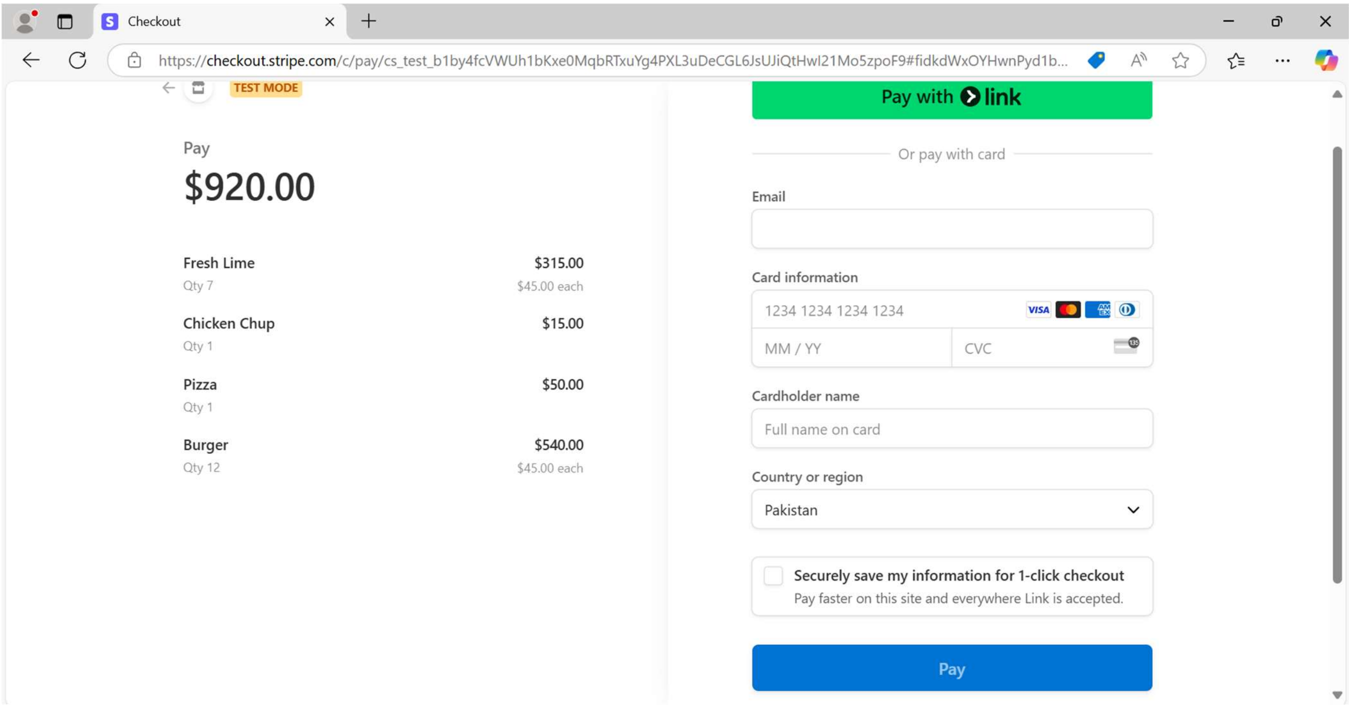
All user data is securely stored on clerk to ensure reliability and scalability.

I also implemented an order confirmation page that:

- Retrieves the stored user data from Clerk.
- Shows a personalized thank-you message with the user's name.
- Displays the user's entered details and a summary of the ordered products.

This flow streamlines the checkout experience and provides a personal touch to improve the overall user experience.

Check out page:



Checkout
https://checkout.stripe.com/c/pay/cs_test_b1by4fcVWUh1bKxe0MqbRTxuYg4PXL3uDeCGL6JsUJiQtHwl21Mo5zpoF9#fidkdWxOYHwnPyd1b...
TEST MODE

Pay

$920.00

Fresh Lime Qty 7	$315.00 $45.00 each
Chicken Chup Qty 1	$15.00
Pizza Qty 1	$50.00
Burger Qty 12	$540.00 $45.00 each

Pay with link

Or pay with card

Email

Card information

1234 1234 1234 1234

MM / YY CVC

Cardholder name

Full name on card

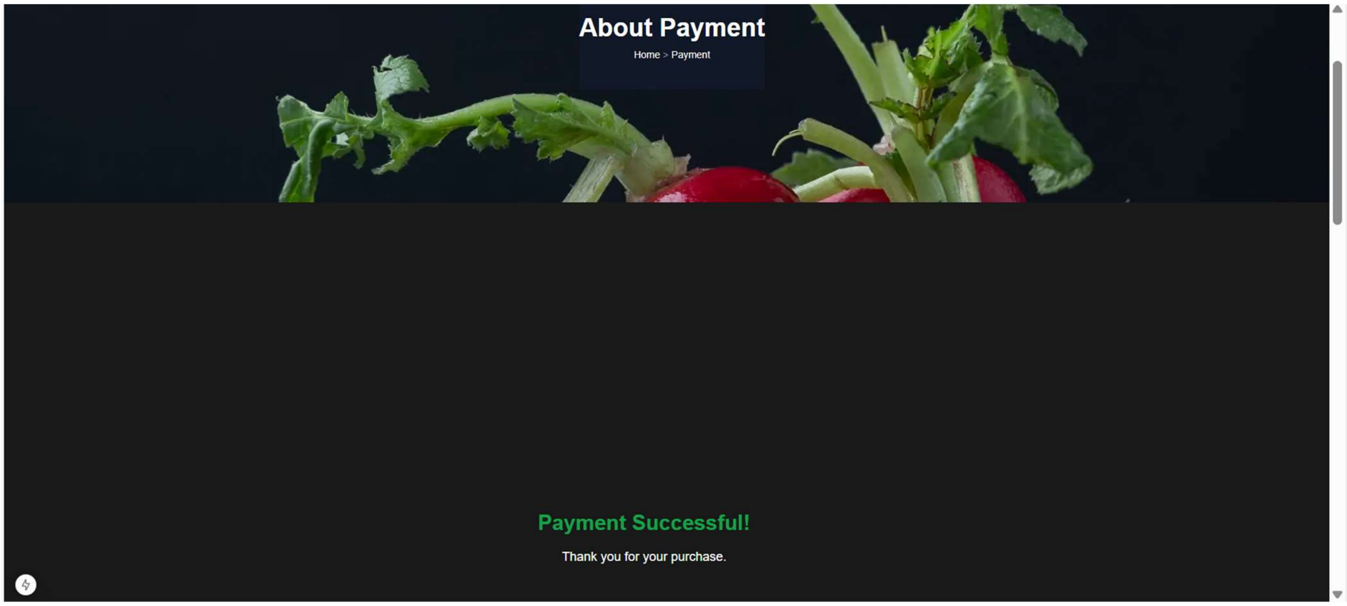
Country or region

Pakistan

□ Securely save my information for 1-click checkout
Pay faster on this site and everywhere Link is accepted.

Pay

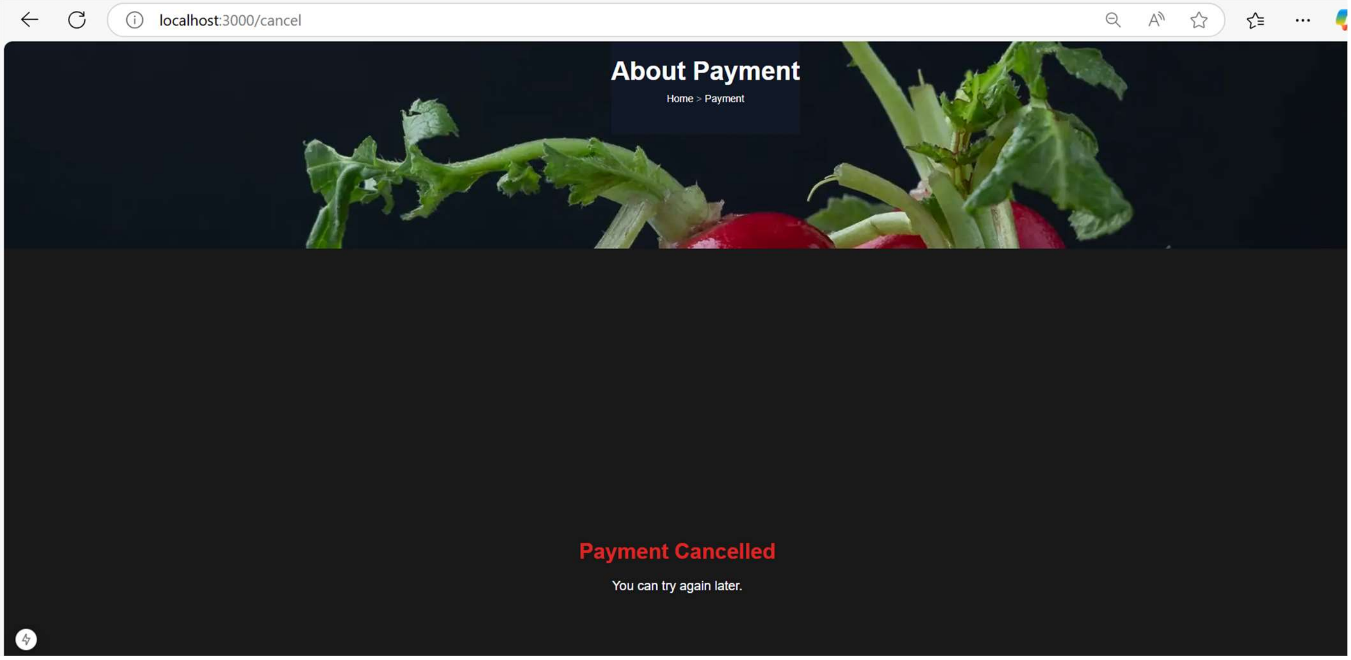
Payment confirmation page:



Payment Successful!

Thank you for your purchase.

Payment Canceletion page:


localhost:3000/cancel

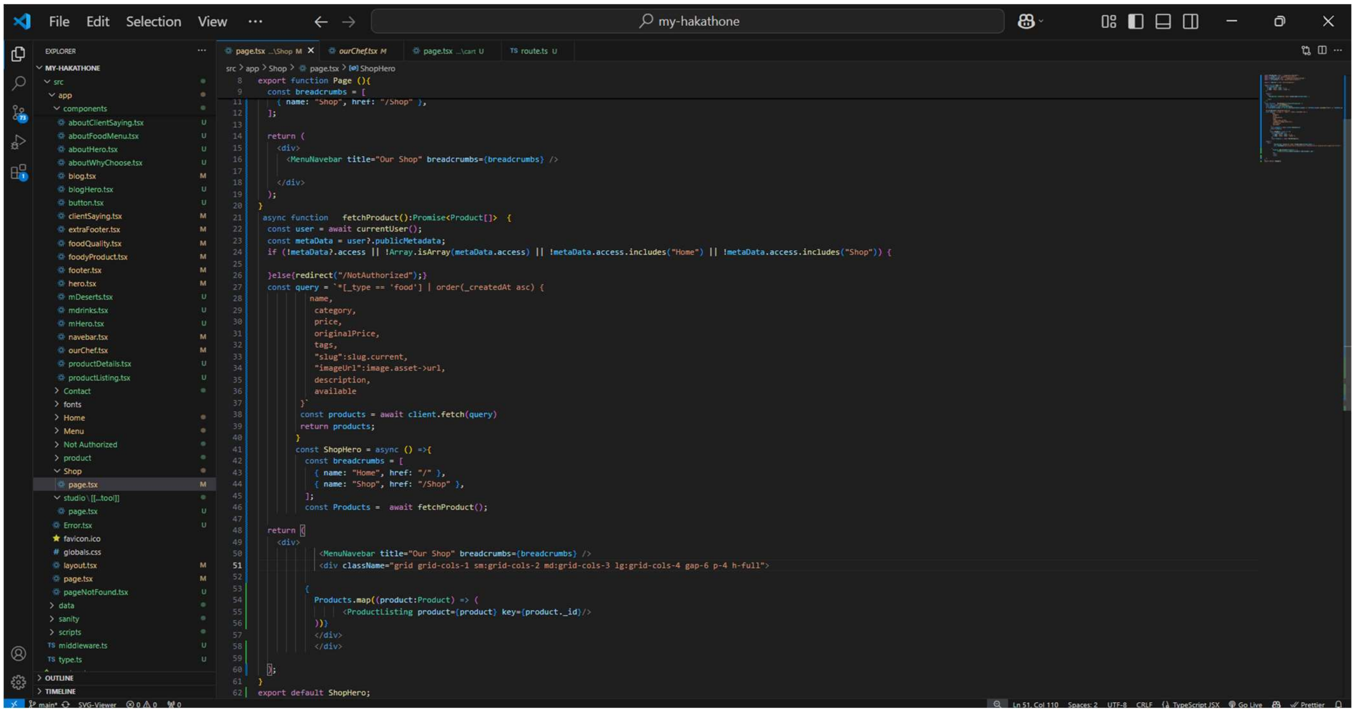
About Payment

Home > Payment

Payment Cancelled

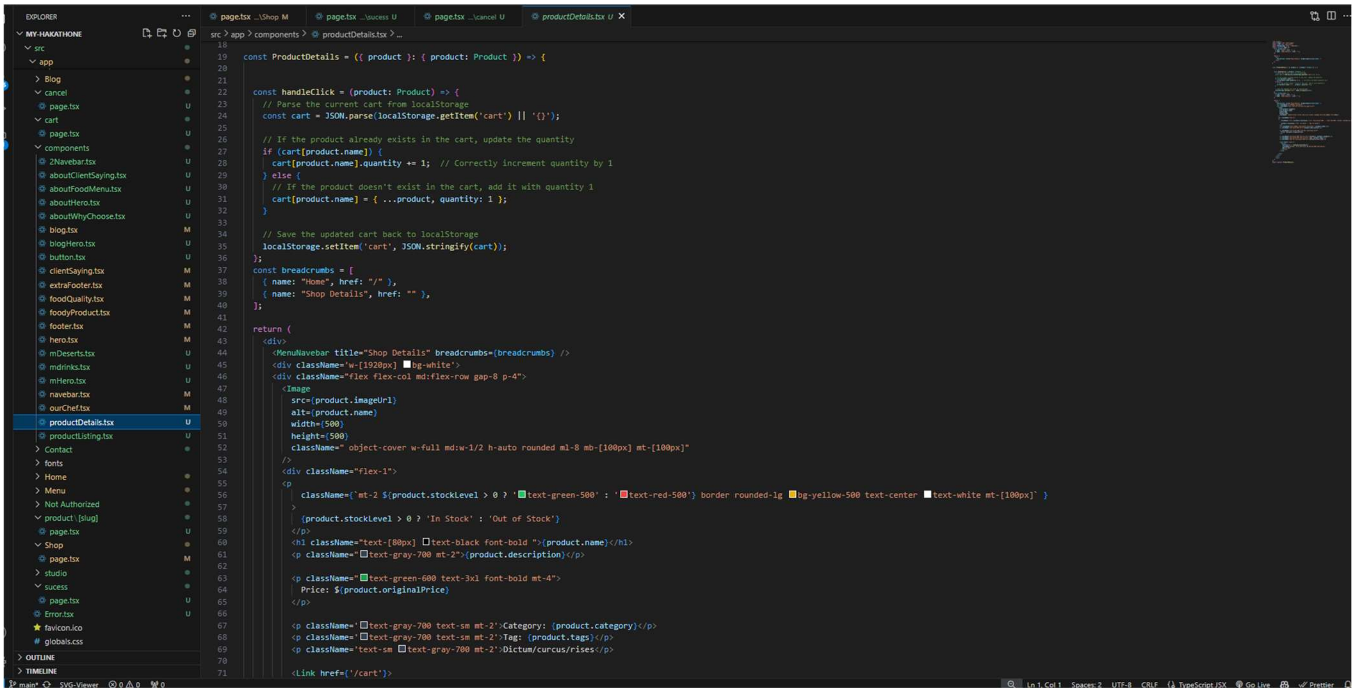
You can try again later.

Product list component code snippet

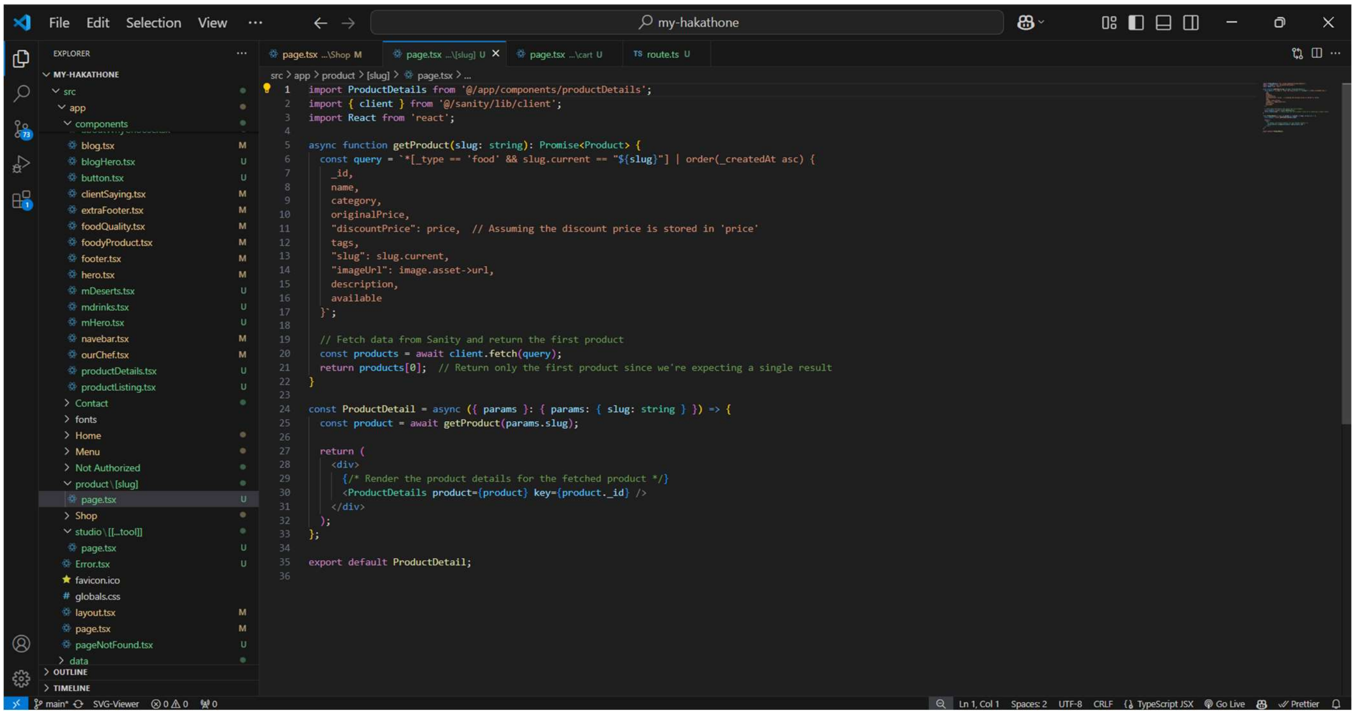


```tsx
export function Page (){
  const breadcrumbs = [
    { name: "Shop", href: "/Shop" },
  ];

  return (
    <div>
      <MenuNavebar title="Our Shop" breadcrumbs={breadcrumbs} />

    </div>
  );
}
 async function   fetchProduct():Promise<Product[]>  {
  const user = await currentUser();
  const metaData = user?.publicMetadata;
  if (!metaData?.access || !Array.isArray(metaData.access) || !metaData.access.includes("Home") || !metaData.access.includes("Shop")) {

  }else{redirect("/NotAuthorized");}
  const query = `*[_type == 'food'] | order(_createdAt asc) {
           name,
            category,
            price,
            originalPrice,
            tags,
            "slug":slug.current,
            "imageUrl":image.asset->url,
            description,
            available
         }`
         const products = await client.fetch(query)
         return products;
        }
        const ShopHero = async () =>{
          const breadcrumbs = [
            { name: "Home", href: "/" },
            { name: "Shop", href: "/Shop" },
          ];
          const Products =  await fetchProduct();

  return (
    <div>
             <MenuNavebar title="Our Shop" breadcrumbs={breadcrumbs} />
             <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6 p-4 h-full">

          {
            Products.map((product:Product) => (
                  <ProductListing product={product} key={product._id}/>
            ))}
            </div>
            </div>

  );
}
export default ShopHero;
```

Product details component code snippet



```tsx

const ProductDetails = ({ product }: { product: Product }) => {


  const handleClick = (product: Product) => {
    // Parse the current cart from localStorage
    const cart = JSON.parse(localStorage.getItem('cart') || '{}');

    // If the product already exists in the cart, update the quantity
    if (cart[product.name]) {
      cart[product.name].quantity += 1;  // Correctly increment quantity by 1
    } else {
      // If the product doesn't exist in the cart, add it with quantity 1
      cart[product.name] = { ...product, quantity: 1 };
    }

    // Save the updated cart back to localStorage
    localStorage.setItem('cart', JSON.stringify(cart));
  };
  const breadcrumbs = [
    { name: "Home", href: "/" },
    { name: "Shop Details", href: "" },
  ];

  return (
    <div>
      <MenuNavebar title="Shop Details" breadcrumbs={breadcrumbs} />
      <div className='w-[1920px] bg-white'>
      <div className="flex flex-col md:flex-row gap-8 p-4">
        <Image
          src={product.imageUrl}
          alt={product.name}
          width={500}
          height={500}
          className=" object-cover w-full md:w-1/2 h-auto rounded ml-8 mb-[100px] mt-[100px]"
        />
        <div className="flex-1">
        <p
          className={`mt-2 ${product.stockLevel > 0 ? 'text-green-500' : 'text-red-500'} border rounded-lg bg-yellow-500 text-center text-white mt-[100px]` }
          >
          {product.stockLevel > 0 ? 'In Stock' : 'Out of Stock'}
          </p>
          <h1 className="text-[80px] text-black font-bold ">{product.name}</h1>
          <p className="text-gray-700 mt-2">{product.description}</p>

          <p className="text-green-600 text-3xl font-bold mt-4">
            Price: ${product.originalPrice}
          </p>

          <p className='text-gray-700 text-sm mt-2'>Category: {product.category}</p>
          <p className='text-gray-700 text-sm mt-2'>Tag: {product.tags}</p>
          <p className='text-sm text-gray-700 mt-2'>Dictum/curcus/rises</p>

          <Link href={'/cart'}>
```



```tsx
import ProductDetails from '@/app/components/productDetails';
import { client } from '@/sanity/lib/client';
import React from 'react';

async function getProduct(slug: string): Promise<Product> {
  const query = `*[_type == 'food' && slug.current == "${slug}"] | order(_createdAt asc) {
    _id,
    name,
    category,
    originalPrice,
    "discountPrice": price,  // Assuming the discount price is stored in 'price'
    tags,
    "slug": slug.current,
    "imageUrl": image.asset->url,
    description,
    available
  }`;

  // Fetch data from Sanity and return the first product
  const products = await client.fetch(query);
  return products[0];  // Return only the first product since we're expecting a single result
}

const ProductDetail = async ({ params }: { params: { slug: string } }) => {
  const product = await getProduct(params.slug);

  return (
    <div>
      {/* Render the product details for the fetched product */}
      <ProductDetails product={product} key={product._id} />
    </div>
  );
};

export default ProductDetail;

```