

DAY 4 IMPLEMENTATION REPORT

Created by Ismat Fatima # 333789

Introduction

FoodTuck is a Q-commerce (Quick Commerce) restaurant website designed to make fast food delivery easy and quick. The purpose of this platform is to deliver customers' favorite food items to their doorsteps in a very short time. Unlike traditional e-commerce, Q-commerce emphasizes rapid delivery services, focusing on delivering orders in minutes rather than hours or days.

FoodTuck is not just an online marketplace; it is a specialized restaurant service that promises quick delivery, ensuring customers can satisfy their cravings promptly. The primary functions of this website include:

Order Placement:

Customers can place orders for their favorite food items online.

Quick Delivery:

FoodTuck's logistic partners ensure orders are delivered in the shortest possible time.

Menu Browsing:

Customers can browse a menu featuring various cuisines and dishes.

User Profiles:

Personalized profiles for regular customers allow them to track their previous orders and favorite dishes.

Real-time Order Tracking:

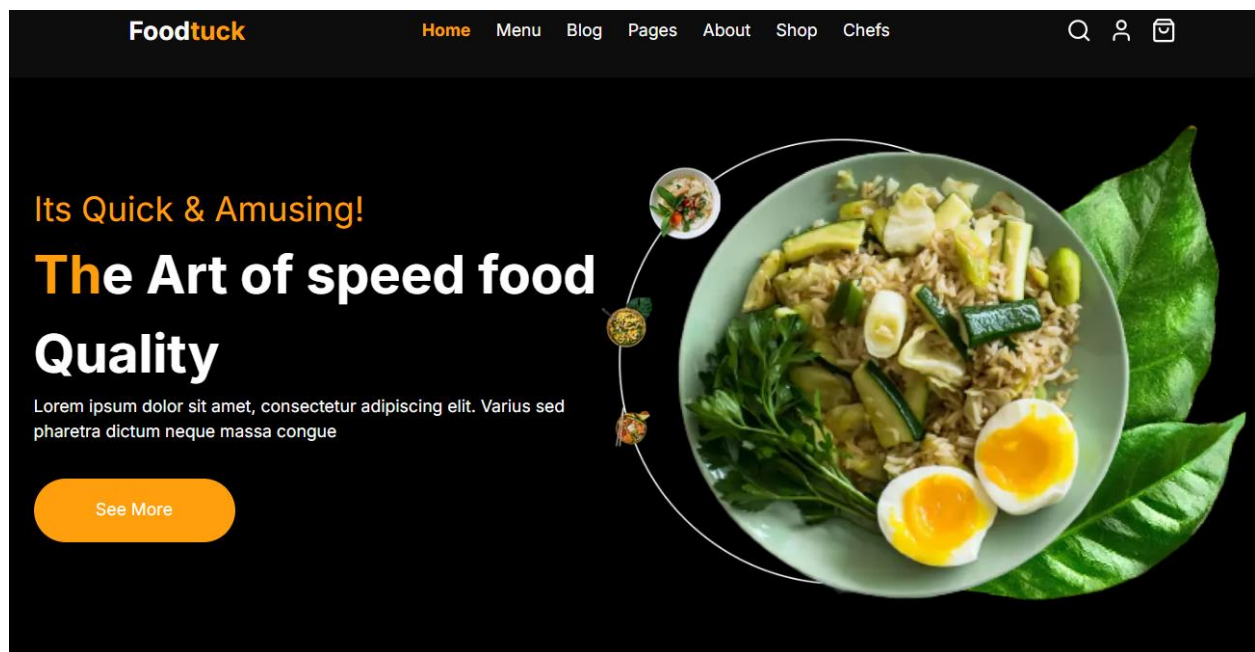
Customers can track their orders in real-time to know exactly where their order is. The design and functionality of this website are planned to provide a

seamless user experience, processing and delivering orders quickly and efficiently.

Dynamic Routing Details:

Home Page (/):

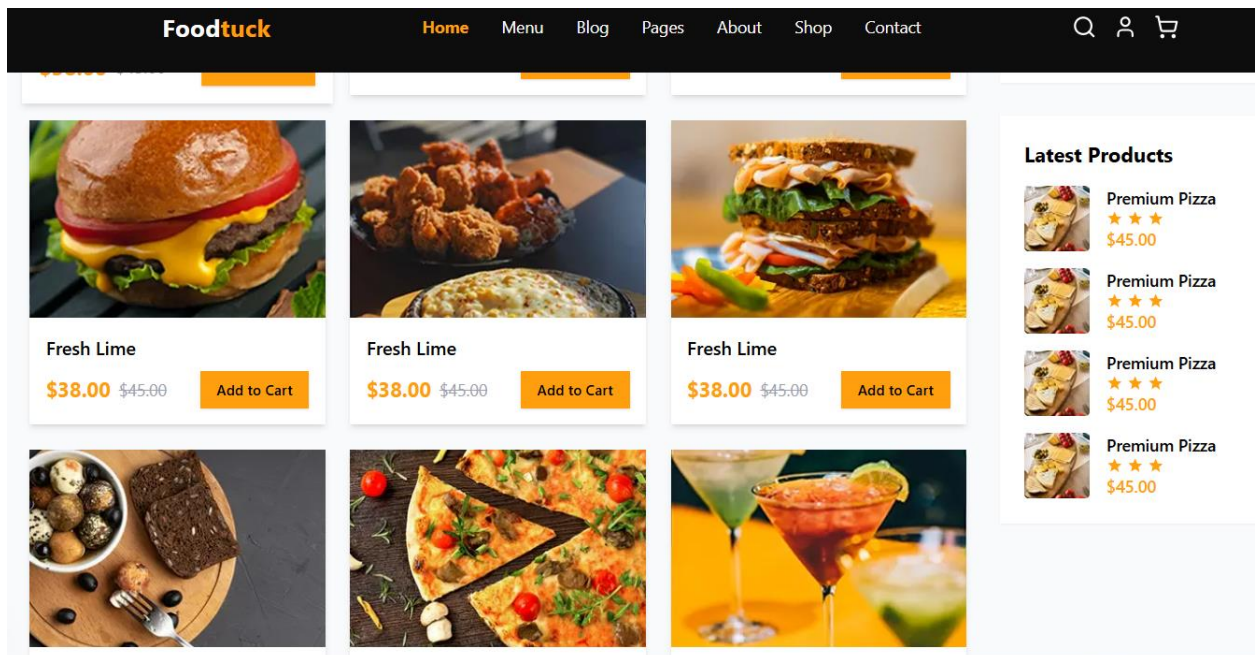
Static route for the homepage showing the general content.



Product Page (/product/[id]):

Uses **dynamic routing** to load product details based on the `id` parameter.

Each product's unique ID is fetched from the database or CMS to render specific product information.



```

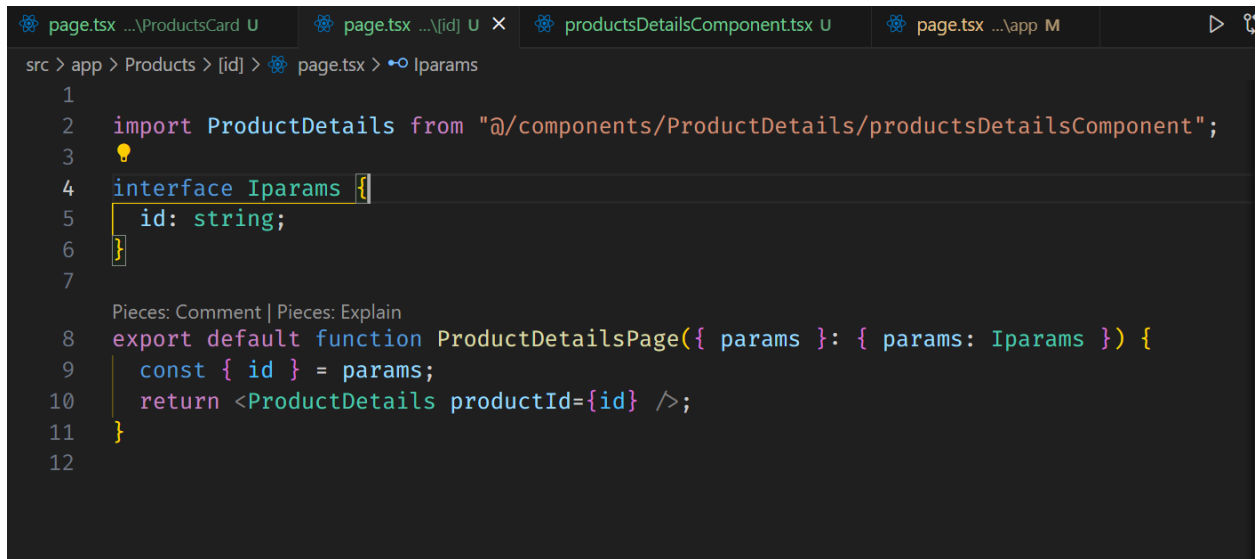
src > components > ProductDetails > productsDetailsComponent.tsx > IProducts > image
1  "use client";
2  // import { useCart } from "@app/cart/context/CartContext";
3  import { client } from "@sanity/lib/client";
4  import Image from "next/image";
5  import React, { useEffect, useState } from "react";
6  import { Montserrat } from "next/font/google";
7  import { urlFor } from "@sanity/lib/image";
8  import { CiHeart } from "react-icons/ci";
9  import { FaEye, FaStar } from "react-icons/fa";
10 import { IoCartOutline } from "react-icons/io5";
11 // import ProductCardSlide from "../ProductComponent/productCardSlider";
12
13 const montserrat = Montserrat({ subsets: ["latin"], weight: ["700"] });
14
15 interface IProducts {
16   id: string;
17   heading: string;
18   subheading: string;
19   image: {
20     src: string;

```

```

page.tsx ...ProductsCard U  page.tsx ...[id] U  productsDetailsComponent.tsx U X  page.tsx ...app M
src > components > ProductDetails > productsDetailsComponent.tsx > IProducts > image
31 const ProductDetails = ({ productId }: { productId: string }) => {
32   const [result, setResult] = useState<IProducts | null>(null);
33   const [selectedColor, setSelectedColor] = useState<string>("");
34   const [loading, setLoading] = useState<boolean>(true);
35   // const { addToCart } = useCart();
36
37   // Array of colors to use
38   const colors = [
39     { name: "blue", class: "bg-myBlue" },
40     { name: "green", class: "bg-myDarkGreen" },
41     { name: "orange", class: "bg-myOrange" },
42     { name: "dark", class: "bg-myDark" },
43   ];
44
45   const handleColorChange = (colorClass: string) => {
46     setSelectedColor(colorClass);
47   };
48

```



```
1
2 import ProductDetails from "@components/ProductDetails/productsDetailsComponent";
3
4 interface Iparams {
5   id: string;
6 }
7
8 export default function ProductDetailsPage({ params }: { params: Iparams }) {
9   const { id } = params;
10  return <ProductDetails productId={id} />;
11 }
12
```

The screenshot shows a code editor with four tabs: 'page.tsx ...\ProductsCard U', 'page.tsx ...\[id] U', 'productsDetailsComponent.tsx U', and 'page.tsx ...\app M'. The active tab is 'page.tsx ...\[id] U'. The code is in TypeScript and defines an interface 'Iparams' with a property 'id' of type 'string'. It then exports a default function 'ProductDetailsPage' that takes a 'params' object of type '{ params: Iparams }' and returns a JSX element '<ProductDetails productId={id} />'. The code is numbered 1 through 12.

Cart Page (/cart):

A dynamic route that dynamically renders items based on the user's session or state.

```
.env.local TS fetch.ts page.tsx ...Product X page.tsx ...app
src > app > Product > page.tsx > default
1  "use client";
2  import React, { useEffect, useState } from "react";
3  import sanityClient from "@sanity/client";
4  import Image from "next/image";
5
6  const sanity = sanityClient({
7    projectId: "ci05x4q0",
8    dataset: "production",
9    apiVersion: "2023-01-01",
10    useCdn: true,
11  });
12
13  interface Product {
14    discountPercentage: any;
15    _id: string;
16    title: string;
17    price: number;
18    description: string;
19    imageUrl: string;
20    productImage: {
```

```
.env.local page.tsx X
src > app > Product > page.tsx > default
26  }
27
28  const ProductCards: React.FC = () => {
29    const [products, setProducts] = useState<Product[]>([]);
30    const [cart, setCart] = useState<Product[]>([]);
31
32    const fetchProducts = async () => {
33      try {
34        const query = `*[type = "product"]{
35          _id,
36          title,
37          price,
38          description,
39          discountPercentage,
40          "imageUrl": productImage.asset->url,
41          tags
42        }`;
43
44        const data = await sanity.fetch(query);
45        setProducts(data);
```

Sort By:

Newest

Price: Low to High

Price: High to Low

Show:

12 Items

24 Items

36 Items



Fresh Lime

\$38.00 ~~\$45.00~~

Add to Cart



Fresh Lime

\$38.00 ~~\$45.00~~

Add to Cart



Fresh Lime

\$38.00 ~~\$45.00~~




Add to Cart



Checkout Page (/checkout):

A static route that dynamically handles user-specific information and cart data during the checkout process.

Foodtuck[Home](#)[Menu](#)[Blog](#)[Pages](#)[About](#)[Shop](#)[Chefs](#)



Checkout

Home / Checkout

Shipping Address

First Name


Email


Company


Last Name

Phone Number




Zip Code

**Chicken Tikka Kabab**
150 gm net
50\$

**Chicken Tikka Kabab**
150 gm net
50\$

**Chicken Tikka Kabab**
150 gm net

Foodtuck[Home](#)[Menu](#)[Blog](#)[Pages](#)[About](#)[Shop](#)[Chefs](#)



First Name

Email

Company

City

Address 1

Billing Address

☐ Same as shipping address

Back to cart

Last Name

Phone Number


Zip Code


Country


Select your country

Address 2

Proceed to shipping

**Chicken Tikka Kabab**
150 gm net
50\$

**Chicken Tikka Kabab**
150 gm net
50\$

**Chicken Tikka Kabab**
150 gm net
50\$

Sub-total130\$

ShippingFree

Discount25%

Tax54.76\$

Total432.65\$

Place an order

Signup Page (/signup)

Dynamic Routing: Though the signup page itself might have a static route, it dynamically handles different user states.

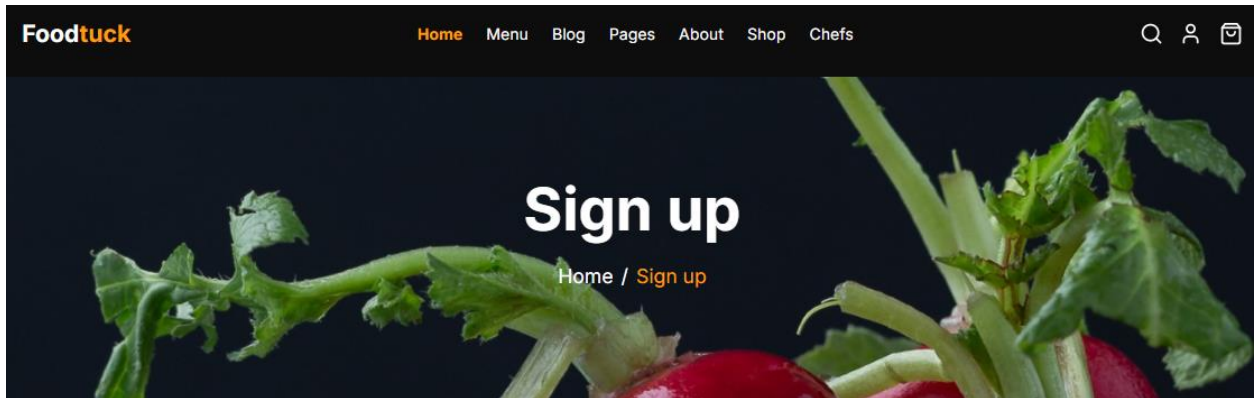
Functionality:

Form Submission: The form collects user details like name, email, password, etc.

Dynamic Feedback: Provides real-time feedback to the user, such as error messages for incorrect inputs or success messages upon successful registration.

Redirection: After successful signup, the user is dynamically redirected to the appropriate page, such as the homepage or a profile setup page.

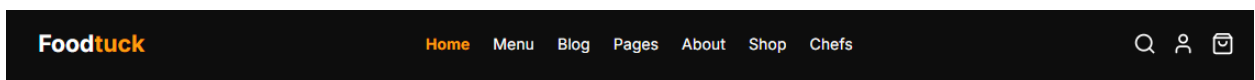
This ensures that the signup process is interactive and adjusts dynamically based on user input and status.



Sign Up

Name

Email



Sign Up

Name

Email


Password


☐ Remember me?

Sign Up

[Forgot password?](#)

or

 Sign up with Google

 Sign up with Apple

