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.

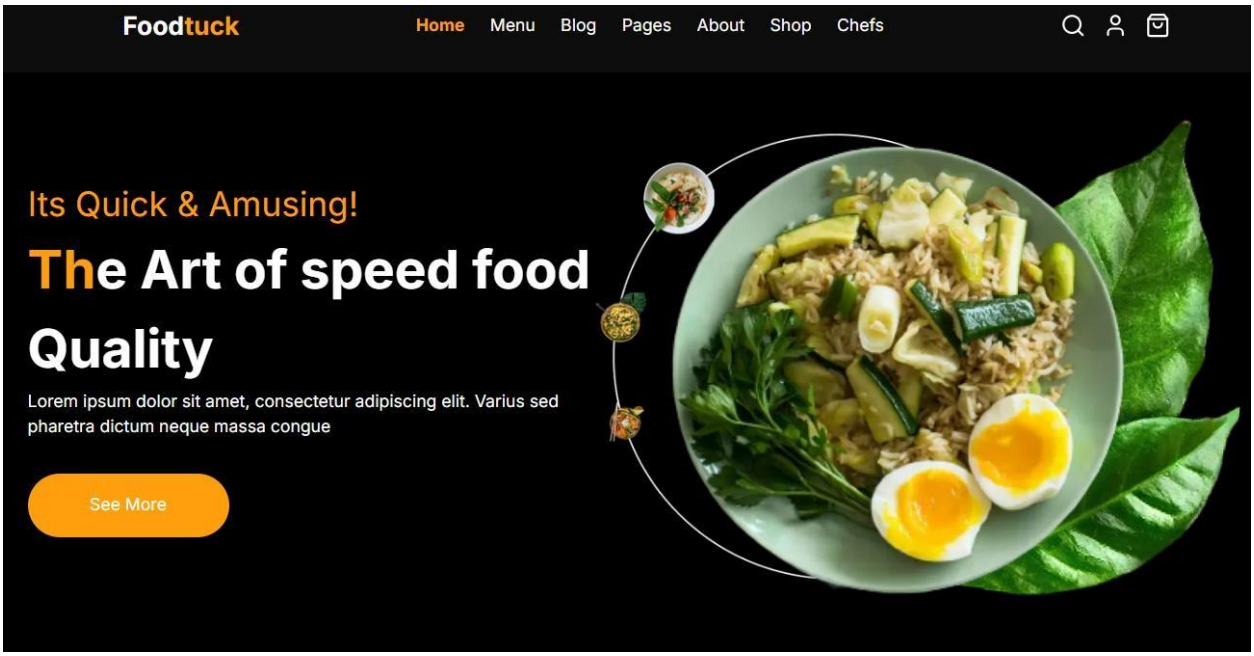
Real-time Order Tracking:

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 (/):

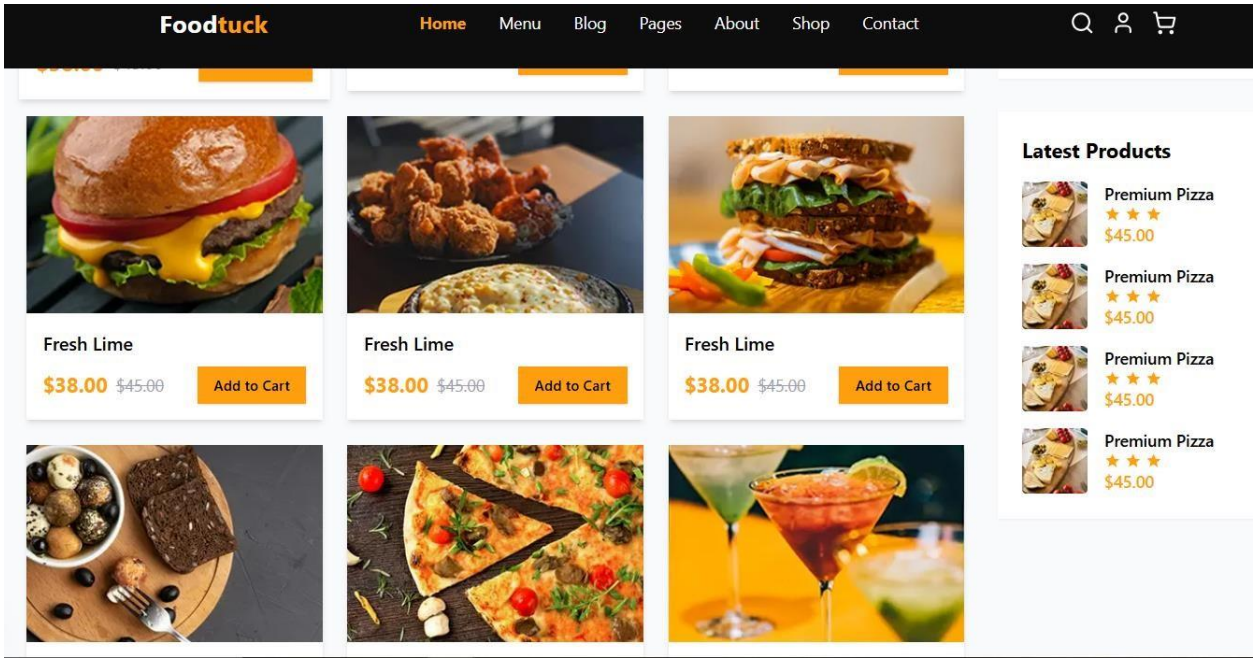
Static route for the homepage showing the general content.



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

```
"use client";
// import { useCart } from "@/app/cart/context/CartContext";
import { client } from "@/sanity/lib/client";
import Image from "next/image";
import React, { useEffect, useState } from "react";
import { Montserrat } from "next/font/google";
import { urlFor } from "@/sanity/lib/image";
import { CiHeart } from "react-icons/ci";
import { FaEye, FaStar } from "react-icons/fa";
import { IoCartOutline } from "react-icons/io5";
// import ProductCardSlide from "../ProductComponent/productCardSlider";

const montserrat = Montserrat({ subsets: ["latin"], weight: ["700"] });

interface IProducts {
  id: string;
  heading: string;
  subheading: string;
  image: {
```

page.tsx ...\ProductsCard U | page.tsx ...\[id] U | productsDetailsComponent.tsx U | 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
```

page.tsx ...\ProductsCard U | page.tsx ...\[id] U | productsDetailsComponent.tsx U | page.tsx ...\app M

src > app > Products > [id] > page.tsx > Iparams

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

Cart Page (/cart):

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

.env.local | fetch.ts | page.tsx ...\Product | page.tsx ...\app

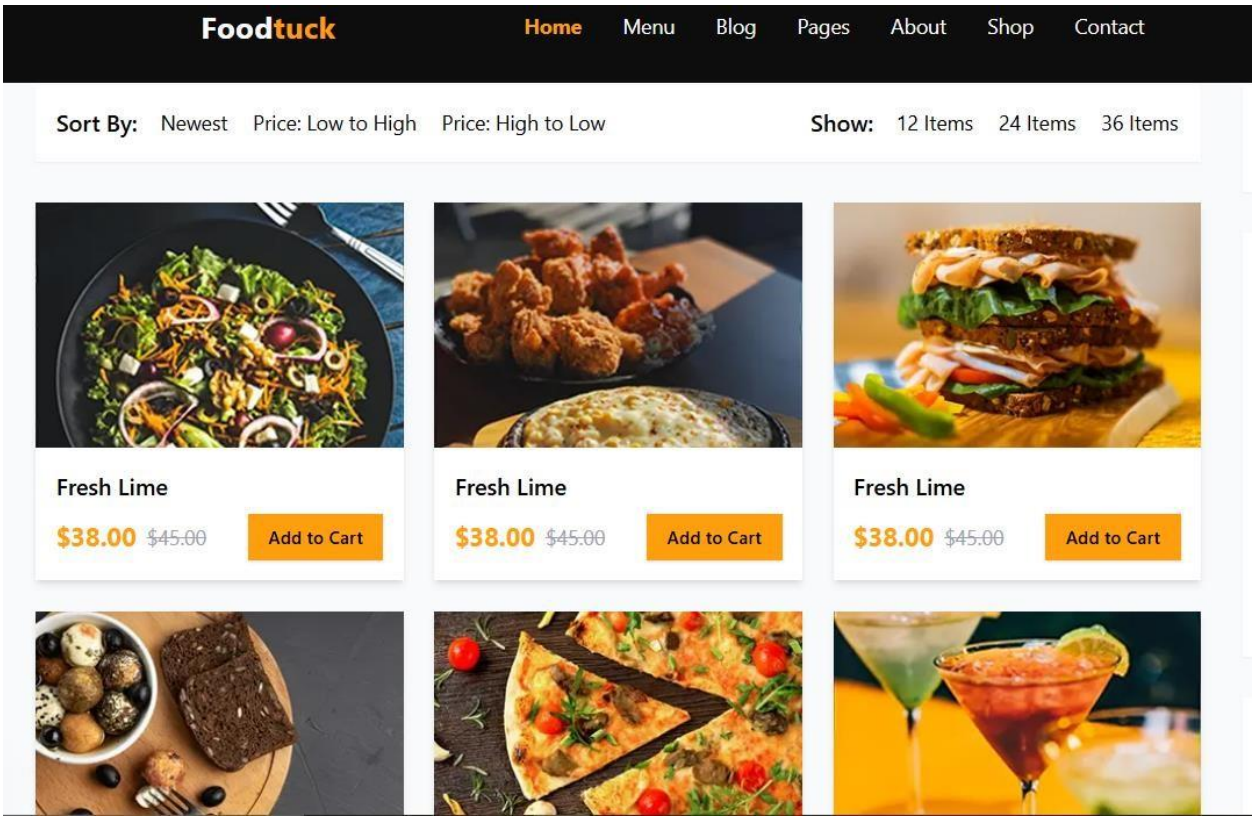
src > app > Product > page.tsx > default

```
"use client";
import React, { useEffect, useState } from "react";
import sanityClient from "@sanity/client";
import Image from "next/image";

const sanity = sanityClient({
  projectId: "ci05×4q0",
  dataset: "production",
  apiVersion: "2023-01-01",
  useCdn: true,
});

interface Product {
  discountPercentage: any;
  _id: string;
  title: string;
  price: number;
  description: string;
  imageUrl: string;
  productImage: {
```

.env.local page.tsx

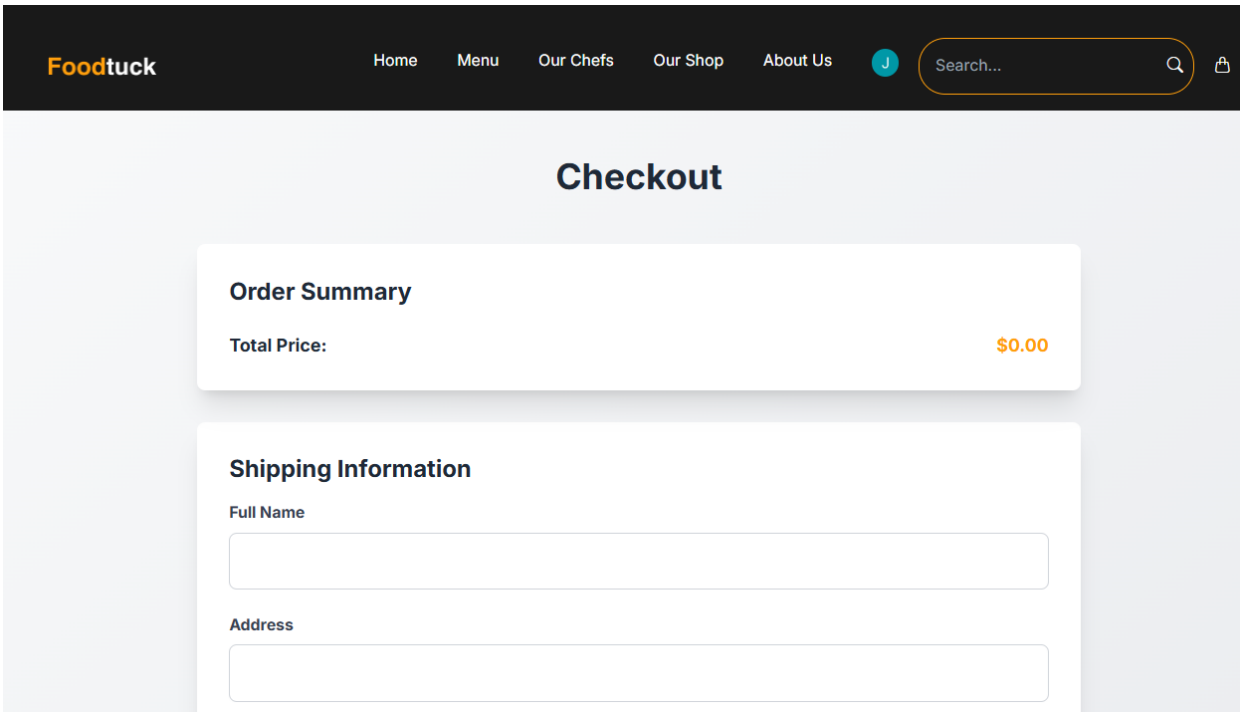
src > app > Product > page.tsx > default

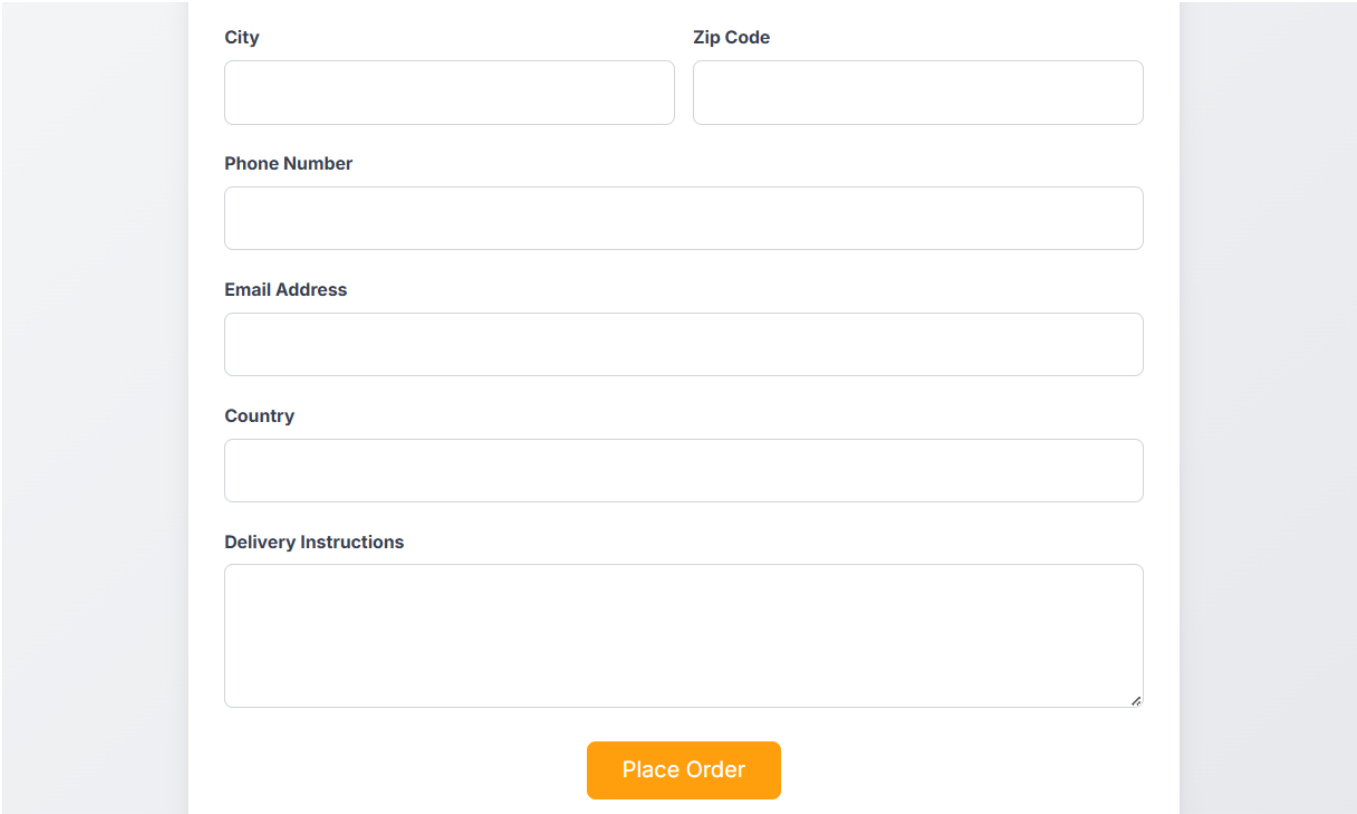
```
26  }
27
28  const ProductCards: React.FC = () ⇒ {
29    const [products, setProducts] = useState<Product[]>([]);
30    const [cart, setCart] = useState<Product[]>([]);
31
32    const fetchProducts = async () ⇒ {
33      try {
34        const query = `*[type == "product"]{
35                _id,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



Checkout Page (/checkout):

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


Foodtuck
Home
Menu
Our Chefs
Our Shop
About Us
J
Search...
Checkout
Order Summary
Total Price:
$0.00
Shipping Information
Full Name
Address


City
Zip Code
Phone Number
Email Address
Country
Delivery Instructions
Place Order

Signup Page (/signup)

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

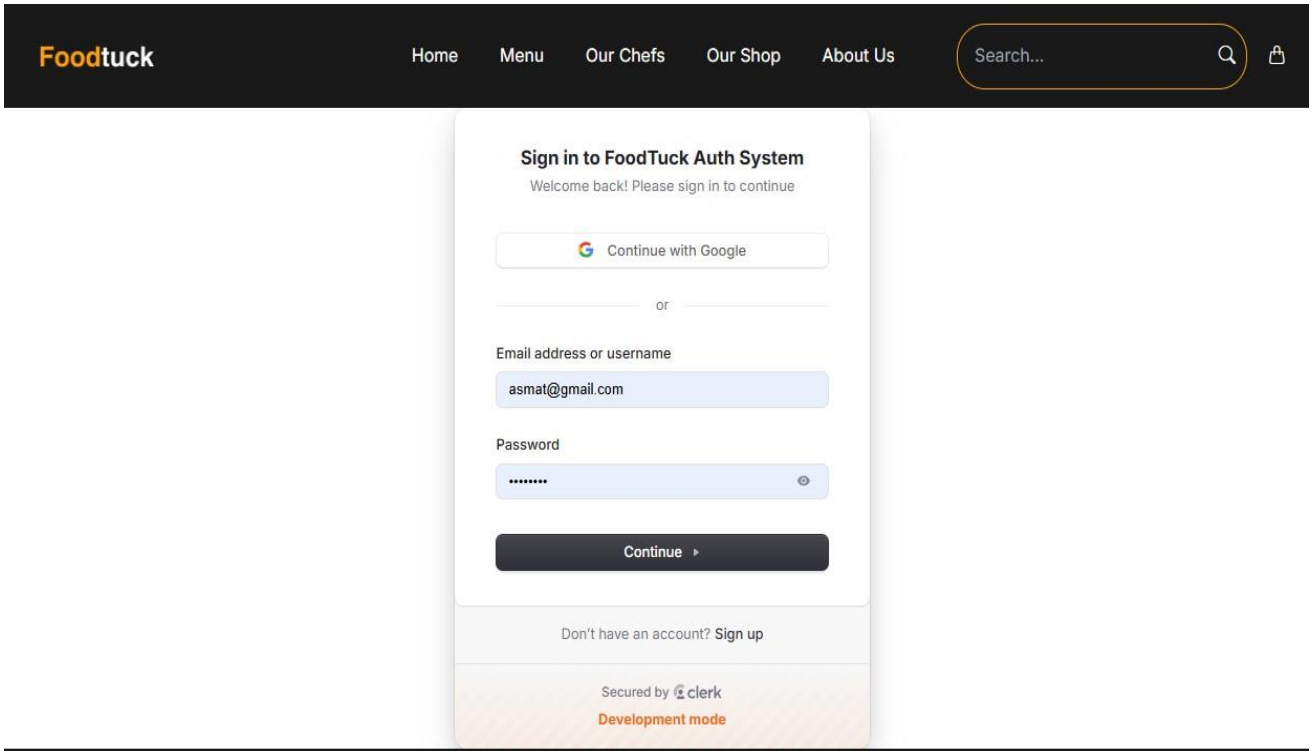
Functionality:

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.


Foodtuck
Home
Menu
Our Chefs
Our Shop
About Us
Search...
Sign in to FoodTuck Auth System
Welcome back! Please sign in to continue
Continue with Google
or
Email address or username
asmat@gmail.com
Password
Continue
Don't have an account? Sign up
Secured by clerk
Development mode