Day 4 - Dynamic Frontend Components for [Foodtuck]

REPORT

Prepared by: SHAHBAZ AHMED

Date: 19-January-2025

Objective:

The objective of Day 4 was to design and develop dynamic frontend components for displaying marketplace data fetched from Sanity CMS or APIs. The focus was on building modular, reusable, and responsive components while adhering to real-world practices to ensure scalability and usability.

Key Learning Outcomes:

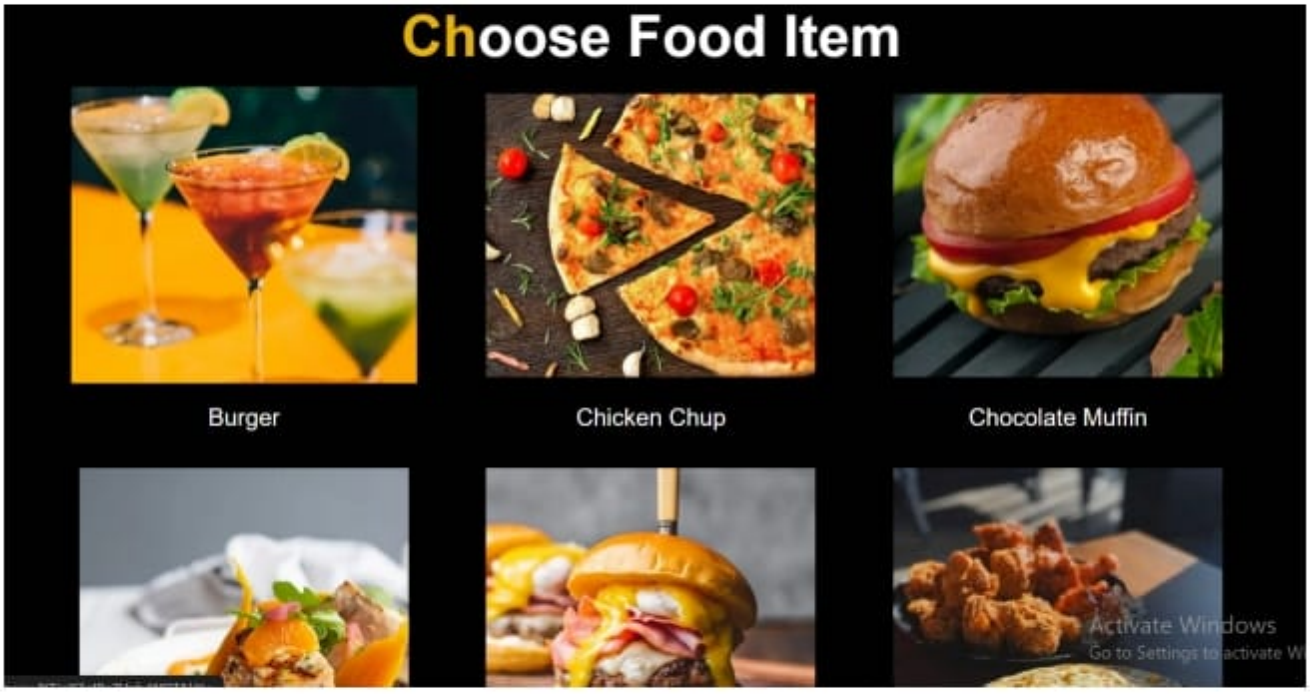
1. Development of dynamic frontend components to display data from Sanity CMS or APIs.
2. Implementation of reusable and modular components.
3. Understanding and application of state management techniques.
4. Importance of responsive design and UX/UI best practices.
5. Experience in replicating professional workflows for real-world projects.

Product Listing Component

Implemented Components:

1. Product Listing Component:

- Rendered product data dynamically in a grid layout.
- Included fields like:
  * Product Name
  * Price
  * Image
  * View Details
  * Add to Cart



Product Detail Component Assignment

Product Detail Component

Created individual product detail pages using dynamic routing in Next.js.

Included detailed fields such as:

- Product Description

- Price

- Image

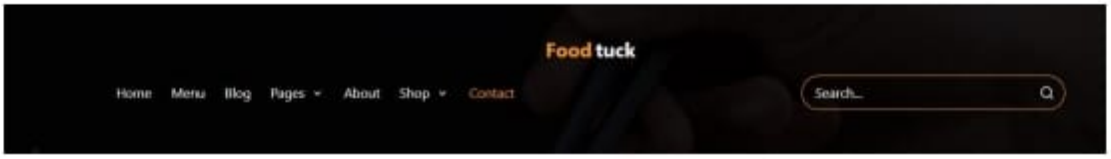
- Rating & Reviews

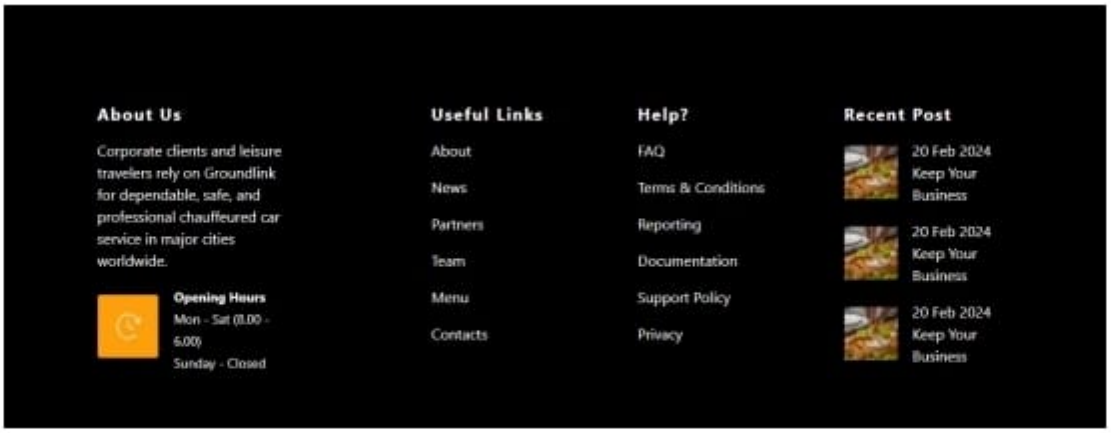
- Share to Social Media

Product Detail Page Example:

Footer and Header Components:

- **Built consistent navigation and branding elements.**
- **Included links to key pages (e.g., Home, About, Contact).**
- **Ensured responsiveness and accessibility.**





Steps Taken to Build and Integrate Components:

1. **Setup:**
    - **Established a connection between the Next.js project and Sanity CMS.**
    - **Fetched and tested data to confirm availability.**
2. **Development:**
    - **Reusable Component Creation:**
        - **Designed modular components such as ProductCard**
        - **Passed data via props to ensure flexibility and reusability.**
    - **State Management:**
        - **Managed global and local state using React's useState and useContext hooks.**
        - **Ensured smooth data flow between components.**
    - **Styling:**
        - **Used Tailwind CSS for responsive designs.**
        - **Focused on maintaining a professional and uniform appearance.**
3. **Testing:**
    - **Conducted thorough testing for responsiveness and functionality across various devices and browsers.**
    - **Debugged and resolved issues related to API integration and dynamic routing.**

Challenges and Solutions:

1. **Challenge: Managing dynamic data rendering from Sanity CMS.**
    - **Solution: Utilized Next.js's getServerSideProps to fetch and pre-render data efficiently.**
2. **Challenge: Implementing state management for cart and checkout functionality.**
    - **Solution: Used React Context API to manage the global state and ensure data consistency.**

3. Challenge: Ensuring responsiveness for all components.
    - Solution: Leveraged Tailwind CSS's utility classes and media queries to create responsive layouts.

Best Practices Followed:

1. Modular and reusable components to ensure scalability.
2. State management with React hooks for efficient data handling.
3. Responsive designs following UX/UI best practices.
4. Performance optimization through lazy loading of images and pagination.

Expected Output:

1. A fully functional product listing page displaying dynamic data from Sanity CMS.
2. Individual product detail pages implemented using dynamic routing.
3. Advanced search functionality for filtering products.
4. A cart component with dynamic state management.
5. A multi-step checkout process with mock payment implementation.
6. Consistent and responsive footer and header components.
7. Notifications and FAQ sections for enhanced user experience.

Code Snippets:

1. **ProductCard Component:**

```tsx
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";
import React from "react";

const Foods = async () => {
  const query = await client.fetch(
    `*[_type == "food"]{
      name,
      originalPrice,
      category,
      available,
      image,
      description,
      price,
      _id
    }`
  );

  console.log(query);

  return (
    <div className="container mx-auto p-4">
      <h1 className="text-2xl font-bold text-center mb-6">Food Menu</h1>
      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6">
        {query.map((food: any) => (
          <div
            key={food._id}
            className="border rounded-lg p-4 shadow-md hover:shadow-lg transition-shadow"
          >
            <Image
              src={urlFor(food.image).url()}
              alt={food.name}
              width={150}
              height={150}
              className="w-full h-auto rounded-md"
            />
            <h2 className="text-lg font-semibold mt-4">{food.name}</h2>
            <p className="text-gray-600 mt-2">${food.price}</p>
            <p className="text-sm text-gray-500 mt-1">{food.category}</p>
          </div>
        ))}
      </div>
    </div>
  );
};

export default Foods;
```

2. **ProductDetail Component :**

```tsx
// src/app/shop/[id]/page.tsx
"use client";
import { useState } from "react";
import { useRouter } from "next/navigation";
import Image from "next/image";
import Navbar from "@/app/components/Navbar";
import Footer from "@/app/components/Footer";
import Stillyouneed from "@/app/components/Stillyouneed";
import Link from "next/link";

const ProductDetailsPage = ({ params }: { params: { id: string } }) => {
  const { id } = params; // Get the product ID from the dynamic route


  const products = [
    {
      id: 1,
      name: "Fresh Lime",
      description:

"A refreshing and zesty lime flavor that awakens your taste buds with its invigor
ating essence, meticulously crafted to achieve the perfect balance of tanginess a
nd natural sweetness. Each sip delivers a delightful burst of citrusy goodness, m
aking it an irresistible treat for every occasion, leaving you craving more."
,
      price: "200 Rupees/-",
      imageUrl: "/pictures/shop1.png",
      additionalInfo:
        "A delightful treat for your taste buds, perfect for any occasion!",
      reviews: 4.5,
      specifications: [
        "Natural lime essence",
        "No artificial additives",
        "Bursting with flavor",
        "Healthy and refreshing",
      ],
```

3. Cart Component:

```tsx
"use client"
import Navbar from "../components/Navbar";
import Footer from "../components/Footer";
import Image from "next/image";
import React, { useState, useEffect } from "react";
import { RxCross1 } from "react-icons/rx";
import Stillyouneed from "../components/Stillyouneed";
import { PiCheckSquareOffset } from "react-icons/pi";
import Link from "next/link";
import { useCart } from "../context/CartContext";

// Define the type for Product
type Product = {
  id: number;
  name: string;
  price: number;
  quantity: number;
  image: string;
};

// Define the type for CartItem
type CartItem = Product & {
  total: number;
};

const ShoppingCart = () => {
  const { cartItems, updateCartItem, removeFromCart } = useCart();
  const [cartSubtotal, setCartSubtotal] = useState(0);
  const [shippingCharge, setShippingCharge] = useState(10);
// Default shipping charge
  const [totalAmount, setTotalAmount] = useState(0);

  useEffect(() => {
    // Calculate subtotal and total amount whenever the cart items change
    const subtotal = cartItems.reduce(
      (acc, item) => acc + item.price * item.quantity,
      0
    );
    const total = subtotal + shippingCharge;

    setCartSubtotal(subtotal);
    setTotalAmount(total);
  }, [cartItems, shippingCharge]);

  const increment = (id: number) => {
    const product = cartItems.find((item) => item.id === id);
    if (product) {
      updateCartItem(id, product.quantity + 1);
    }
  };

  const decrement = (id: number) => {
    const product = cartItems.find((item) => item.id === id);
    if (product && product.quantity > 1) {
      updateCartItem(id, product.quantity - 1);
    }
  };
```

Screenshots and Screen Recordings:

(Include visuals of the implemented components such as the product listing page, product detail pages, the cart component and search bar.)

Conclusion:

By following the outlined steps and best practices, the dynamic frontend components for the marketplace were successfully developed and integrated. The components are modular, reusable, and responsive, providing a professional and scalable solution for real-world projects.

Self-Validation Checklist:

Frontend Component Development	Styling and Responsiveness	Code Quality	Documentation and Submission	Final Review
✓	✓	✓	✓	✓

Professional Practices Emphasized:

- **Modular and reusable component design.**
- **State management and dynamic data binding.**
- **Responsive and user-friendly UI design.**
- **Thorough documentation for code and processes.**