

Functional Deliverables

Product Listing Page with Dynamic Data

localhost:3000/shop

Oreo Frappe \$6-\$10

Pina Colada \$8

Cold Coffee \$9-\$14

Alfredo Fettuccine Pasta \$8-\$11

Cheese Butter \$8

Country Burger \$45-\$50

Search Product

Category

- Drink
- Pasta
- Steak
- Sandwich
- Main Course
- Appetizer
- Burger
- Dessert

Perfect Taste
Classic Restaurant
45.00\$

localhost:3000/shop

Pizza \$43-\$60

Chicken Chup \$12-\$15

Sandwiches \$25-\$30

BBQ Chicken Burger \$8-\$10

Beef Double Patty Burger \$12-\$15

Fresh Lime \$38-\$45

Shop Now

Filter By Price

From \$0 to \$8000 Filter

Latest Products

- Pizza ★★★★☆ \$35
- Cupcake ★★★★☆ \$35
- Cookies ★★★★☆ \$35
- Burger ★★★★☆ \$35

Product Tags

Services Our Menu Pizza

Cupcake Burger Cookies

Our Shop Tandoori Chicken

Individual Product Detail Pages with Accurate Routing and Data Rendering

localhost:3000/shop/3

The screenshot shows a product detail page for a 'Country Burger'. On the left, there's a vertical stack of five smaller food images. The main image on the right shows a classic country-style burger served with fries. A 'In Stock' button is at the top right. Below the image, the product name 'Country Burger' is displayed in a large font. A brief description follows: 'Classic country-style burger served with fries.' The price is listed as '\$45.50'. Below the price, a rating of '5.0 Rating' and '22 Review' is shown. A category 'Dictum/cursus/Risus' and a tag 'Our Shop' are also present. At the bottom, there's a 'Share' button with social media icons and an 'Add to Cart' button.

In Stock

Prev Next

Country Burger

Classic country-style burger served with fries.

45\$ 50\$

★★★★★ | 5.0 Rating | 22 Review

Dictum/cursus/Risus

- 1 + Add to Cart

Add to Wishlist Compare

Category: Pizza

Tag: Our Shop

Share : [Facebook](#) [Twitter](#) [Pinterest](#) [Instagram](#) [Email](#)

localhost:3000/shop/12

The screenshot shows a product detail page for an 'Oreo Frappe'. Similar to the first page, it features a vertical stack of five food images on the left and a main image of the frappe on the right. The product name 'Oreo Frappe' is centered above a description: 'A smooth blend of vanilla ice cream, your favorite oreo cookies, milk and sugar and topped with whipped cream and oreo cookies.' The price is '\$6.40'. Below the price, a rating of '5.0 Rating' and '22 Review' is shown. A category 'Dictum/cursus/Risus' and a tag 'Our Shop' are also present. At the bottom, there's a 'Share' button with social media icons and an 'Add to Cart' button. The developer tools' Network tab is open, showing a successful fetch request for the food data. The console log shows the fetched data, including the food's URL, description, and price.

Dimensions: Responsive 1291 x 773 60% No throttling

Foodtruck

Shop Details

Home > Shop details

Search...

In Stock

Prev Next

Oreo Frappe

A smooth blend of vanilla ice cream, your favorite oreo cookies, milk and sugar and topped with whipped cream and oreo cookies.

6\$ 40\$

★★★★★ | 5.0 Rating | 22 Review

Dictum/cursus/Risus

Share : [Facebook](#) [Twitter](#) [Pinterest](#) [Instagram](#) [Email](#)

Hide network

Preserve log

Selected context only

Group similar messages in console

Show CORS errors in console

Eager evaluation

Autocomplete from history

Treat code evaluation as user action

Fetching data for 12

Food Data Fetched Successfully

Data

Single Food:

Food ID: 12

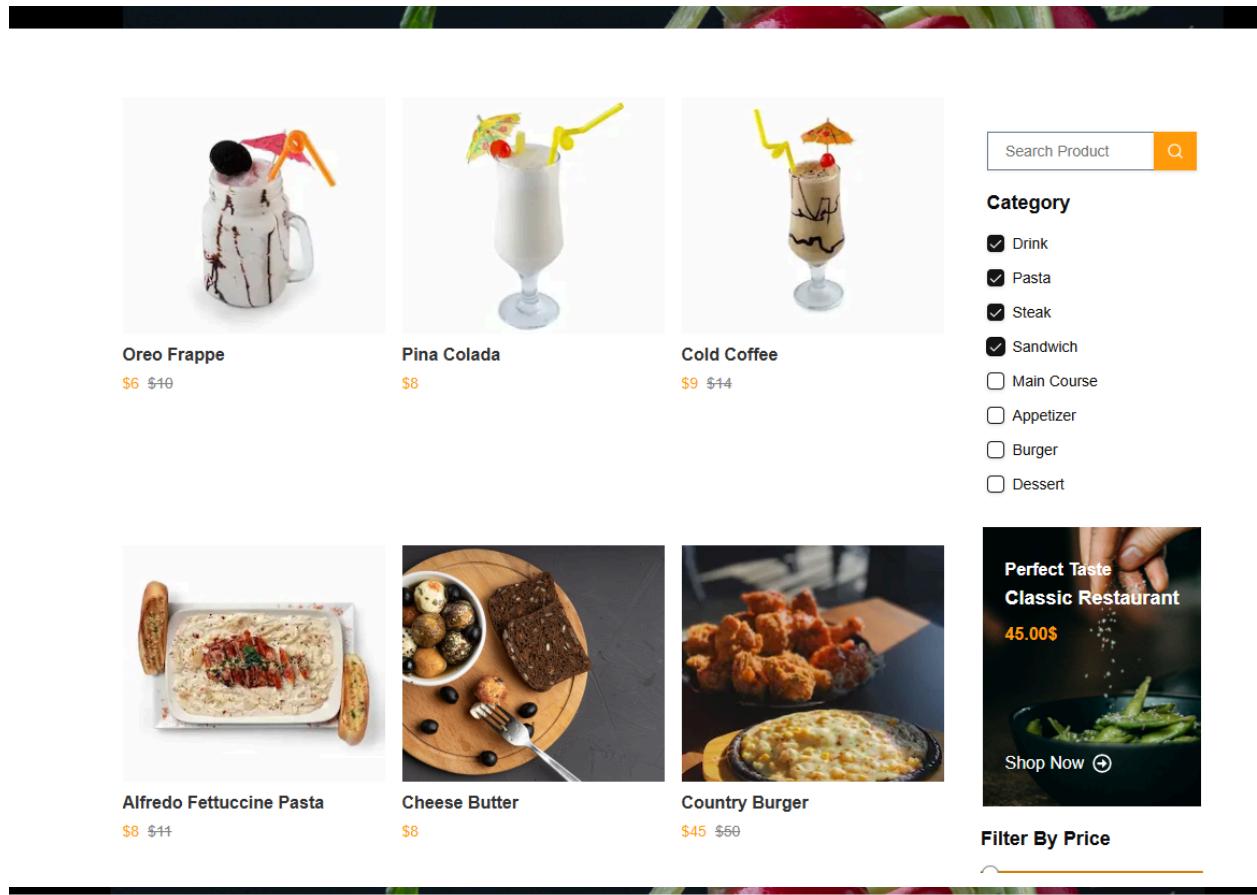
Category Filter, Search Bar, Pagination



Category

- Drink
- Pasta
- Steak
- Sandwich
- Main Course
- Appetizer
- Burger
- Dessert

Filter By Price

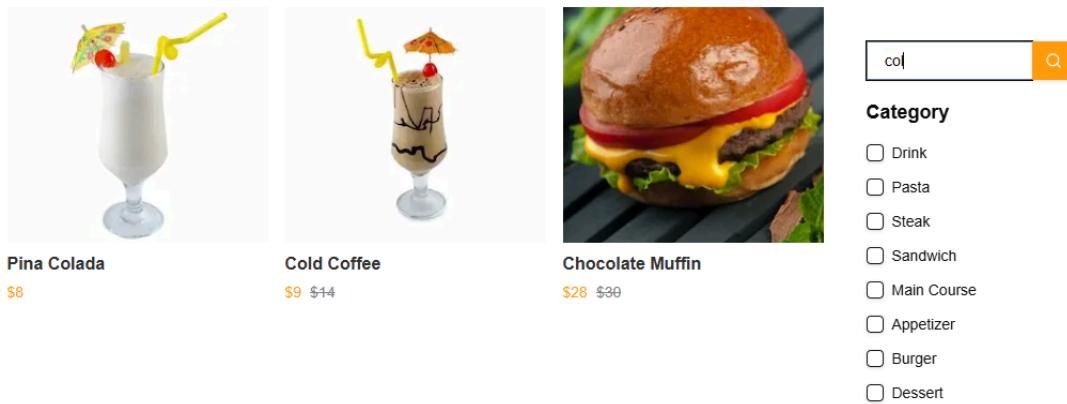


 Oreo Frappe \$6 \$10	 Pina Colada \$8	 Cold Coffee \$9 \$14
 Alfredo Fettuccine Pasta \$8 \$11	 Cheese Butter \$8	 Country Burger \$45 \$50
		 Perfect Taste Classic Restaurant 45.00\$ Shop Now 



Category

- Drink
- Pasta
- Steak
- Sandwich
- Main Course
- Appetizer
- Burger
- Dessert



 Pina Colada \$8	 Cold Coffee \$9 \$14	 Chocolate Muffin \$28 \$30
--	---	--

**Pizza**

\$43 \$60

**Chicken Chup**

\$12 \$15

**Sandwiches**

\$25 \$30

**BBQ Chicken Burger**

\$8 \$10

**Beef Double Patty Burger**

\$12 \$15

**Fresh Lime**

\$38 \$45

Latest Products**Pizza**★★★★★
\$35**Cupchake**★★★★★
\$35**Cookies**★★★★★
\$35**Burger**★★★★★
\$35**Product Tags**[Services](#) [Our Menu](#) [Pizza](#)[Cupcake](#) [Burger](#) [Cookies](#)[Our Shop](#) [Tandoori Chicken](#)[1](#) [2](#) [Next >](#)**Burger**★★★★★
\$35**Product Tags**[Services](#) [Our Menu](#) [Pizza](#)[Cupcake](#) [Burger](#) [Cookies](#)[Our Shop](#) [Tandoori Chicken](#)[◀ Previous](#) [1](#) [2](#) [Next ▶](#)**Similar Products****Oreo Frappe**

\$6 \$10

**Pina Colada**

\$8

**Cold Coffee**

\$9 \$14

**Alfredo Fettuccine Pasta**

\$8 \$11



Code Deliverables

Shop Card

```
src > components > ShopCard.tsx > ShopCard
1 import Image from "next/image";
2
3 interface ShopCardType {
4   id?: number;
5   ImagePath: string;
6   AltText: string;
7   ImageWidth?: number;
8   ImageHeight?: number;
9   CurrentPrice: number;
10  OldPrice?: number;
11  DishName: string;
12 }
13
14 export default function ShopCard({ ImagePath, ImageHeight, ImageWidth, AltText, CurrentPrice, DishName, OldPrice}: ShopCardType) {
15   return(
16     <div className="w-full overflow-hidden text-white bg-transparent hover:scale-105 transition duration-300">
17       /* <div className="relative "> */
18       <Image
19         className="object-cover h-[220px]"
20         src={ImagePath}
21         alt={AltText}
22         width={ImageWidth}
23         height={ImageHeight}
24       />
25       /* </div> */
26       <div className="my-2">
27         <p className=" font-bold text-wrap text-[#333333]">{DishName}</p>
28         <p className=" text-[#FF9F0D] text-sm mt-1">${CurrentPrice}<br/>{OldPrice} && (<span className="line-through text-[#828282] ml-2">${OldPrice}</span>)</p>
29       </div>
30     </div>
31   )
32 }
33
34 }
```

Search Bar

```
src > components > SearchBar.tsx > ...
1 import { ChangeEvent } from "react";
2 import { Button } from "./ui/button";
3 import { Input } from "./ui/input";
4 import { FiSearch } from "react-icons/fi";
5
6
7 interface SearchCardType {
8   PlaceholderText: string;
9   InputBgColor?: string;
10  value: string;
11  onChange: (e: ChangeEvent<HTMLInputElement>) => void;
12 }
13
14 export default function searchBar({PlaceholderText, InputBgColor = "transparent", value, onChange}: SearchCardType){
15   return(
16     <div className="flex mt-8">
17       <Input
18         className="py-4 pl-4 border-slate-500 rounded-none border-r-0"
19         type="text"
20         placeholder={PlaceholderText}
21         style={{backgroundColor: InputBgColor}}
22         value={value}
23         onChange={onChange}
24       />
25       <Button type="button" className="bg-[#FF9F0D] rounded-none px-3 py-[16.7px]>
26         <FiSearch />
27       </Button>
28     </div>
29   )
30 }
31 }
```

API Integration Methods in utils.ts

```
src > lib > ts utils.ts > ...
1 import { client } from "@/sanity/lib/client";
2 import { clsx, type ClassValue } from "clsx"
3 import { twMerge } from "tailwind-merge"
4
5 export function cn(...inputs: ClassValue[]) {
6   return twMerge(clsx(inputs))
7 }
8
9 export async function fetchShopData(query: string) {
10   try {
11     // const query = `*[_type == "food"]{id, "imageUrl":image.asset->url, name, price, originalPrice}`;
12     const food = await client.fetch(query);
13     console.log("Food Data Fetched Successfully");
14     if (!food.length) throw new Error('No Food Items Found!');
15     return food;
16   } catch (error) {
17     console.error('Error fetching data:', error);
18     throw error;
19   }
20 }
21
22 export async function fetchChefData(query: string) {
23   try {
24     // const query = `*[_type == "chef"]{"imageUrl": image.asset->url, name, position,}`;
25     const chef = await client.fetch(query);
26     console.log("Chef Data Fetched Successfully");
27     if (!chef.length) throw new Error('No Chef Found!');
28     return chef;
29   } catch (error) {
30     console.error('Error fetching data:', error);
31     throw error;
32   }
33 }
34 }
```

Dynamic Routing

```
src > app > shop > [food_id] > page.tsx > ...
22 import { useState } from "react";
23 import { useParams } from "next/navigation";
24 import { fetchShopData } from "@/lib/utils";
25
26 interface Food {
27   id: number;
28   name: string;
29   imageUrl: string;
30   description: string;
31   price: number;
32   originalPrice?: number;
33   available: boolean;
34 }
35
36 export default function ShopDetails() {
37   const [foodData, setFoodData] = useState<Food | null>(null);
38   const [allFood, setAllFood] = useState<Food[]>([[]]);
39   const [error, setError] = useState<string>('');
40   const [loading, setLoading] = useState<boolean>(true);
41   const [activeImage, setActiveImage] = useState<string>("/assets/images/shop/shop_detail/Food_1.svg");
42   const [currentIndex, setCurrentIndex] = useState<number>(0);
43   const itemsPerPage: number = 4;
44
45   const thumbnails = [
46     "/assets/images/shop/shop_detail/Food_1.svg",
47     "/assets/images/shop/shop_detail/Food_2.svg",
48     "/assets/images/shop/shop_detail/Food_3.svg",
49     "/assets/images/shop/shop_detail/Food_4.svg",
50     "/assets/images/shop/shop_detail/Food_5.svg",
51   ];
52
53   const {food_id} = useParams();
54
55   console.log("Food ID: ", food_id);
56 }
```

Technical Report

Steps Taken to Build and Integrate Components

Component Development

Requirement Analysis: Defined the purpose and functionality of each component.

Component Design: Structured the components using modular and reusable patterns.

Implementation: Developed components using Nextjs and TailwindCSS.

Styling and Responsiveness: Applied TailwindCSS to ensure a responsive UI.

Testing: Conducted unit testing and integration testing to validate component behavior.

Integration Process

State Management: Utilized appropriate state management techniques for seamless data flow.

API Integration: Fetched and displayed data from APIs using best practices for error handling.

Dynamic Routing: Implemented dynamic routing using useParams() to retrieve route parameters.

Event Handling: Ensured event-driven interactions were handled efficiently.

Challenges Faced and Solutions Implemented

API Integration Errors

Issue: The fetched data id type did not match the id type from parameters, causing the data to be undefined in the console. The id from parameters was a string, while the expected id in the API query was a number.

Solution: Converted the parameter id from a string to a number using Number(food_id) before making the API request to ensure type consistency.

Handling Dynamic Routes

Issue: Difficulty accessing and managing dynamic route parameters.

Solution: Used useParams() from React Router to extract route parameters dynamically.

Best Practices Followed During Development

Code Quality and Maintainability

- Followed DRY (Don't Repeat Yourself) principle.
- Used meaningful variable and function names.
- Documented code with comments for clarity.

Testing and Debugging

- Wrote unit tests for key components.
- Used debugging tools to identify and fix issues efficiently.

Version Control and Collaboration

- Used Git for version control with clear commit messages.
-