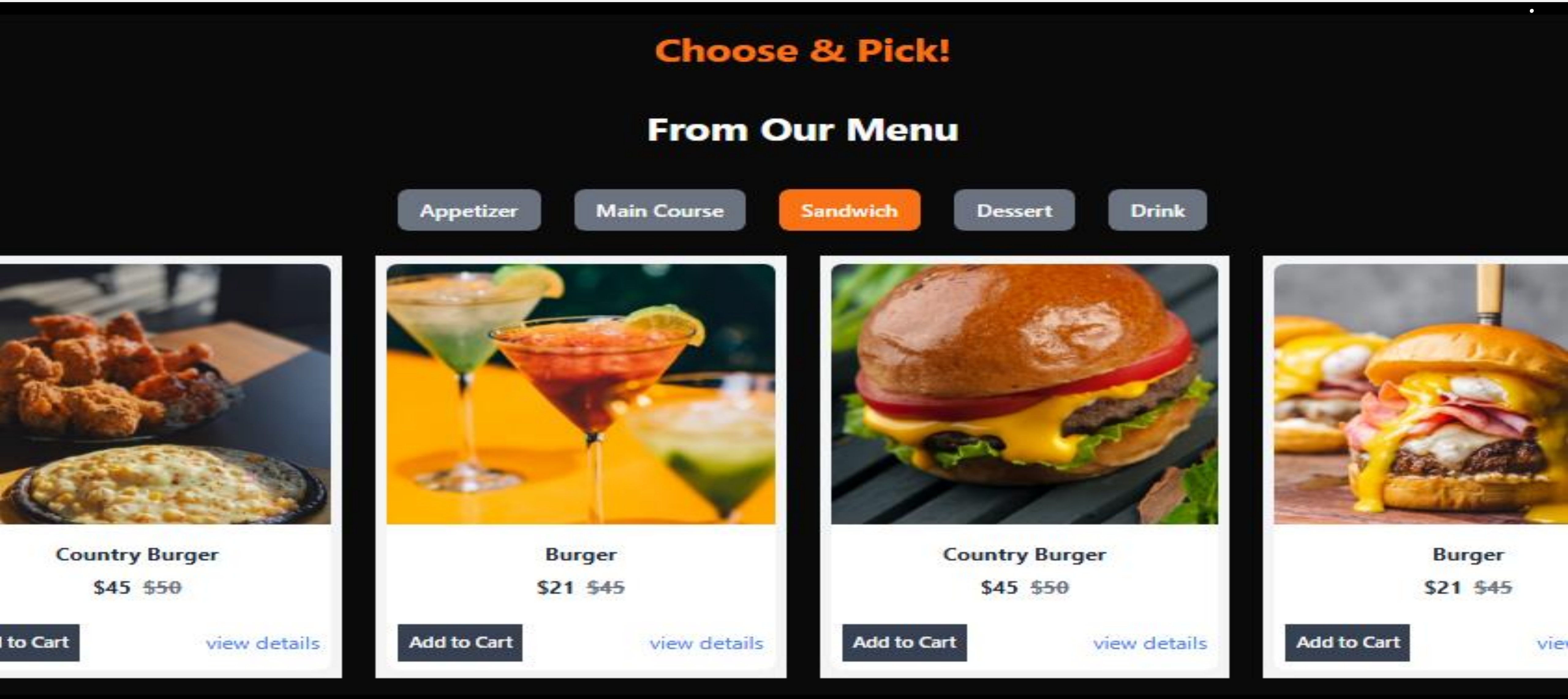
HACKATHON (DAY4)

Dynamic Frontend Components

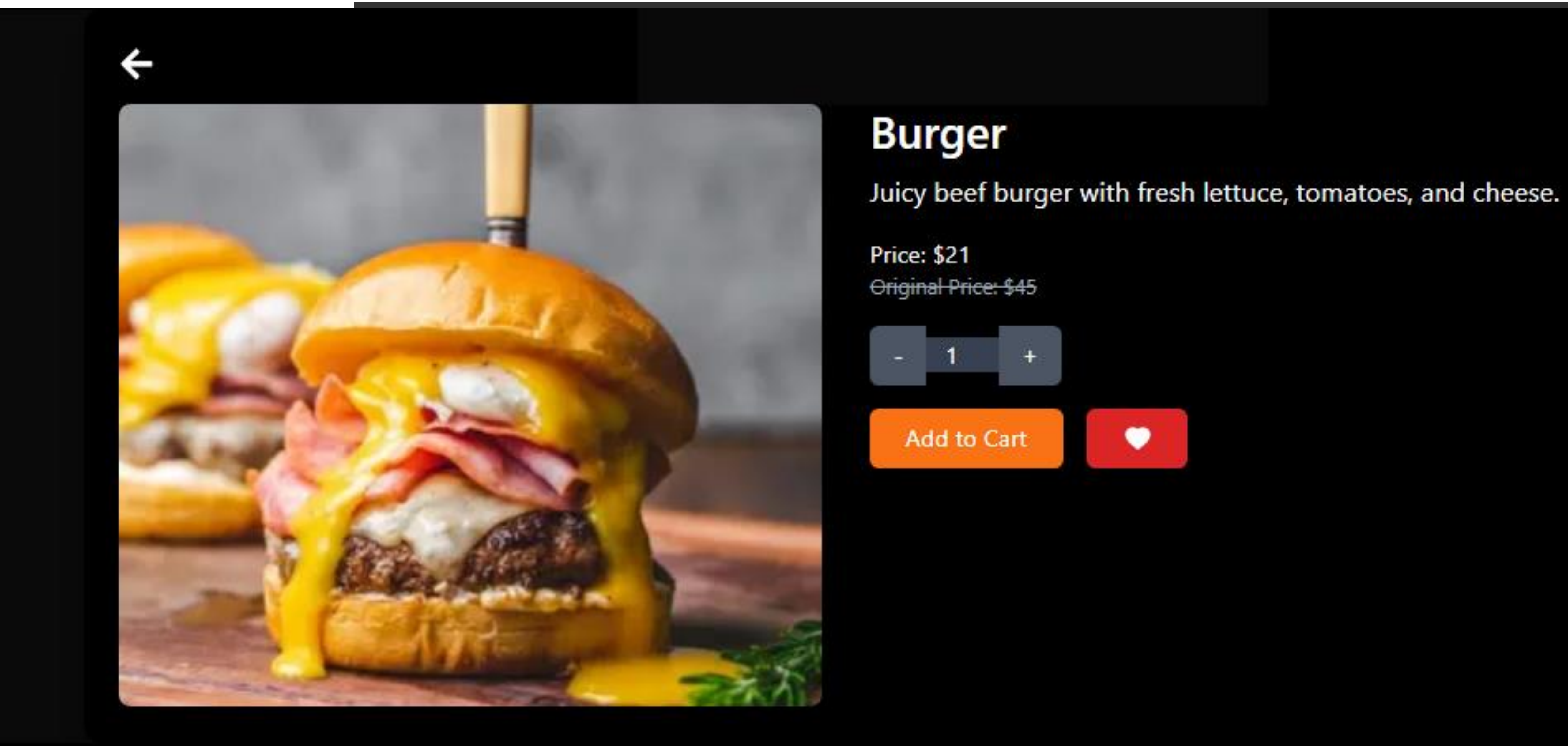
1. Functional Deliverables:

TREY research

Product Listing page with dynamic data .



Individual product detail pages with accurate routing and data rendering.



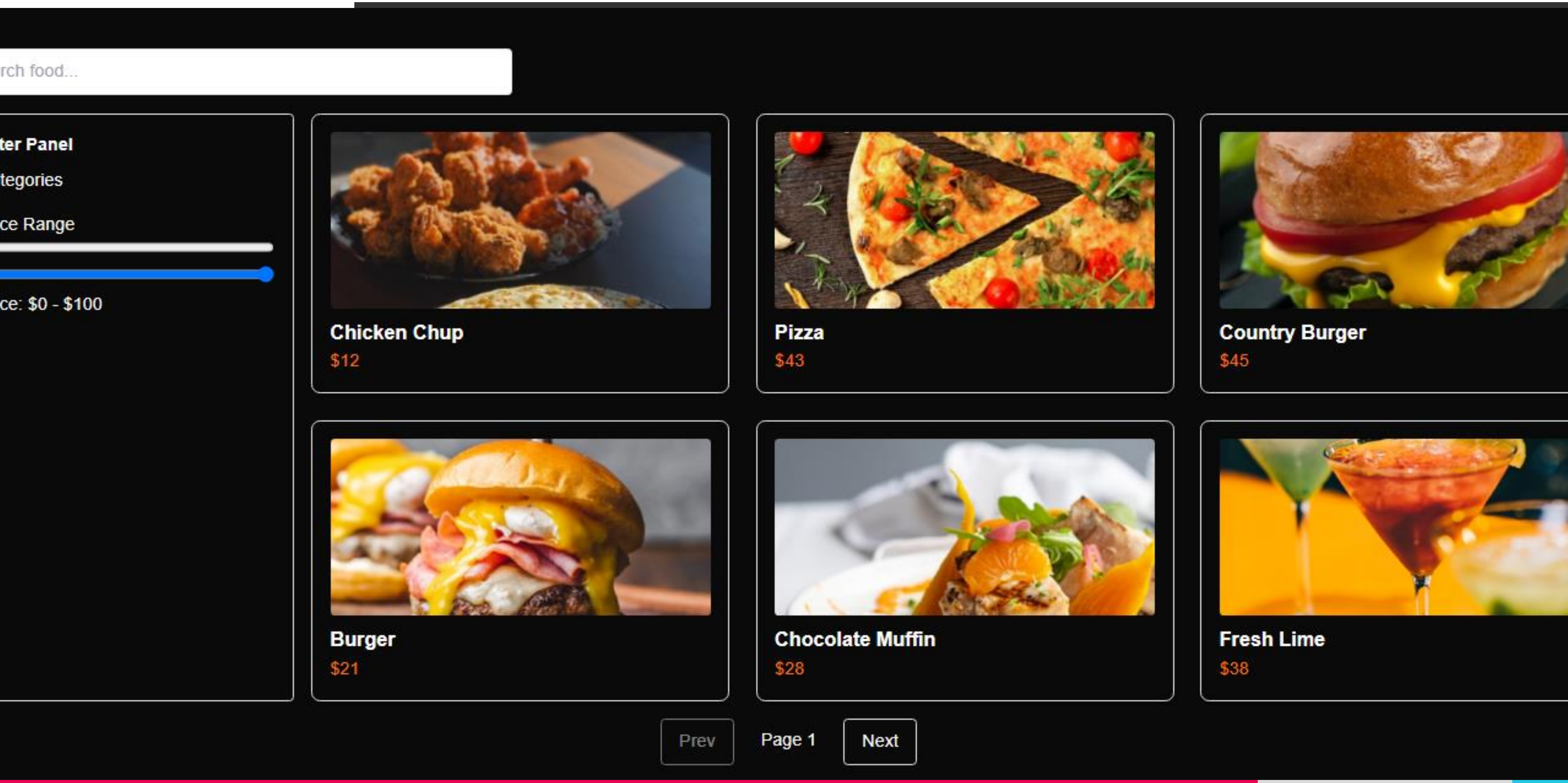
Burger

Juicy beef burger with fresh lettuce, tomatoes, and cheese.

Price: $21

~~Original Price: $45~~

Working category filters, search bar, and pagination.



rch food...

ter Panel

tegories

ce Range

ce: $0 - $100

Chicken Chup
$12

Pizza
$43

Country Burger
$45

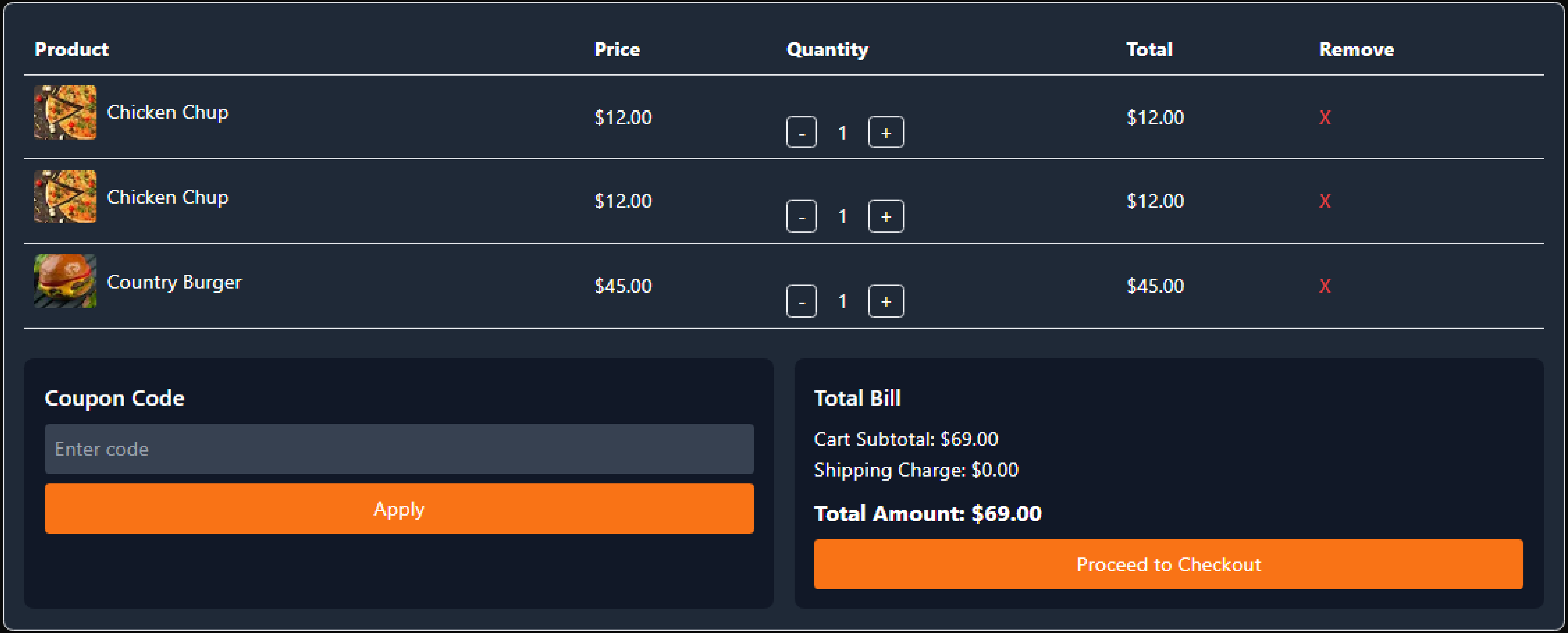
Burger
$21

Chocolate Muffin
$28

Fresh Lime
$38

Prev Page 1 Next

Shooping Cart with food , food details , quantity , discount & Total.



Product	Price	Quantity	Total	Remove
Chicken Chup	$12.00	- 1 +	$12.00	X
Chicken Chup	$12.00	- 1 +	$12.00	X
Country Burger	$45.00	- 1 +	$45.00	X

Coupon Code

Enter code

Apply

Total Bill

Cart Subtotal: $69.00

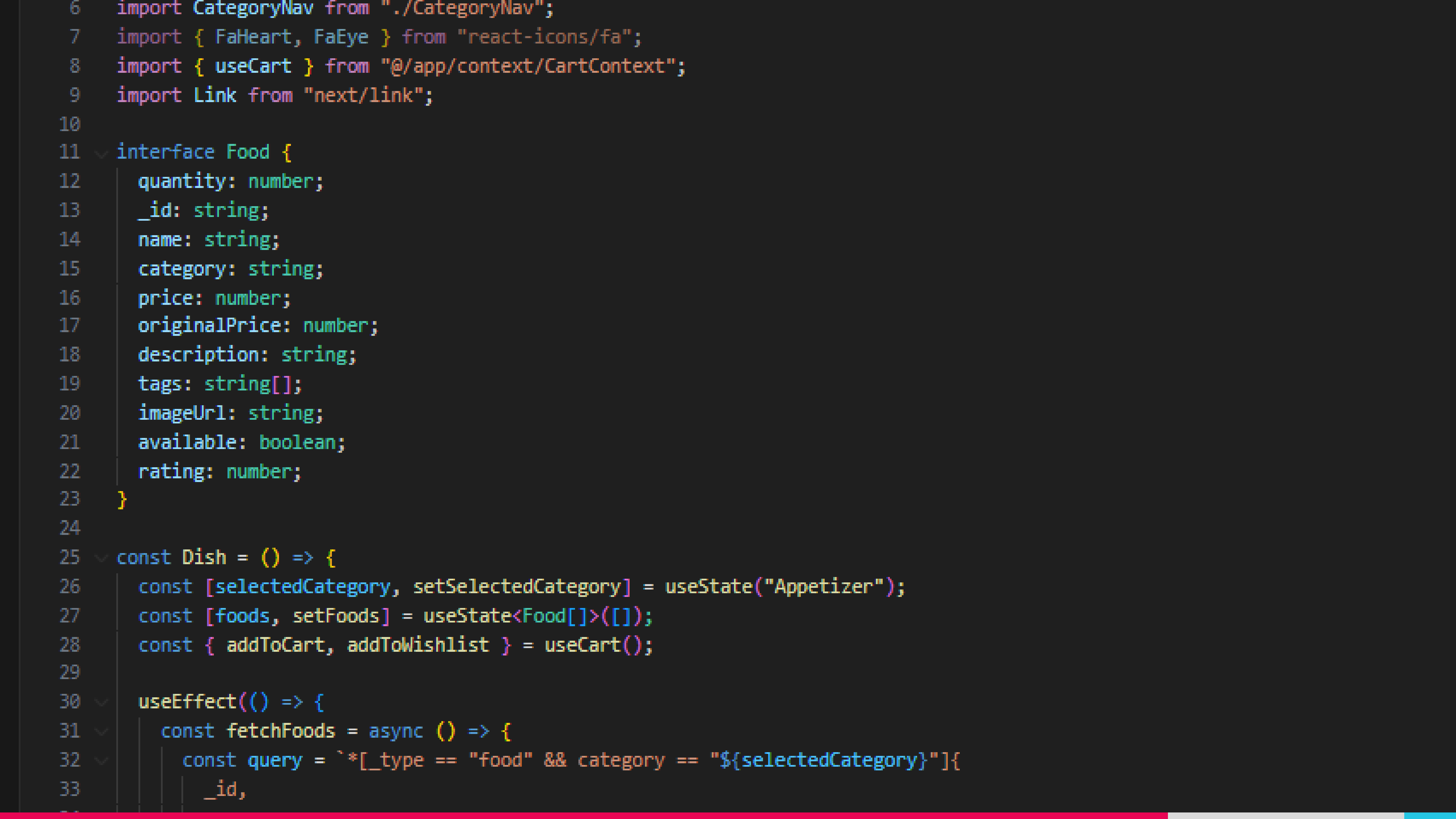
Shipping Charge: $0.00

Total Amount: $69.00

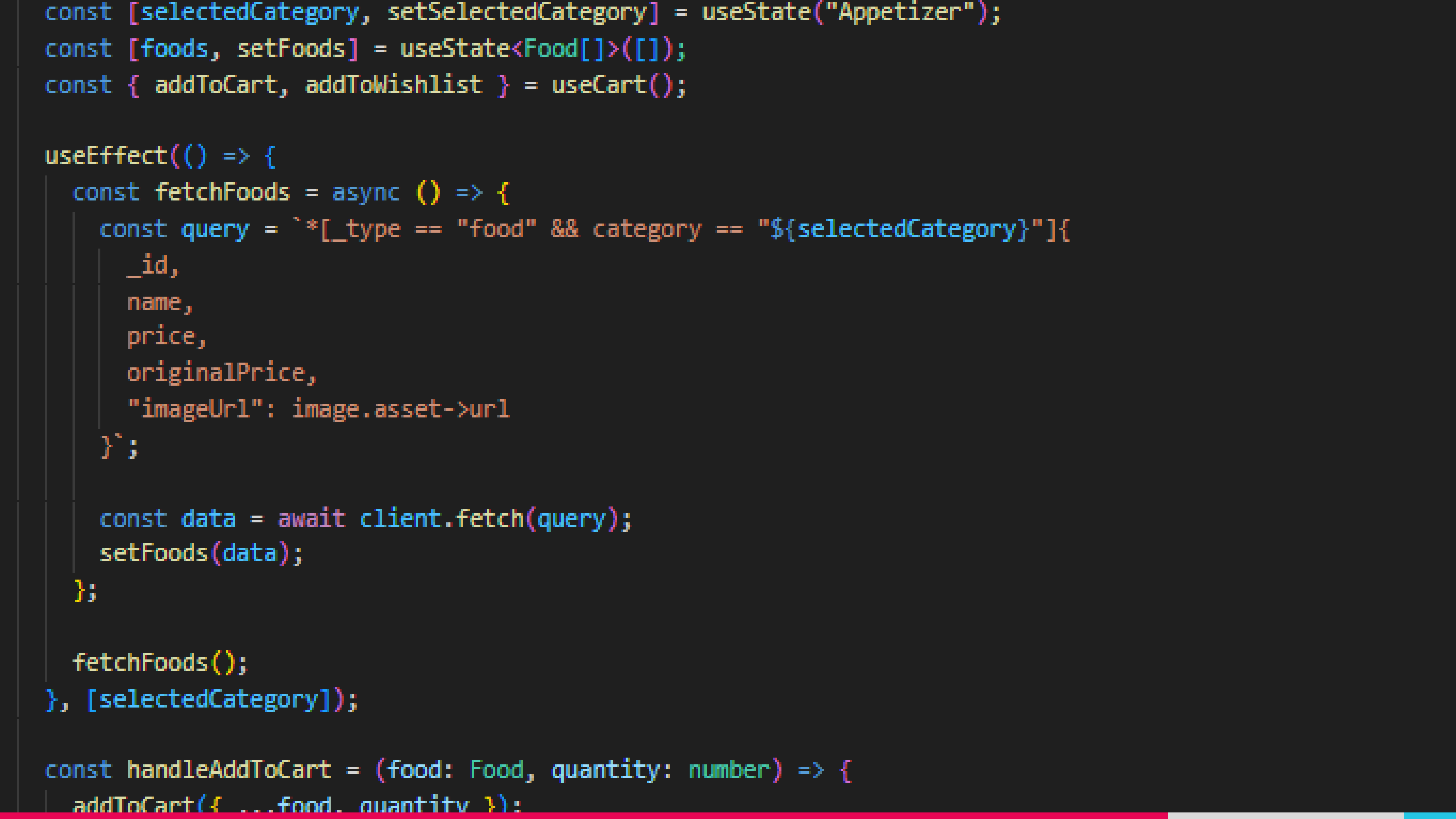
Proceed to Checkout

2. Code Deliverables:

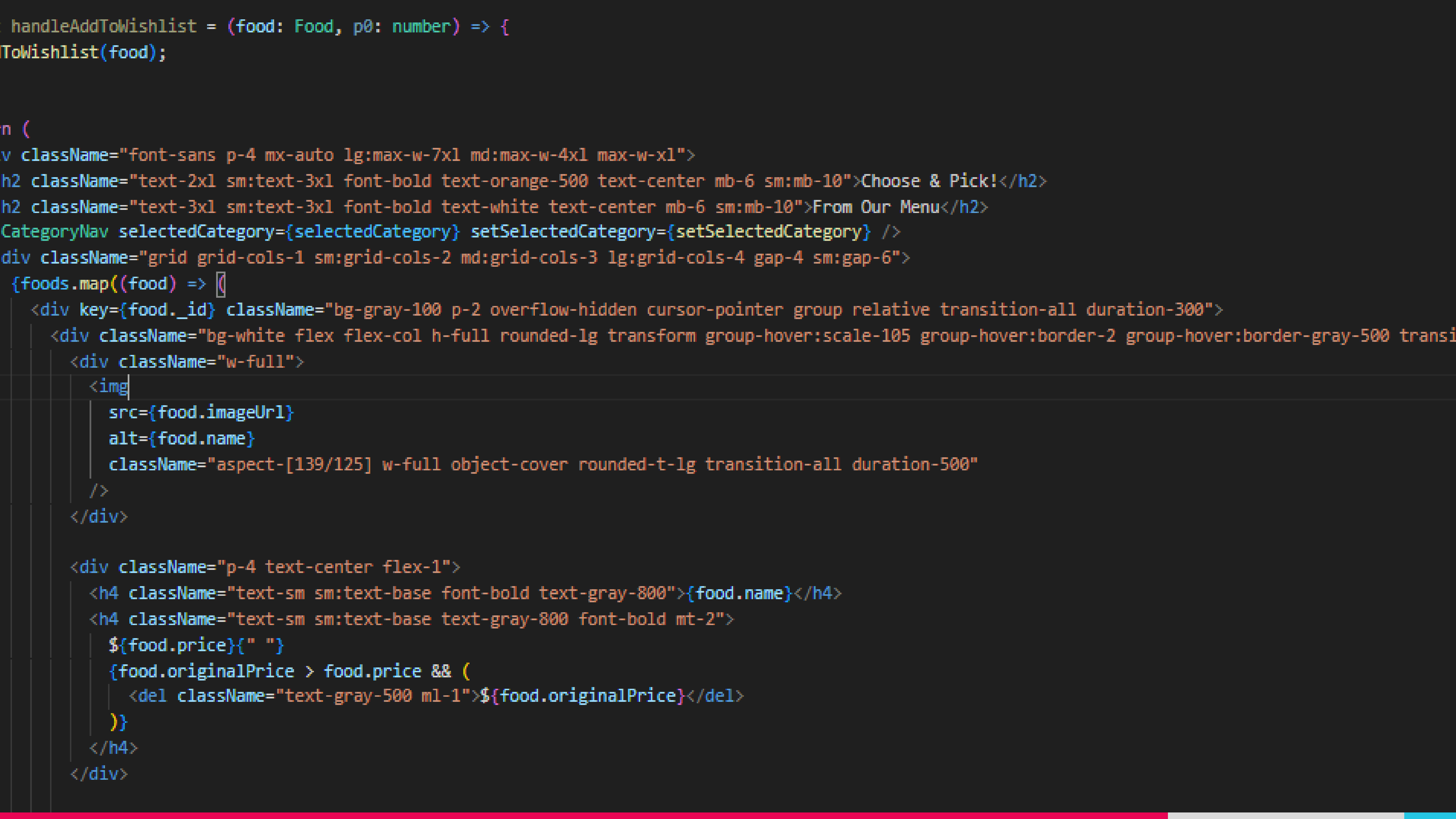
TREY
research



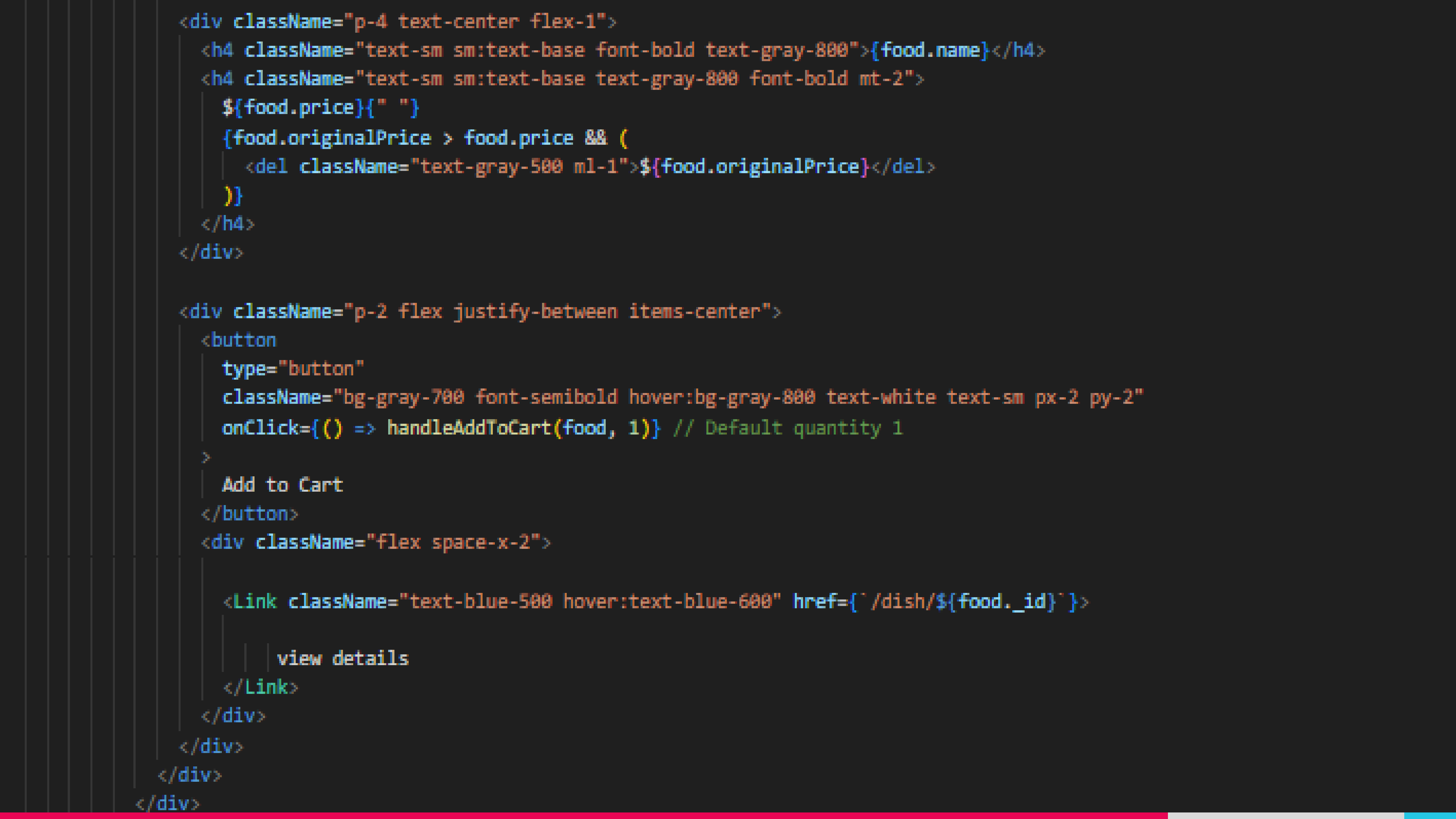
```
import CategoryNav from "./CategoryNav";
import { FaHeart, FaEye } from "react-icons/fa";
import { useCart } from "@/app/context/CartContext";
import Link from "next/link";

interface Food {
  quantity: number;
  _id: string;
  name: string;
  category: string;
  price: number;
  originalPrice: number;
  description: string;
  tags: string[];
  imageUrl: string;
  available: boolean;
  rating: number;
}

const Dish = () => {
  const [selectedCategory, setSelectedCategory] = useState("Appetizer");
  const [foods, setFoods] = useState<Food[]>([]);
  const { addToCart, addToWishlist } = useCart();

  useEffect(() => {
    const fetchFoods = async () => {
      const query = `*[_type == "food" && category == "${selectedCategory}"]{
        _id,
```



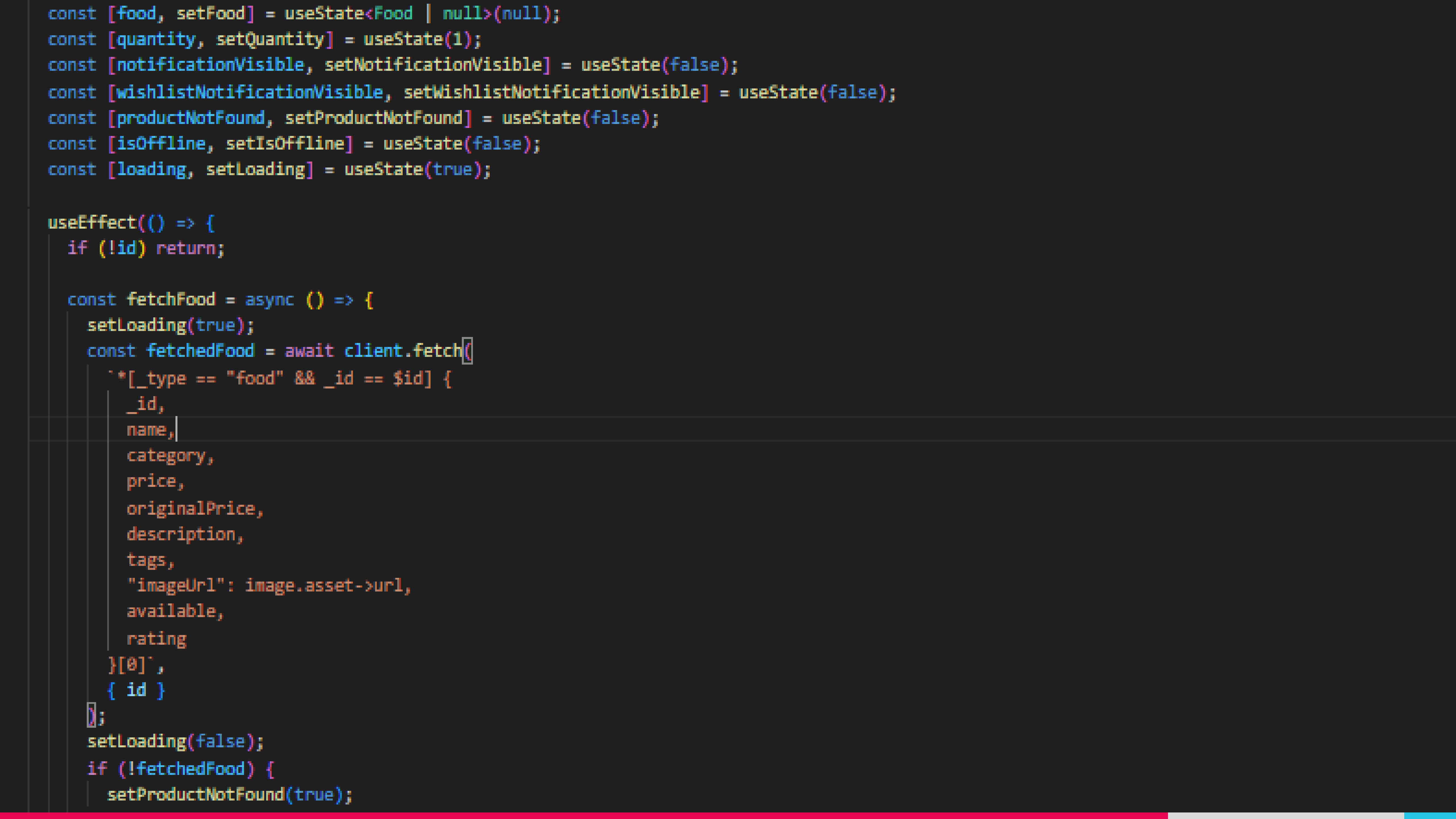
```
const [selectedCategory, setSelectedCategory] = useState("Appetizer");
const [foods, setFoods] = useState<Food[]>([]);
const { addToCart, addToWishlist } = useCart();

useEffect(() => {
  const fetchFoods = async () => {
    const query = `*[_type == "food" && category == "${selectedCategory}"]{
      _id,
      name,
      price,
      originalPrice,
      "imageUrl": image.asset->url
    }`;

    const data = await client.fetch(query);
    setFoods(data);
  };

  fetchFoods();
}, [selectedCategory]);

const handleAddToCart = (food: Food, quantity: number) => {
  addToCart({ ...food, quantity });
```



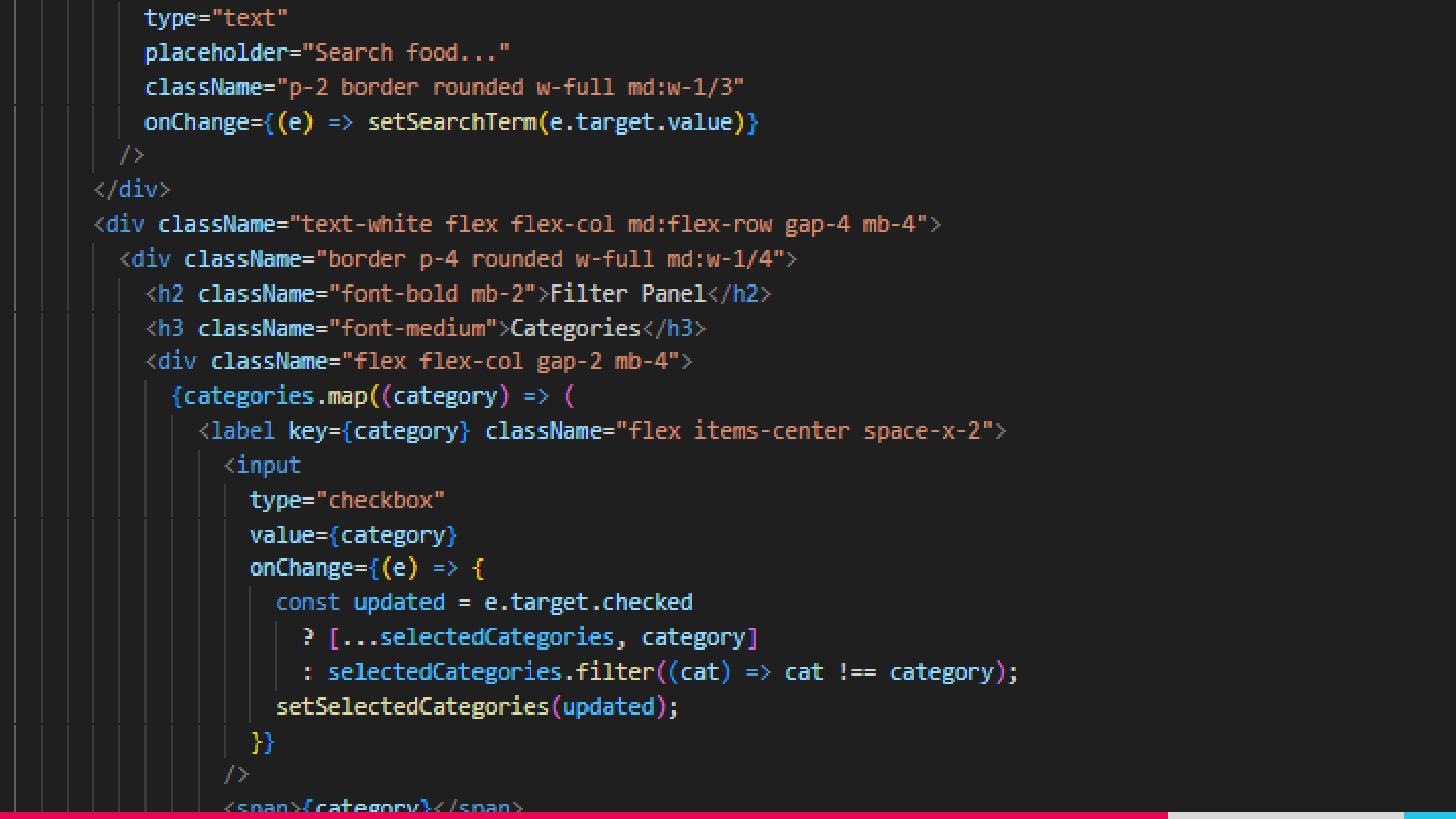
```
: handleAddToWishlist = (food: Food, p0: number) => {
ToWishlist(food);


n (
v className="font-sans p-4 mx-auto lg:max-w-7xl md:max-w-4xl max-w-xl">
h2 className="text-2xl sm:text-3xl font-bold text-orange-500 text-center mb-6 sm:mb-10">Choose & Pick!</h2>
h2 className="text-3xl sm:text-3xl font-bold text-white text-center mb-6 sm:mb-10">From Our Menu</h2>
CategoryNav selectedCategory={selectedCategory} setSelectedCategory={setSelectedCategory} />
div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4 sm:gap-6">
 {foods.map((food) => (
   <div key={food._id} className="bg-gray-100 p-2 overflow-hidden cursor-pointer group relative transition-all duration-300">
     <div className="bg-white flex flex-col h-full rounded-lg transform group-hover:scale-105 group-hover:border-2 group-hover:border-gray-500 transi
       <div className="w-full">
         <img
           src={food.imageUrl}
           alt={food.name}
           className="aspect-[139/125] w-full object-cover rounded-t-lg transition-all duration-500"
         />
       </div>

       <div className="p-4 text-center flex-1">
         <h4 className="text-sm sm:text-base font-bold text-gray-800">{food.name}</h4>
         <h4 className="text-sm sm:text-base text-gray-800 font-bold mt-2">
           ${food.price}{" "}
           {food.originalPrice > food.price && (
             <del className="text-gray-500 ml-1">${food.originalPrice}</del>
           )}
         </h4>
       </div>
```



```
                <div className="p-4 text-center flex-1">
                  <h4 className="text-sm sm:text-base font-bold text-gray-800">{food.name}</h4>
                  <h4 className="text-sm sm:text-base text-gray-800 font-bold mt-2">
                    ${food.price}{" "}
                    {food.originalPrice > food.price && (
                      <del className="text-gray-500 ml-1">${food.originalPrice}</del>
                    )}
                  </h4>
                </div>

                <div className="p-2 flex justify-between items-center">
                  <button
                    type="button"
                    className="bg-gray-700 font-semibold hover:bg-gray-800 text-white text-sm px-2 py-2"
                    onClick={() => handleAddToCart(food, 1)} // Default quantity 1
                  >
                    Add to Cart
                  </button>
                  <div className="flex space-x-2">

                    <Link className="text-blue-500 hover:text-blue-600" href={`/dish/${food._id}`}>

                          view details
                    </Link>
                  </div>
                </div>
              </div>
            </div>
```



```
const [food, setFood] = useState<Food | null>(null);
const [quantity, setQuantity] = useState(1);
const [notificationVisible, setNotificationVisible] = useState(false);
const [wishlistNotificationVisible, setWishlistNotificationVisible] = useState(false);
const [productNotFound, setProductNotFound] = useState(false);
const [isOffline, setIsOffline] = useState(false);
const [loading, setLoading] = useState(true);

useEffect(() => {
  if (!id) return;

  const fetchFood = async () => {
    setLoading(true);
    const fetchedFood = await client.fetch(
      `*[_type == "food" && _id == $id] {
        _id,
        name,
        category,
        price,
        originalPrice,
        description,
        tags,
        "imageUrl": image.asset->url,
        available,
        rating
      }[0]`,
      { id }
    );
    setLoading(false);
    if (!fetchedFood) {
      setProductNotFound(true);
```



```
    type="text"
    placeholder="Search food..."
    className="p-2 border rounded w-full md:w-1/3"
    onChange={(e) => setSearchTerm(e.target.value)}
  />
</div>
<div className="text-white flex flex-col md:flex-row gap-4 mb-4">
  <div className="border p-4 rounded w-full md:w-1/4">
    <h2 className="font-bold mb-2">Filter Panel</h2>
    <h3 className="font-medium">Categories</h3>
    <div className="flex flex-col gap-2 mb-4">
      {categories.map((category) => (
        <label key={category} className="flex items-center space-x-2">
          <input
            type="checkbox"
            value={category}
            onChange={(e) => {
              const updated = e.target.checked
                ? [...selectedCategories, category]
                : selectedCategories.filter((cat) => cat !== category);
              setSelectedCategories(updated);
            }}
          />
          <span>{category}</span>
```

3. Report: “steps taken”.

All Steps Taken & Coding In Best Practice.

Below are the all necessary Steps.

Steps Taken..

- Create components like food card , food details , Search bar , filter panel , pagination & Category etc.
  - By using API display data in dynamically from CMS.
  - Each food have their own id & by clicking on specific food there are details of each food.
  - Implement responsiveness in each components (for mobile, tab and computer).

Clear Code by using best practice..

- Cleared code for readability & reusability.
  - Folder structure in best way to understand easily.
  - Implement lazy loading.
  - Added notification for Add to cart & Add to Wish list .
  - All data update dynamically in cart , wish list when added.

Conclusion:---

By integrating all best practices, clear code , The Website gets a well organized & awesome Frontend. A dynamic Components have all data comes from Sanity CMS (Content Management System) Backend. And also a well organized folder and a security which makes a website great.

THANK YOU!

TREY research