**Prepared by :** **Narmeen Zubair**

**Hackathon 3**

Marketplace Builder

**Q-commerce (Food Tuck)**

# DAY 04: BUILDING DYNAMIC FRONTEND COMPONENTS FOR YOUR MARKETPLACE

## Objective:

The primary objective of **Day 4** is to provide a comprehensive guide for designing **dynamic frontend components** that can display marketplace data fetched from Sanity CMS, enabling developers to create reusable , modular, and scalable UI components that can be easily integrated into our platform.

## Goals:

The Goal of **Day 4** is to build a flexible and **customizable frontend framework** that allows for easy integration of new features and functionalities.

**Task Overview :**

## Key Components We Built :

1.Product Listing Component :

- Rendered product Data dynamically in a grid layout which includes fields like:
  - Product Name
  - Product Image
  - Price
  - Category

**Prepared by :** **Narmeen Zubair**

## Key Features :

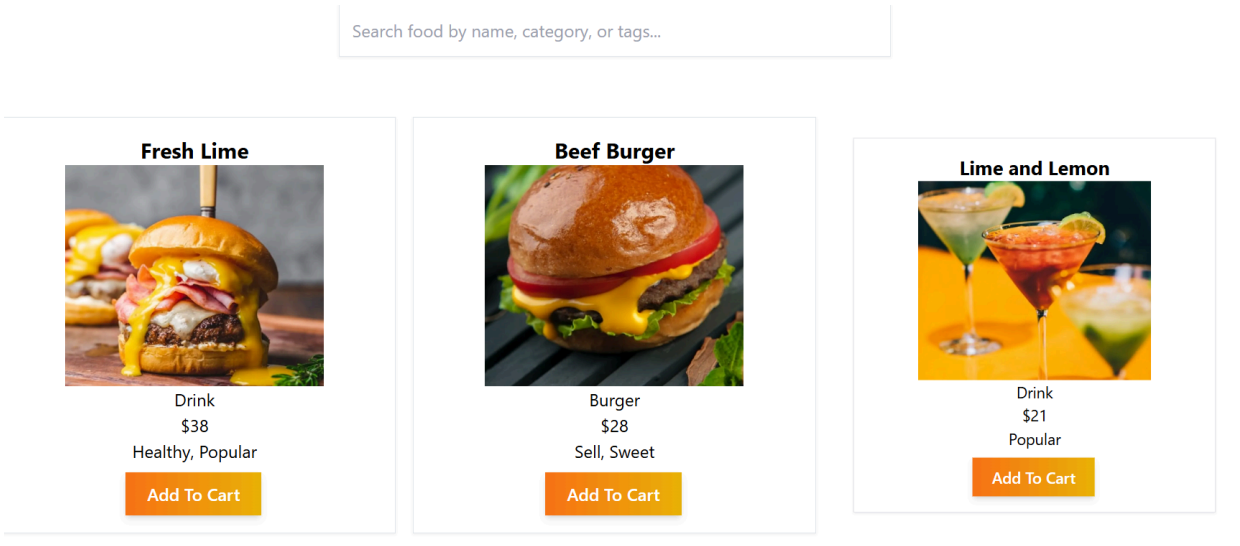
- Displays a list of products with essential details such as product name, price, image.
- Allows users to  quickly view product details and add products to cart .
- Supports filtering and sorting of products based on various criteria such as name, category and tags.

➔ FetchingDataDynamically

```
const Shop = () => {
  const [food, setFood] = useState<Food[]>([]);
  const [searchQuery, setSearchQuery] = useState("");

  useEffect(() => {
    async function fetchFood() {
      const fetchedFood: Food[] = await client.fetch(allFood);
      setFood(fetchedFood);
    }
    fetchFood();
  }, []);
```

➔ Output 👇👇



## 2.Product Detail Component :

➢ Created individual product detail pages using dynamic routing in Next.js.which included fields like:

- Product Name
- Product Description
- Product Description
- Price
- Ratings
- Add To Wishlist
- Add to Cart

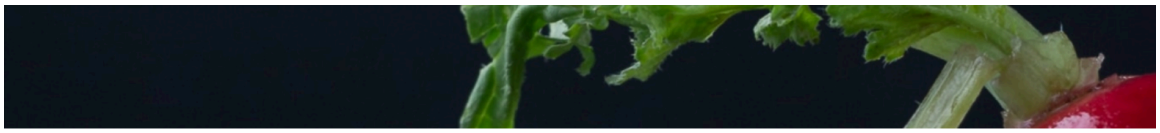
### Key Features :

- BY clicking on single product it enables it essential details such as Name, Price, Description, and Ratings.
- Show high quality product Images from different angles.
- Displays customer reviews and ratings.
- Allows users to customize their orders such as extra cheese, extra sauce and extra spices.

```
interface ProductPageProps {
  params: { slug: string };
}

// Data fetching in the server-side function
async function getFood(slug: string): Promise<Food> {
  return client.fetch(
    groq`*[_type == "food" && slug.current == $slug][0]{
      _id,
      name,
      _type,
      image,
      price,
      discription,
    }`,
    { slug }
  );
}
```

➔ Individual Product Detail



**Lime and Lemon**

$21

4.5

refreshing drinks with lemonate flavour and mint.

Add To Cart

## 3.Cart Component :

➢ Added Add to Cart functionality to add the items in the cart,which displays added items , quantity and total price of the product.

**Key Features :**

- Displays a summary of the product in the cart including product name , quantity and subtotal.
- Show a list of products with options to edit or remove products in the cart .
- Provide clear and prominent call-to-action(CTA) to proceed to checkout.
- Displays notifications such as “Items added to cart ” or “items removed from cart ” to inform users about changes made to cart.

**Prepared by :** **Narmeen Zubair**

➔ Code Snippets

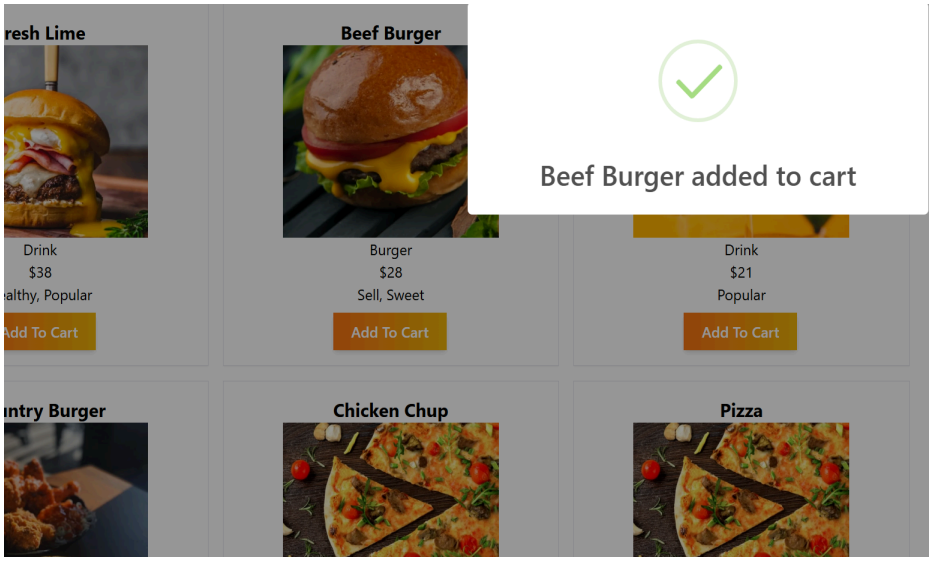
```
const Cart = () => {
  const [cartItems, setCartItems] = useState<Food[]>([]);

  useEffect(() => {
    setCartItems(getCartItems());
  }, []);
```

```
export const addtoCart = (food:Food) =>{
    const cart:Food[] = JSON.parse(localStorage.getItem('cart') || '[]')

    const existingFoodIndex= cart.findIndex(item =>item._id === food._id)
    if (existingFoodIndex> -1){
        cart[existingFoodIndex].inventory += 1
    }
    else{
        cart.push({
            ...food,inventory:1
        })
    }
    localStorage.setItem('cart',JSON.stringify(cart))
}
```

➔ Output



# 4.Wishlist Component :

- ➢ Created a wishlist component that allows users to save the product for future reference.
  - Used local storage management tools to persist data.

## Key Features :

- Show a list of products in the wishlist with options to remove each product.
- Displays notifications such as “Product added to wishlist ” to inform users about changes made to wishlist.

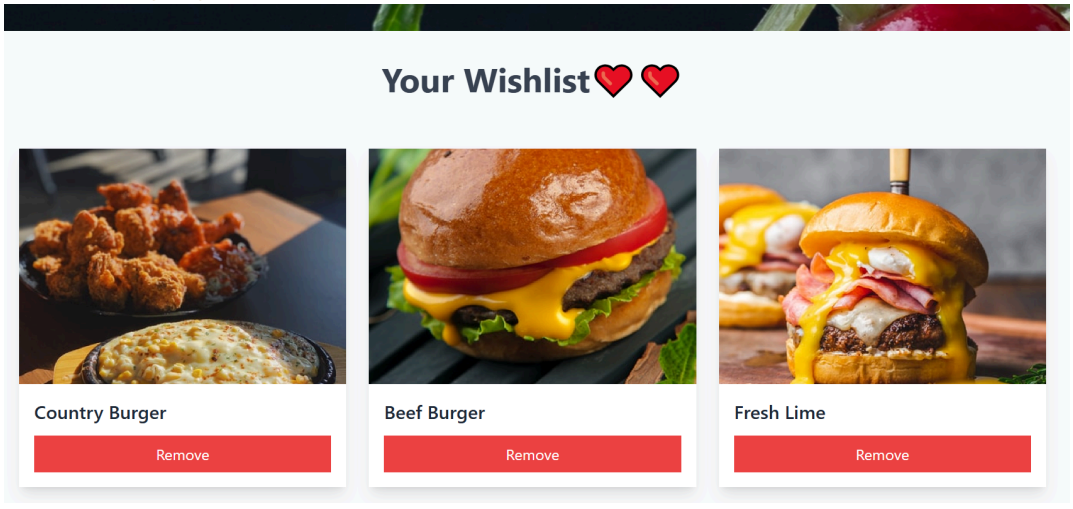
➔ Code Snippets

```
export default function WishlistButton({ item }: { item: WishlistItem }) {
  const [wishlist, setWishlist] = useState<WishlistItem[]>([]);

  useEffect(() => {
    const storedWishlist = JSON.parse(localStorage.getItem('wishlist') || '[]') as WishlistI
    setWishlist(storedWishlist);
    console.log("Wishlist loaded from localStorage:", storedWishlist);
  }, []);

  const addToWishlist = (item: WishlistItem, setWishlist: React.Dispatch<React.SetStateActio
    let wishlist = JSON.parse(localStorage.getItem("wishlist") || "[]");
```

OutPut 👇👇:



# 4.SearchBar Filtration :

➢ Implemented search functionality to filter products by names or tags.

## Key Features :

- Allows users to search for products using specific keywords
- Enables users to filter search results by category

```
const filteredFood = food.filter(
  (item) =>
    item.name.toLowerCase().includes(searchQuery.toLowerCase()) ||
    item.category.toLowerCase().includes(searchQuery.toLowerCase()) ||
    (item.tags ? item.tags.toString() : "").split(",")

    .some((tag) => tag.toLowerCase().includes(searchQuery.toLowerCase()))

);
```

➔ Output 👇👇: