Report: Implementing a Dynamic Product Listing Component

Project Day 4 - Building Dynamic Frontend Components

Objective:
The primary objective of Day 4 is to design and develop dynamic frontend components that can display marketplace data fetched from Sanity CMS or external APIs. This process focuses on modularity, reusability, and applying real-world development practices to build scalable and responsive web applications.

Task Overview

Objective:

Build a Product Listing Component for a marketplace.

Requirements:

1. Fetch product data dynamically using Sanity CMS or an external API.

2. Display the data in a grid layout of cards with the following details:
   - **o Product Name**
   - **o Price**
   - **o Image**
   - **o Stock Status**

3. Ensure responsiveness across devices.

4. Implement modularity by breaking the component into smaller, reusable parts. Tools & Technologies:

• **Framework**: React or Next.js
• **CMS**: Sanity CMS
• **Styling**: Tailwind CSS or plain CSS
• **State Management**: React Hooks.

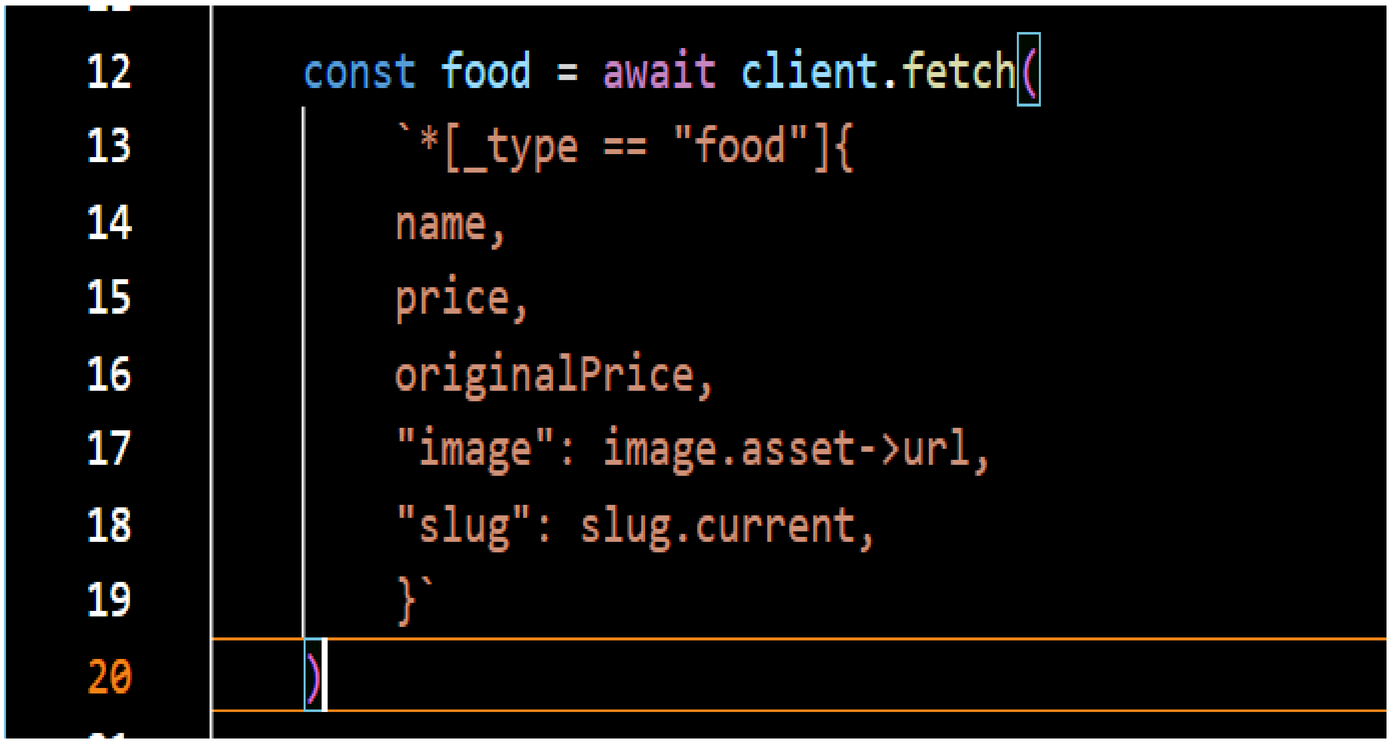
1. Implementation Plan:

1. **Set Up Data Fetching**:
   o Integrate **Sanity CMS** or **API** endpoints to fetch the product data dynamically.
   o Use React hooks (**useEffect**) for data fetching and (**useState**) to store and manage the data.

2. **Design Reusable Components**:
   o Break down the Product Listing Component into smaller parts:
   ▪ **Product Card Component**: Displays individual product details.
   ▪ **Grid Layout Component**: Arranges the product cards in a responsive grid.

3. **Apply Responsive Design**:
   o Use Tailwind CSS or CSS Grid/Flexbox to ensure the grid layout adapts to all screen sizes.

4. **Enhance User Experience**:
   o Highlight important details like stock status with conditional formatting. o Add hover effects for better interactivity.



```
  const food = await client.fetch(
    `*[_type == "food"]{
    name,
    price,
    originalPrice,
    "image": image.asset->url,
    "slug": slug.current,
    }`
  )
```

2. **Product Detail Component**

Objective:

Develop individual product detail pages using dynamic routing in Next.js. These pages will display detailed information about each product, including:

- **Name**
- **Product Description**
- **Price**
- **Category**
- **Stock Availability**

Implementation Plan:

1. Dynamic Routing:
- o Create dynamic routes using the [id].tsx file in the pages/products directory.
- o Fetch product data based on the product ID from a CMS like Sanity or an API.

2. Data Fields:

Each product detail page should include the following fields:

o Product Description:

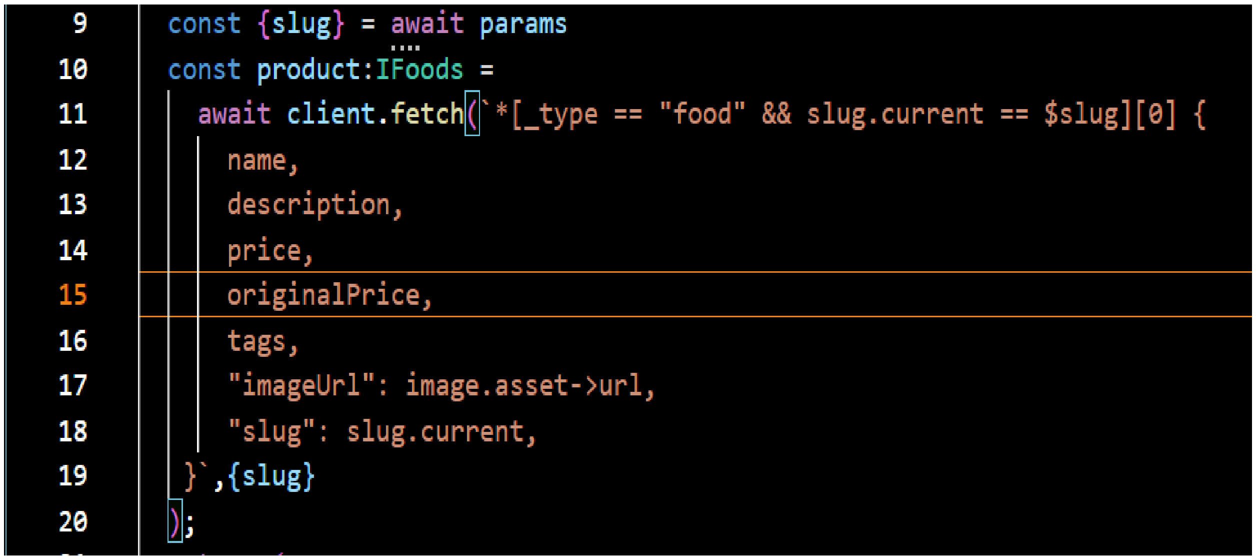
A detailed explanation of the product, fetched from the backend.

o Price:

Displayed prominently for clear visibility.

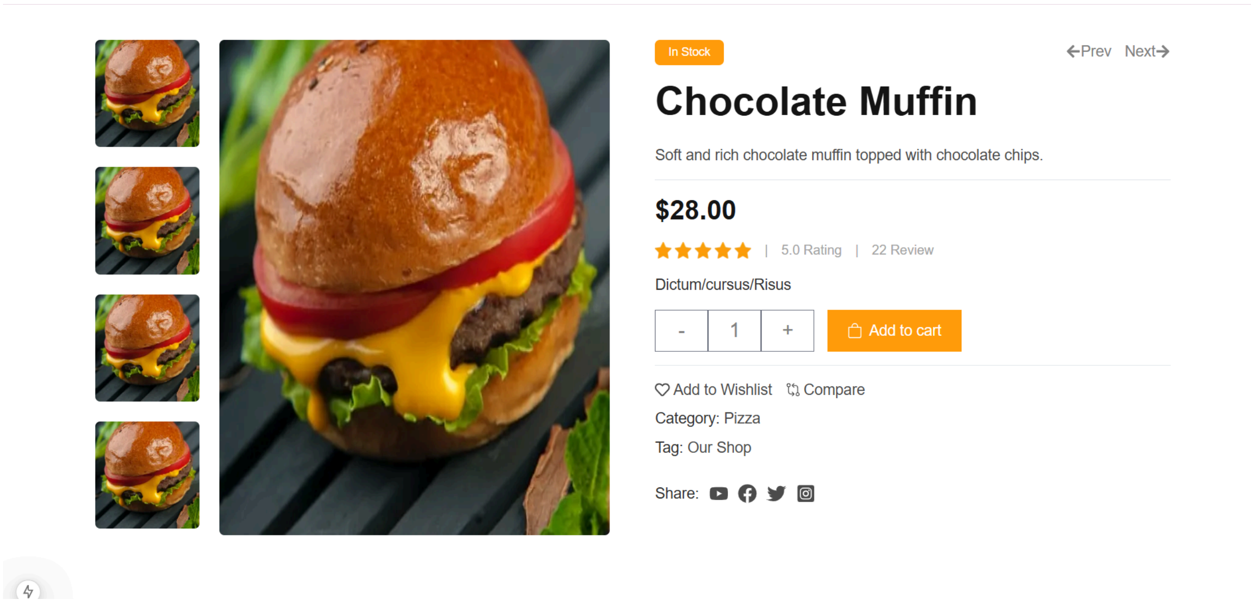
3. Integration with Product Listing:
- o Link each product card in the Product Listing Component to its corresponding detail page using the Link component in Next.js.

4. Styling and Layout:
- o Use Tailwind CSS or plain CSS for a clean and responsive design.
- o Ensure the layout highlights the product description and price for user clarity.



```
const {slug} = await params
const product:IFoods =
  await client.fetch(`*[_type == "food" && slug.current == $slug][0] {
    name,
    description,
    price,
    originalPrice,
    tags,
    "imageUrl": image.asset->url,
    "slug": slug.current,
 }`,{slug}
);
```

Display of Product Detail Page



In Stock

←Prev Next→

Chocolate Muffin

Soft and rich chocolate muffin topped with chocolate chips.

$28.00

★★★★★ | 5.0 Rating | 22 Review

Dictum/cursus/Risus

- 1 + Add to cart

Add to Wishlist Compare

Category: Pizza

Tag: Our Shop

Share:

3: Cart Component

Objective:

To create a Cart Component that displays the items added to the cart, their quantity, and the total price of the cart dynamically.

Implementation Plan:

1. State Management:

o Use React state or a state management library like Redux for storing cart data.

2. Cart Data:

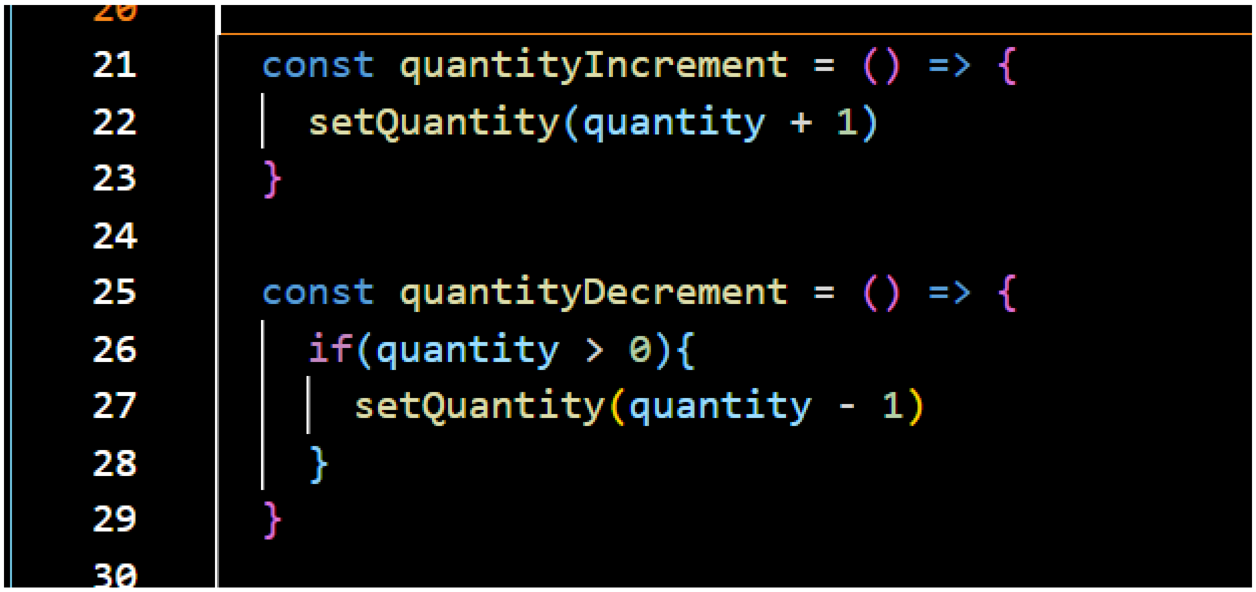
o Include details for each product in the cart:

▪ Product Name
▪ Price
▪ Quantity

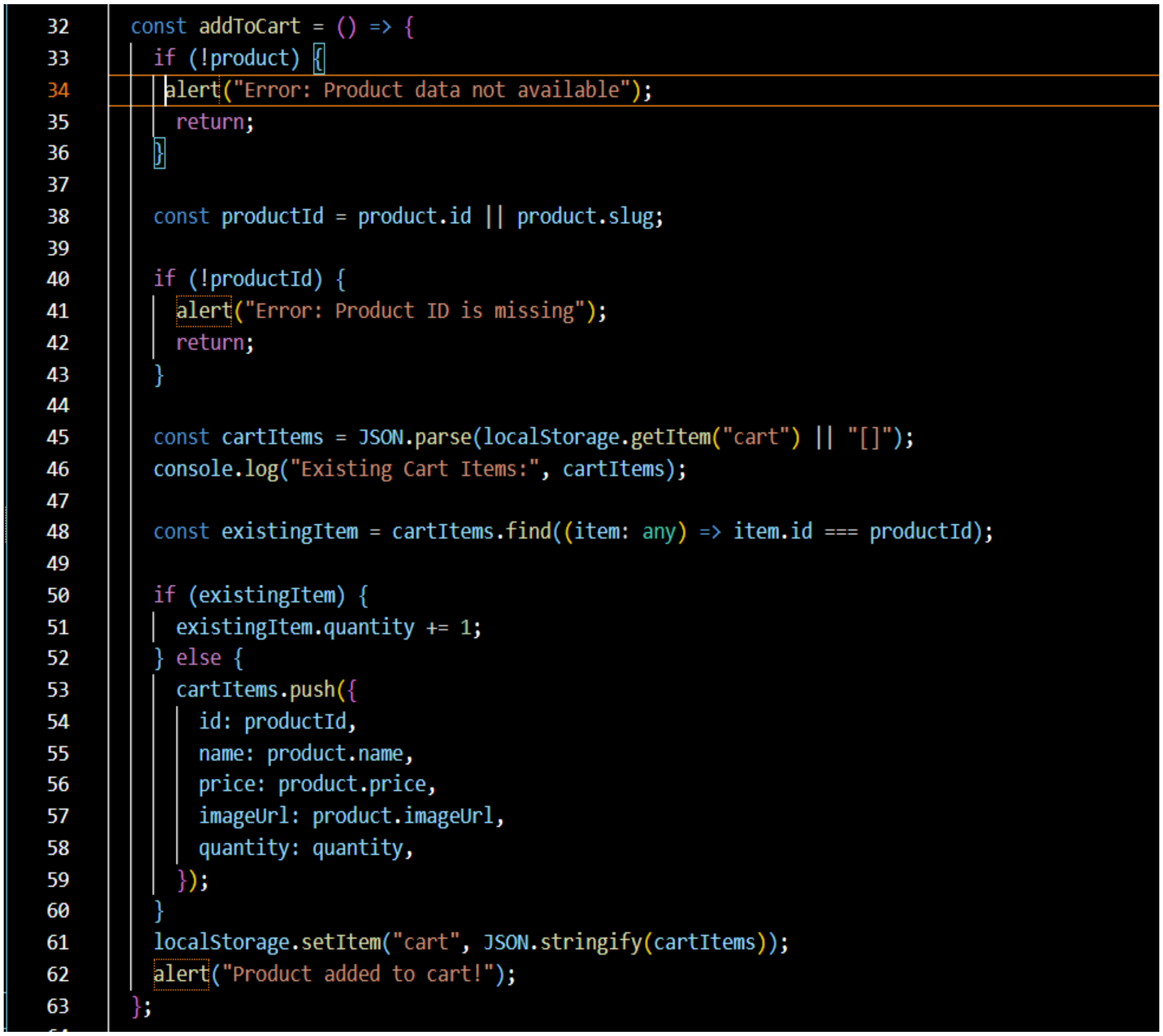
o Calculate and display the total price dynamically based on the items in the cart.

3. Cart Interactions:

o Allow users to increase or decrease the quantity of items.
o Automatically update the total price when the quantity changes.

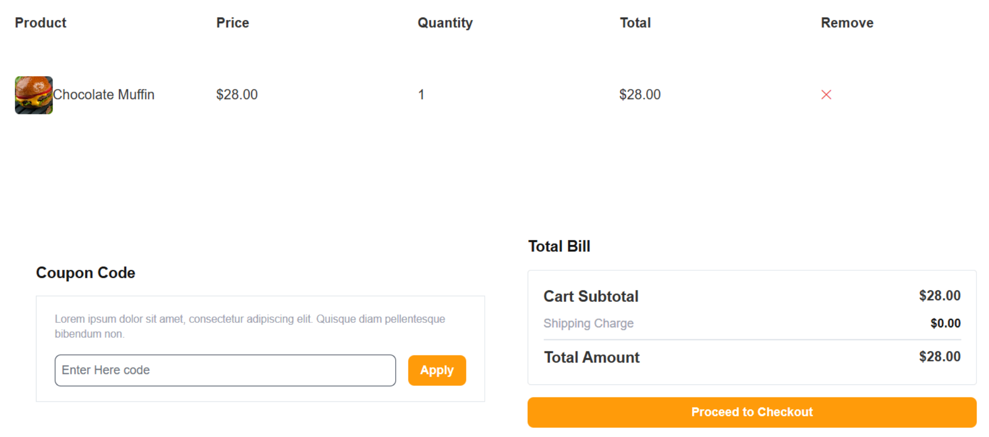


```
const quantityIncrement = () => {
  setQuantity(quantity + 1)
}

const quantityDecrement = () => {
  if(quantity > 0){
    setQuantity(quantity - 1)
  }
}
```



```
const addToCart = () => {
  if (!product) {
    alert("Error: Product data not available");
    return;
  }

  const productId = product.id || product.slug;

  if (!productId) {
    alert("Error: Product ID is missing");
    return;
  }

  const cartItems = JSON.parse(localStorage.getItem("cart") || "[]");
  console.log("Existing Cart Items:", cartItems);

  const existingItem = cartItems.find((item: any) => item.id === productId);

  if (existingItem) {
    existingItem.quantity += 1;
  } else {
    cartItems.push({
      id: productId,
      name: product.name,
      price: product.price,
      imageUrl: product.imageUrl,
      quantity: quantity,
    });
  }
  localStorage.setItem("cart", JSON.stringify(cartItems));
  alert("Product added to cart!");
};
```

Display of Cart Page



Product	Price	Quantity	Total	Remove
Chocolate Muffin	$28.00	1	$28.00	×

Coupon Code

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque diam pellentesque bibendum non.

Total Bill

Cart Subtotal	$28.00
Shipping Charge	$0.00
Total Amount	$28.00

Features Implemented:

1. Dynamic Item Display:

o Each item in the cart is displayed with its name, price, and quantity.
o Subtotal for each item is dynamically calculated.

2. Quantity Update:

o Buttons to increase (+) or decrease (-) the quantity of an item.
o Quantity cannot go below 1.

3. Total Price Calculation:

o The total price updates dynamically as items are added or quantities are changed.

4. Remove Item:

o Users can remove individual items from the cart.

Conclusion

On Day 4 of building dynamic frontend components for a marketplace, the focus was on creating modular, reusable, and responsive components. The following key components were successfully implemented:

1. **Product Listing Component**:
   o Dynamically displayed products in a grid layout with details such as **product name, price, image,** and **stock status**.

2. **Product Detail Component:**
   o Built individual product pages using dynamic routing in **Next.js**, including fields like **product description, price**, and **image**.

3. **Cart Component**:
   o Displayed items added to the cart, quantity management, and total price calculation with dynamic updates