Report: Implementing a Dynamic Product Listing Component
Prepared By: Syeda Laiba
Prepared For: Project Day 4 - Building Dynamic Frontend Components

---

Objective

The primary goal of Day 4 was to design and develop dynamic frontend components capable of displaying marketplace data fetched from Sanity CMS or external APIs. The focus was on ensuring 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:
    - Product Name
    - Price
    - Image
    - 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

Implementation Plan

1. Set Up Data Fetching

- Integrate Sanity CMS or API endpoints to fetch the product data dynamically.
- Use React hooks (`useEffect` for data fetching, `useState` for storing and managing data).

2. Design Reusable Components

- 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

- Use Tailwind CSS or CSS Grid/Flexbox to ensure the grid layout adapts to all screen sizes.

4. Enhance User Experience

- Highlight important details like stock status with conditional formatting.
- Add hover effects for better interactivity.

Product Detail Component

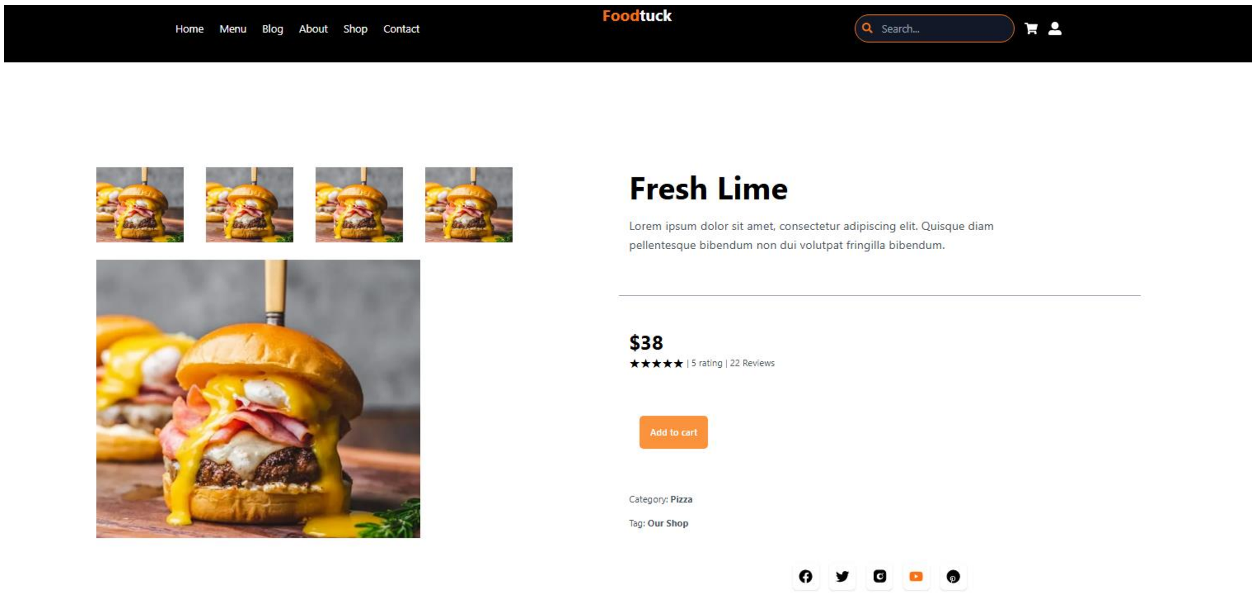
Objective

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

Implementation Plan

1. **Dynamic Routing:**
   - Create dynamic routes using the `[id].tsx` file in the `pages/products` directory.
   - Fetch product data based on the product ID from a CMS like Sanity or an API.
2. **Data Fields:**
   - Include product details like:
     - **Product Description**
     - **Price**
3. **Integration with Product Listing:**
   - 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:**
   - Use Tailwind CSS or plain CSS for a clean and responsive design.
   - Ensure the layout highlights the product description and price for user clarity.

UI Display OF Product Detail Page:



Fresh Lime

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

$38

★★★★★ | 5 rating | 22 Reviews

Category: Pizza

Tag: Our Shop

Step 3: Search Bar with Price Filter

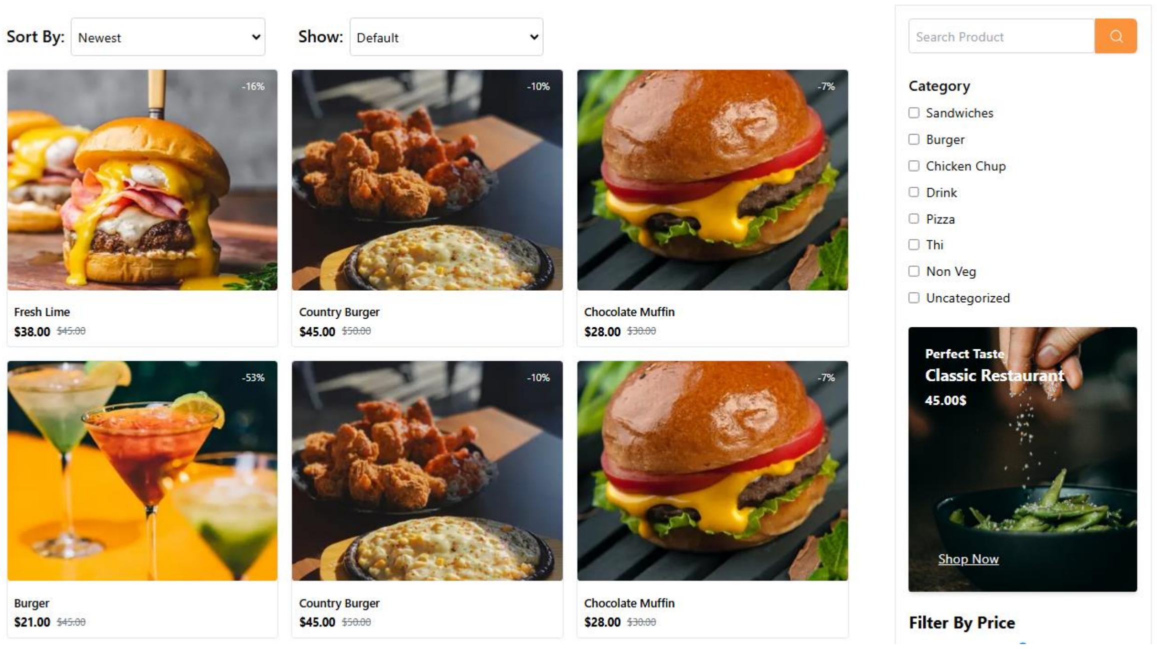
Objective

Implement a search bar and price filters to enhance the product browsing experience.

Implementation Plan

1. **Search Bar Functionality:**
   - Filter products based on their name or associated tags.
   - Update the product list in real-time as the user types.

UI Display :



Sort By: Newest

Show: Default

2. **Price Filtering:**
   - Add options to sort products by price in ascending or descending order.
   - Combine the price filter with the search bar and category filter for seamless interaction.

Step 4: Cart Component

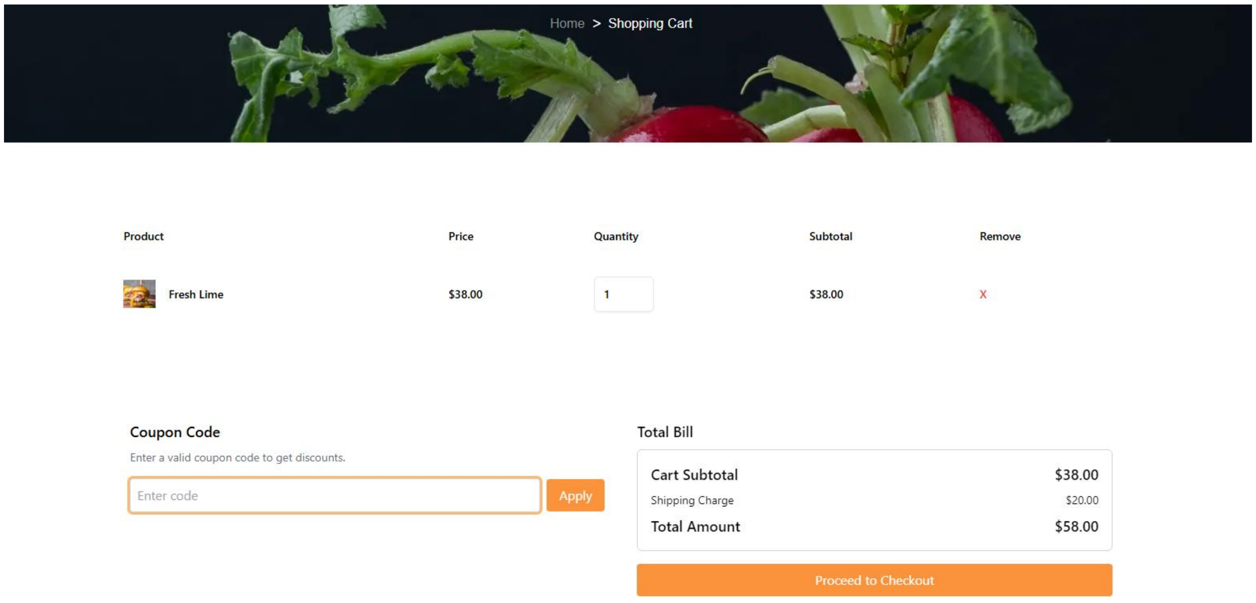
Objective

Create a Cart Component to display the items added to the cart, their quantity, and the total price dynamically.

Implementation Plan

1. **State Management:**
   - Use React state or a state management library like Redux for storing cart data.
2. **Cart Data:**
   - Include details for each product in the cart:
     - Product Name
     - Price
     - Quantity
   - Calculate and display the total price dynamically based on the items in the cart.
3. **Cart Interactions:**
   - Allow users to increase or decrease the quantity of items.
   - Automatically update the total price when the quantity changes.

UI Display Of Cart Page:



Home > Shopping Cart

Product	Price	Quantity	Subtotal	Remove
Fresh Lime	$38.00	1	$38.00	X

Coupon Code

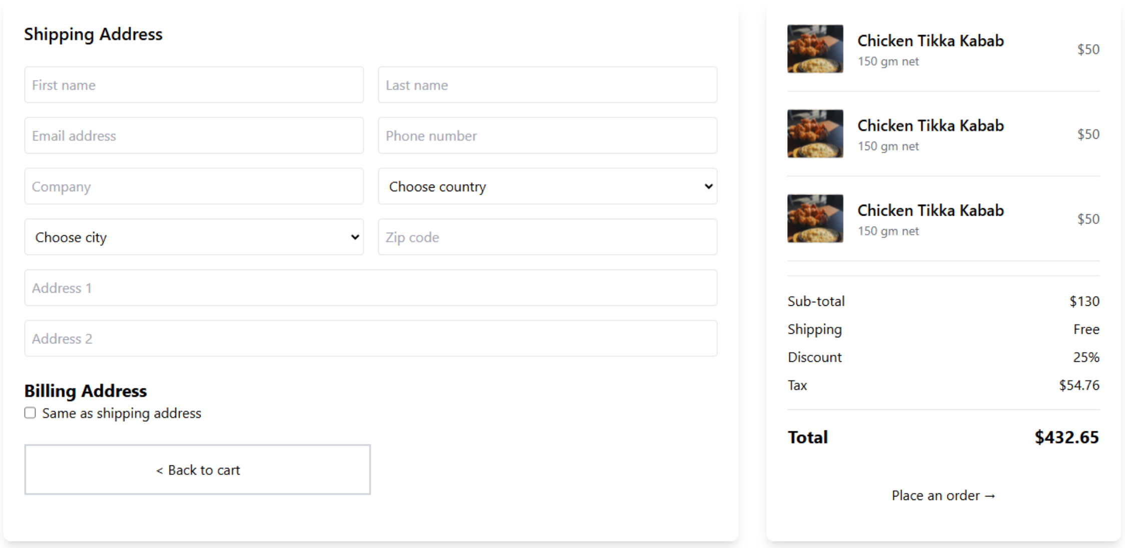
Enter a valid coupon code to get discounts.

Total Bill

Cart Subtotal	$38.00
Shipping Charge	$20.00
Total Amount	$58.00

Features Implemented:

1. **Dynamic Item Display:**
   - Each item in the cart is displayed with its name, price, and quantity.
   - Subtotal for each item is dynamically calculated.
2. **Quantity Update:**
   - Buttons to increase (+) or decrease (-) the quantity of an item.
   - Quantity cannot go below 1.
3. **Total Price Calculation:**
   - The total price updates dynamically as items are added or quantities are changed.
4. **Remove Item:**
   - Users can remove individual items from the cart.


Shipping Address

First name	Last name
Email address	Phone number
Company	Choose country
Choose city	Zip code
Address 1	
Address 2	

Billing Address

☐ Same as shipping address

< Back to cart

Chicken Tikka Kabab	150 gm net	$50
Chicken Tikka Kabab	150 gm net	$50
Chicken Tikka Kabab	150 gm net	$50

Sub-total	$130
Shipping	Free
Discount	25%
Tax	$54.76
Total	**$432.65**

Place an order →

Step 6: Notifications Component

Objective

Create a Notifications Component to display real-time alerts for user actions, such as adding items to the cart, encountering errors, or completing a successful purchase.

Implementation Plan

1. **Real-Time Alerts:**
   - Use toast notifications or modal windows to display alerts.
   - Display notifications for actions like:
     - Item added to the cart
     - Errors (e.g., "Out of stock")
     - Successful actions (e.g., "Purchase complete")
2. **Integration:**
   - Trigger notifications at appropriate moments in the app, such as adding to the cart or completing a transaction.

Conclusion

On Day 4, the focus was on building dynamic frontend components for the marketplace. Key components were successfully implemented, including:

1. **Product Listing Component:** Dynamically displayed products in a grid layout with details such as product name, price, image, and stock status.
2. **Product Detail Component:** Built individual product pages using dynamic routing in Next.js.
3. **Search Bar and Filters:** Implemented functionality to filter products by name or tags and added price filters.
4. **Cart Component:** Displayed items added to the cart with quantity management and total price calculation.

This work establishes a solid foundation for the marketplace, with modular, reusable, and responsive components.