Comforty: Luxury Chairs and Sofas

Technical Documentation
Day 4: Building Dynamic Frontend Components

Table of Contents

Introduction

On Day 4, we focused on building **dynamic frontend components** for the Comforty marketplace. This included creating interactive and responsive components such as product listings, product detail pages, category filters, search bars. The goal was to ensure a seamless and engaging user experience while dynamically fetching and rendering data from Sanity CMS.

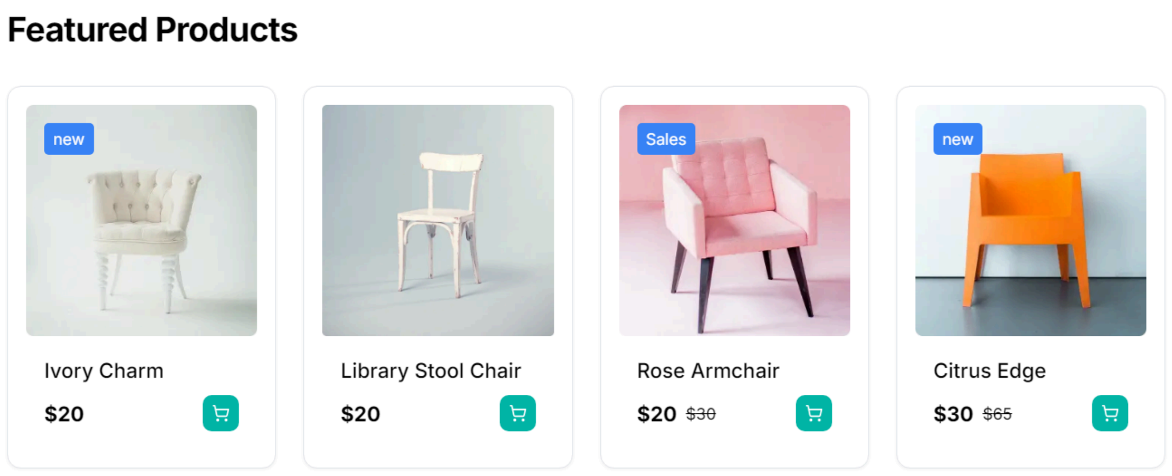
Objective

To design **dynamic, reusable components** for the **Comforty Store**, integrating Sanity CMS and ensuring a scalable and responsive user experience.

Components Built

Products Component

- Dynamic rendering of product data fetched from Sanity CMS.
- Example:



Featured Products

new
Ivory Charm
$20

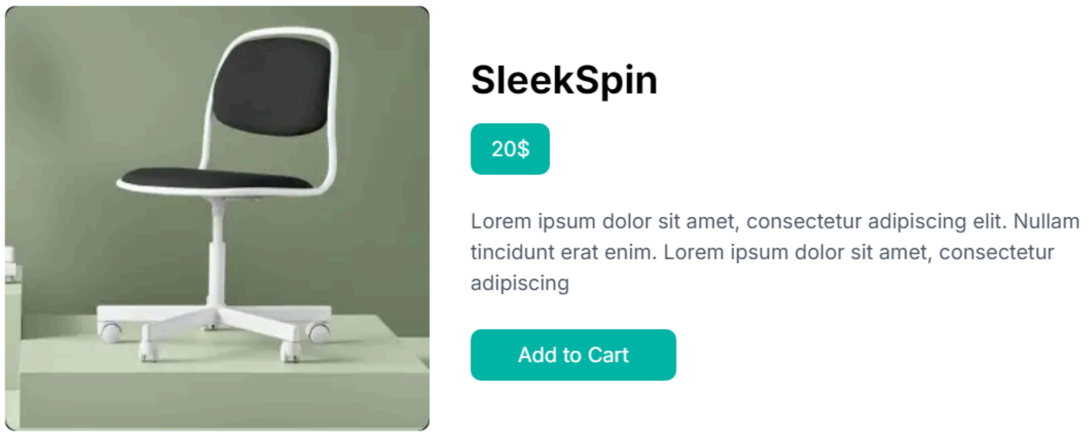
Library Stool Chair
$20

Sales
Rose Armchair
$20 ~~$30~~

new
Citrus Edge
$30 ~~$65~~

Product Detail Page

- Accurate routing and rendering of individual product details.
- Example:



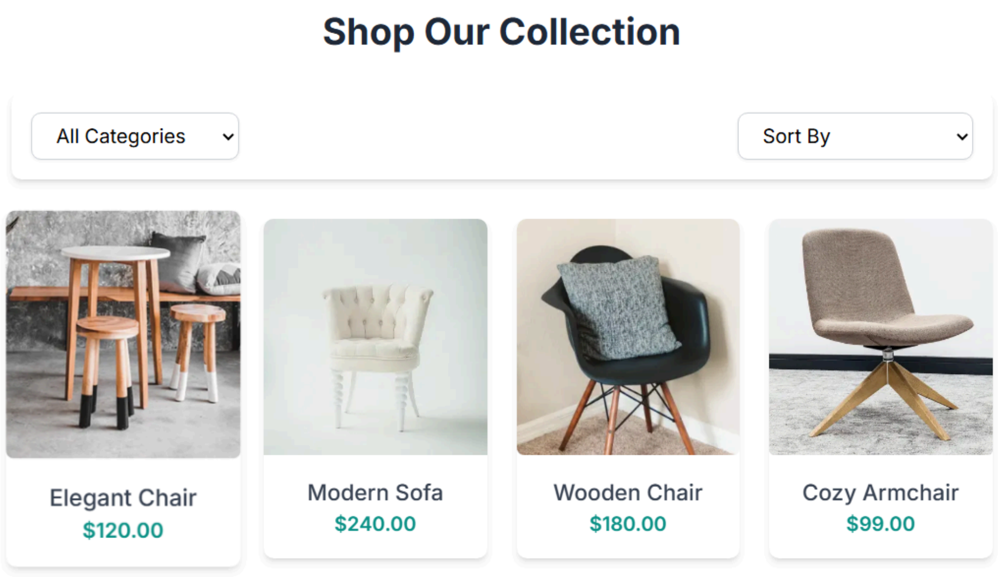
SleekSpin

20$

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

Category Filters

- Functional category filters for easy navigation.
- Example:

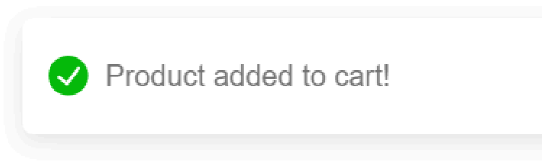


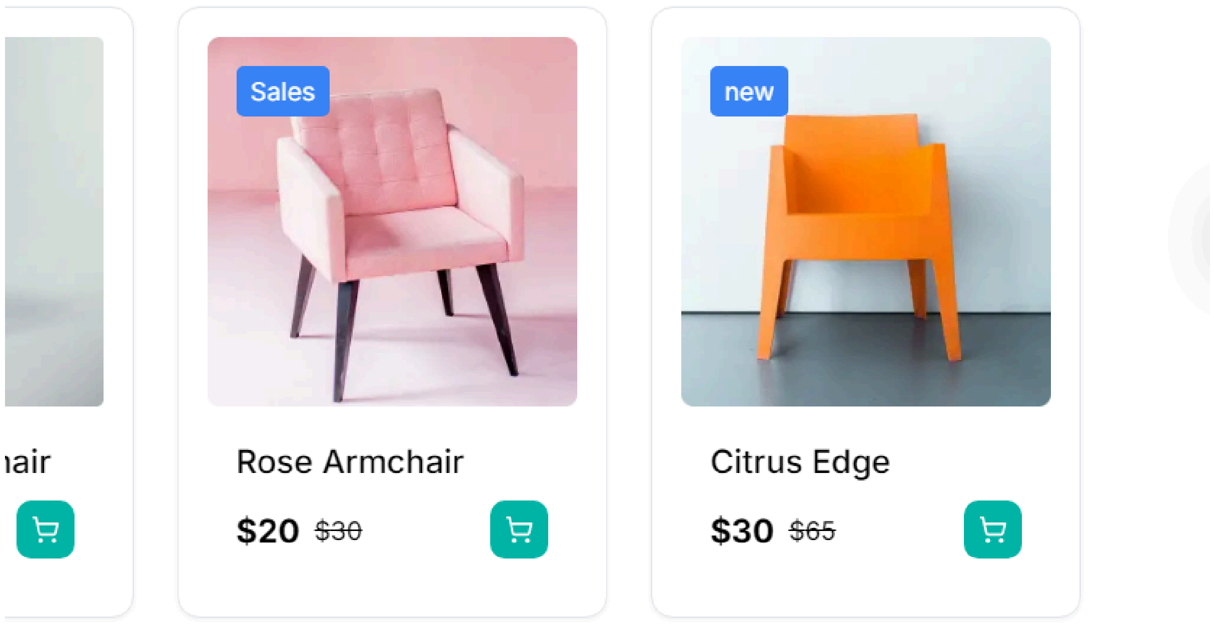
Shop Our Collection

Elegant Chair	$120.00
Modern Sofa	$240.00
Wooden Chair	$180.00
Cozy Armchair	$99.00

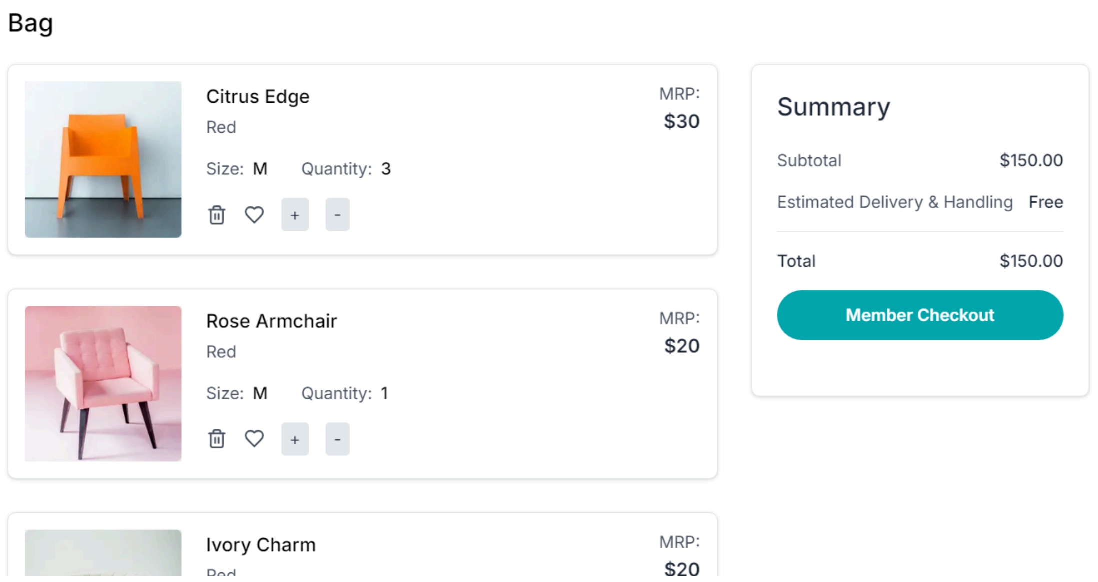
Cart Components

- **Cart**: Add cart items in a cart component when the cart icon is clicked with a notification.
- Example:


Product added to cart!



- **Cart Page**: Provides a detailed view of cart items, allowing quantity adjustments and total price calculation.
- Example:



Bag

Citrus Edge
Red
Size: M   Quantity: 3
MRP: $30

Rose Armchair
Red
Size: M   Quantity: 1
MRP: $20

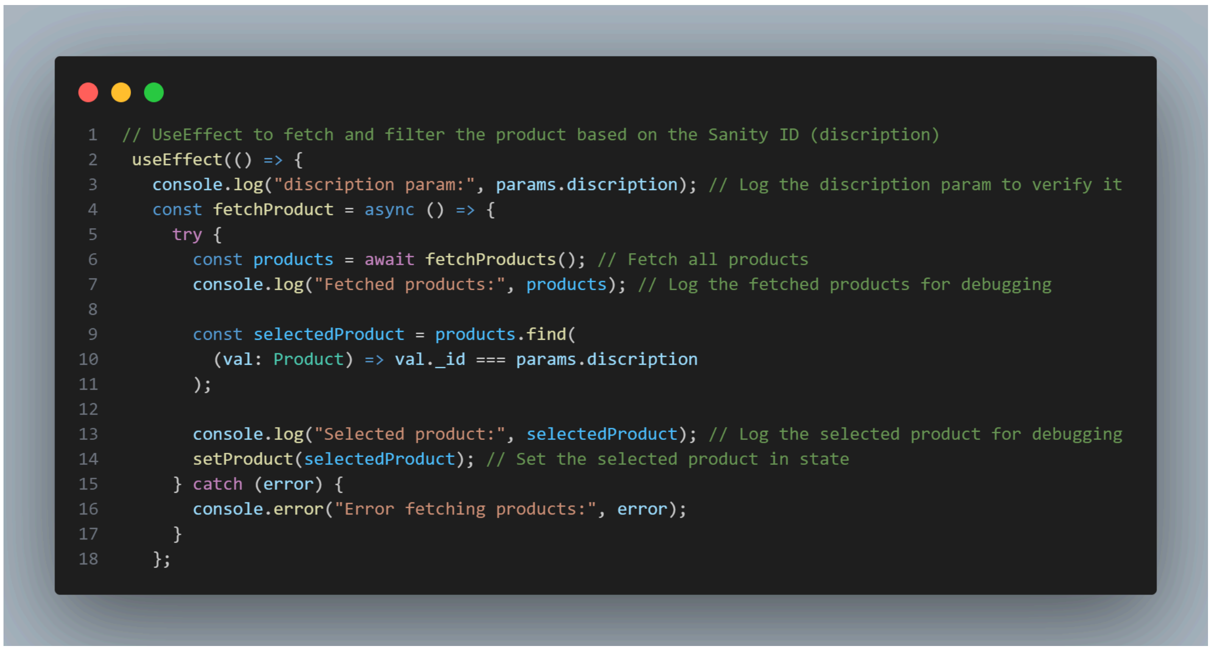
Ivory Charm
MRP: $20

Summary

Subtotal	$150.00
Estimated Delivery & Handling	Free
Total	$150.00

Code Deliverables

Dynamic Routing Logic



```
// UseEffect to fetch and filter the product based on the Sanity ID (discription)
 useEffect(() => {
   console.log("discription param:", params.discription); // Log the discription param to verify it
   const fetchProduct = async () => {
     try {
       const products = await fetchProducts(); // Fetch all products
       console.log("Fetched products:", products); // Log the fetched products for debugging

       const selectedProduct = products.find(
         (val: Product) => val._id === params.discription
       );

       console.log("Selected product:", selectedProduct); // Log the selected product for debugging
       setProduct(selectedProduct); // Set the selected product in state
     } catch (error) {
       console.error("Error fetching products:", error);
     }
   };
```

Technical Report

Steps Taken

- Set up dynamic routing for product detail pages.
- Integrated Sanity CMS API for fetching product data.
- Built reusable components (`ProductCard`, `ProductList`).
- Implemented category and filters logic.

Challenges Faced

- Ensuring accurate data rendering for dynamic routes.
- Optimizing API calls for better performance.

Solutions Implemented

- Used `useEffect` and `useState` hooks for data fetching and state management.
- Implemented caching for API responses to reduce load times.

Best Practices

- Followed modular component design for reusability.
- Used TypeScript for type safety and cleaner code.
- Wrote unit tests for key components.