

Technical Documentation

Project Overview

This document summarizes the integration of various components into the marketplace project. It includes details about each component, dynamic routing, challenges faced, solutions implemented, and best practices followed during the development.

1. Component Integration

The following components were integrated into the marketplace application:

1.1 Add to Cart Component

- **Purpose:** Allows users to add products to their shopping cart.
- **Steps Taken:**
 - Created a `CartContext` to manage the cart state globally.
 - Implemented the `addToCart` function to add products with quantity control.
 - The cart updates dynamically in real-time and persists across page reloads using `localStorage`.
- **Challenges Faced:** Ensuring the cart is updated across different parts of the application and persisting it on page reloads.
- **Solutions Implemented:** Utilized React Context API for global state management and persisted cart data using `localStorage`.

1.2 Wishlist Component

- **Purpose:** Allows users to add products to their wishlist.
- **Steps Taken:**
 - Created a `WishlistContext` to manage wishlist state globally.
 - Implemented the `addToWishlist` function for adding products to the wishlist.
 - Added success notifications for items added to the wishlist.
- **Challenges Faced:** Managing wishlist state across multiple pages and ensuring smooth data transfer between pages.
- **Solutions Implemented:** Used React Context API to manage the wishlist state globally and ensure updates are reflected across the application.

1.3 Product Comparison Component

- **Purpose:** Allows users to compare multiple products side by side based on selected features.
- **Steps Taken:**
 - Created a `ComparisonContext` to track selected products for comparison.
 - Displayed selected products with key features side by side for comparison.
- **Challenges Faced:** Displaying large datasets for comparison in an organized manner.

- **Solutions Implemented:** Filtered the relevant features of products to display them clearly and efficiently.

1.4 Notifications Component

- **Purpose:** Notifies users about specific actions like adding items to the cart or wishlist.
- **Steps Taken:**
 - Implemented a notification system that shows success/failure messages when items are added to the cart or wishlist.
 - Notifications appear dynamically and disappear after a set period.
- **Challenges Faced:** Managing notification visibility for multiple user actions.
- **Solutions Implemented:** Used state management (via React Context) to control notification visibility across the app.

1.5 Analytics Dashboard Component

- **Purpose:** Displays key metrics related to user activity, sales performance, and other important data for administrators.
- **Steps Taken:**
 - Developed a dashboard that presents simulated analytics data (e.g., product views, cart abandonment rates).
 - The dashboard is designed to be dynamic with placeholders for future data.
- **Challenges Faced:** Integrating real-time data into the dashboard.
- **Solutions Implemented:** Static data was used initially as placeholders, with plans to integrate real-time data from external APIs in the future.

1.6 Discount and Promotion Component

- **Purpose:** Allows users to apply discounts and promotions to product prices.
- **Steps Taken:**
 - Created a dynamic discount system that applies percentage-based discounts on product prices.
 - Displayed promotional offers on product pages.
- **Challenges Faced:** Handling multiple discount types and ensuring accurate price calculations.
- **Solutions Implemented:** Applied discount percentages dynamically using fetched data, and displayed the discount details on the product pages.

1.7 Related Products Component

- **Purpose:** Suggests similar or related products based on user preferences or current product category.
- **Steps Taken:**
 - Created a related products section based on categories, using a dynamic query to fetch relevant products.
 - Displayed related products in a grid or carousel format for easy browsing.

- **Challenges Faced:** Ensuring relevant products are suggested and displayed efficiently.
- **Solutions Implemented:** Filtered related products by category to maintain relevance, providing a more personalized shopping experience for users.

1.8 Social Media Sharing Component

- **Purpose:** Allows users to share products and promotions on social media platforms such as Facebook, Twitter, and WhatsApp.
- **Steps Taken:**
 - Added social media share buttons under the "Add to Cart" section on the product page.
 - Included a call-to-action: "Share this product with your friends on social media!"
 - Linked share buttons to external social media URLs for Facebook, Twitter, and WhatsApp.
- **Challenges Faced:** Integrating social media APIs for dynamic sharing.
- **Solutions Implemented:** Used static links for social media sharing, which can be updated in the future with dynamic product information for sharing.

2. Dynamic Routing and Data Fetching

2.1 Dynamic Routing

- **Purpose:** To create a dynamic, SEO-friendly product pages system.
- **Steps Taken:**
 - Implemented dynamic routing for individual product pages using the Next.js file-based routing system.
 - Each product page URL is dynamically generated based on the product's unique ID from Sanity CMS (e.g., `/product/[id]`).
 - Used the `useRouter` hook from Next.js to access the route parameters, specifically the `id` of the product, to fetch and display the product details.

2.2 Data Fetching from Sanity

- **Purpose:** Fetch and display product data dynamically from Sanity CMS.
- **Steps Taken:**
 - Created GROQ queries to fetch the product data based on dynamic product IDs.
 - Integrated Sanity's API using the `client.fetch()` method to fetch product data for each specific product page.
 - Data such as product title, price, images, description, and discount were dynamically fetched and displayed on the respective product pages.
 - Used `useEffect` to fetch product data when the component mounts and ensure the UI is updated when the data is received.
- **Challenges Faced:** Ensuring that dynamic data is fetched efficiently, and handling empty or invalid product data.
- **Solutions Implemented:** Implemented error handling to display a fallback message if no product is found for the given ID.

3. Challenges Faced & Solutions Implemented

1. **State Management:** Managing state for various components like the cart and wishlist, ensuring updates reflect across different pages. **Solution:** Used React Context API to centralize state management, avoiding prop drilling and enabling global state access.
2. **Data Fetching:** Fetching dynamic product data and displaying it efficiently on the front-end. **Solution:** Used GROQ queries with Sanity CMS to fetch product data and implement dynamic pages for product details, related products, and promotions.
3. **UI Consistency:** Maintaining consistent styling and layout across different components. **Solution:** Leveraged Tailwind CSS for responsive design and utility-first classes to ensure uniformity across the application.
4. **Performance Optimization:** Handling performance issues when dealing with large datasets (e.g., product comparisons, related products). **Solution:** Filtered unnecessary data before rendering, optimizing performance by only fetching and displaying relevant information.
5. **Responsive Design:** Ensuring that the components are responsive and look great on different screen sizes. **Solution:** Used Tailwind CSS's built-in responsive utilities to ensure the UI adapts to various devices, providing a seamless user experience.

4. Best Practices Followed

1. **Component Modularity:** Components were designed to be modular and reusable. This ensures scalability and maintainability of the project.
2. **State Management:** The use of React Context API ensures that state is shared across the application, enabling efficient data flow between components.
3. **Responsive UI:** Tailwind CSS was used extensively to create a fully responsive design, ensuring the marketplace works seamlessly across all devices.
4. **Code Optimization:** Efficient data fetching using GROQ queries minimized unnecessary API calls, improving performance.
5. **Future Scalability:** Placeholders were used for certain dynamic content (e.g., analytics and real-time order tracking), making the project ready for future enhancements.

5. Conclusion

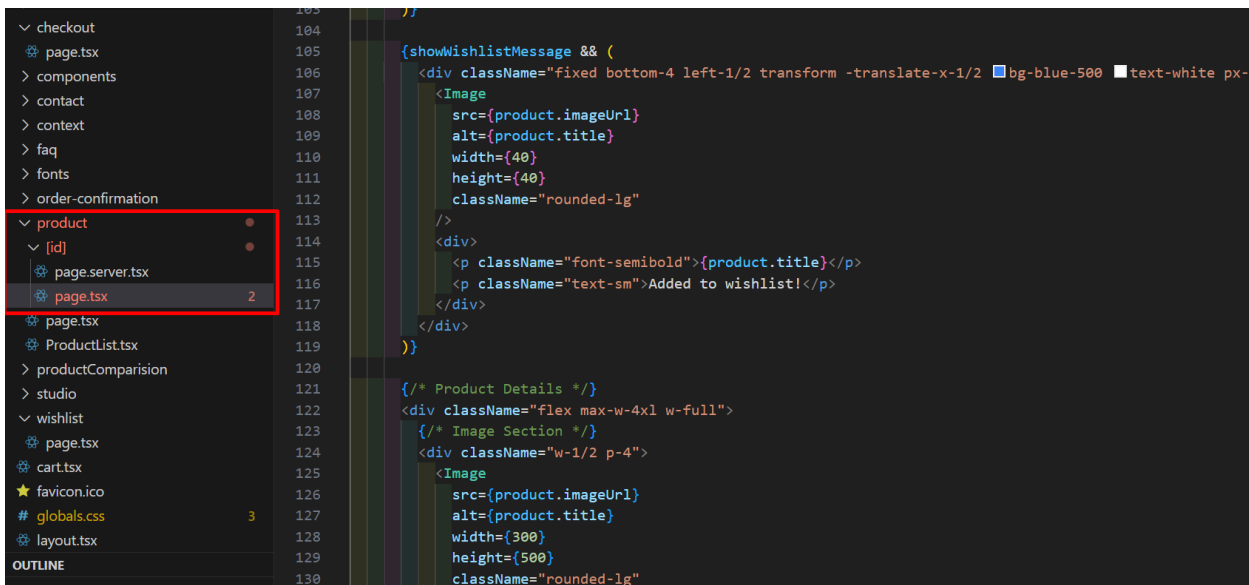
The integration of these components into the marketplace project provides users with essential functionalities such as managing a shopping cart, creating a wishlist, comparing products, viewing discounts, and sharing on social media. By adding dynamic routing and fetching data directly from Sanity CMS, we have ensured that the marketplace is dynamic, SEO-friendly, and easily scalable.

The project is built to be modular and maintainable, using best practices such as React Context API for state management, Tailwind CSS for responsive design, and GROQ queries for efficient data fetching. Although some components, such as the analytics dashboard and order tracking, were implemented with placeholders, the structure is in place for future integrations and dynamic content updates.

This project is well-positioned for future scalability, with room for integrating real-time data fetching, advanced user personalization, and additional functionalities to enhance the user experience.

Here are the screenshots of the project.

1. Dynamic routing snippet.



Modern Cozy

\$20.00

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

Add To Cart

Add To Wishlist

Share this product with your friends on social media:



Product Comparison Page snippet.

Add to Comparison



Remove from Comparison

Add to Comparison

Remove from Comparison

Comparison Details

Clear Comparison

Product	Nordic Spin		Ivory Charm	
Image				
Price	\$30		\$20	
Description	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing		Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing	
Discount	No discount		No discount	

\$20 ~~\$36~~

Add to Comparison

\$20


Add to Comparison

\$40

Add to Comparison

\$20

Add to Comparison




Library Stool Chair

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam...

\$20 ~~\$62~~

Add to Comparison




Nordic Spin

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam...

\$30

Remove from Comparison




Gray Elegance

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam...

\$8

Add to Comparison



Ivory Charm

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam...

\$20

Remove from Comparison

Checkout Page Snippet.

Contact Information

Shipping Address



Order Summary



SleekSpin
Qty: 4

\$80

Subtotal

\$80

Shipping

Free

Total

\$80

Payment Method

Credit/Debit Card

Cash on Delivery


PayPal

Pay with PayPal

You will be redirected to PayPal to complete your payment.

Pay Now

Order page Snippet.



Order Confirmation

Thank You for Your Order!


Your order has been placed successfully. We will send you a confirmation email shortly.


Estimated Delivery Date

Fri Jan 24 2025

Your order will arrive by this date. We'll notify you if there are any delays.

Order Summary

	SleekSpin Qty: 4	\$80
Subtotal		\$80
Shipping		Free
Total		\$80



Social Media Sharing page.



Scandi Dip Set

\$40.00

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

Add To Cart

Add To Wishlist

Share this product with your friends on social media:



Notification Page Snippet.

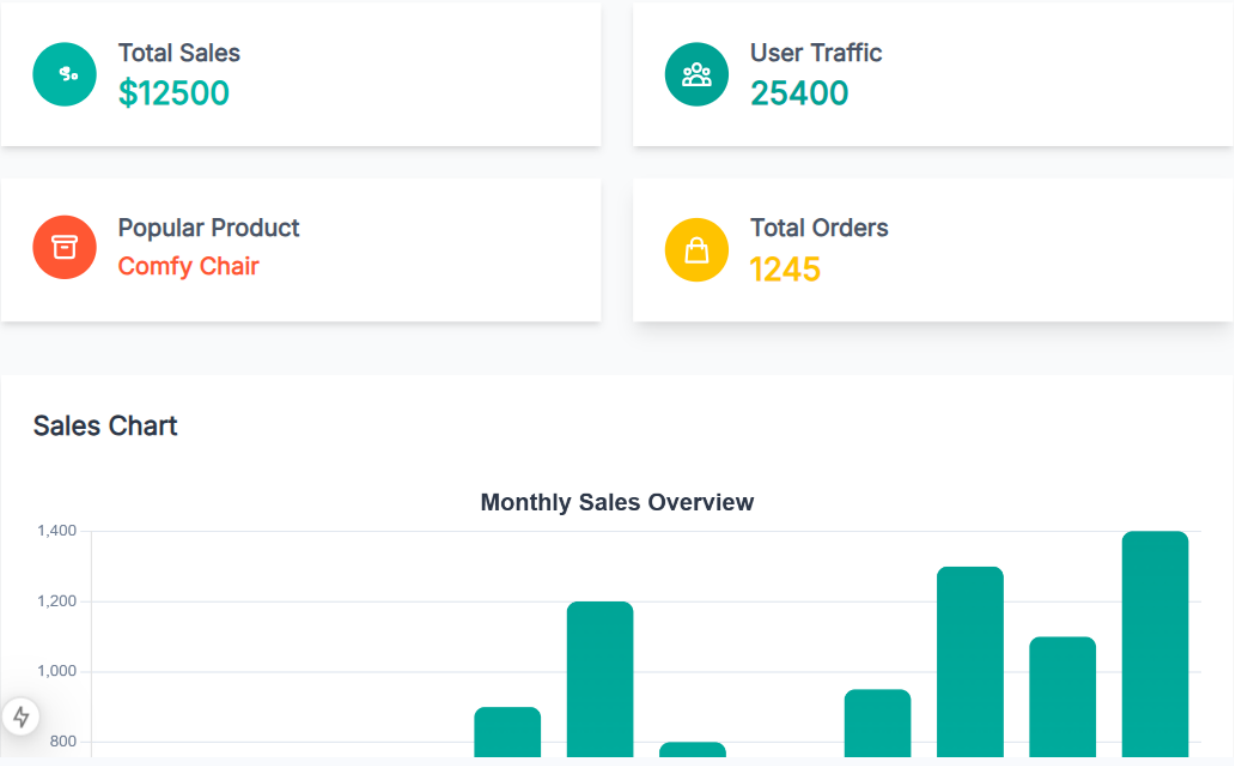
Add To Wishlist



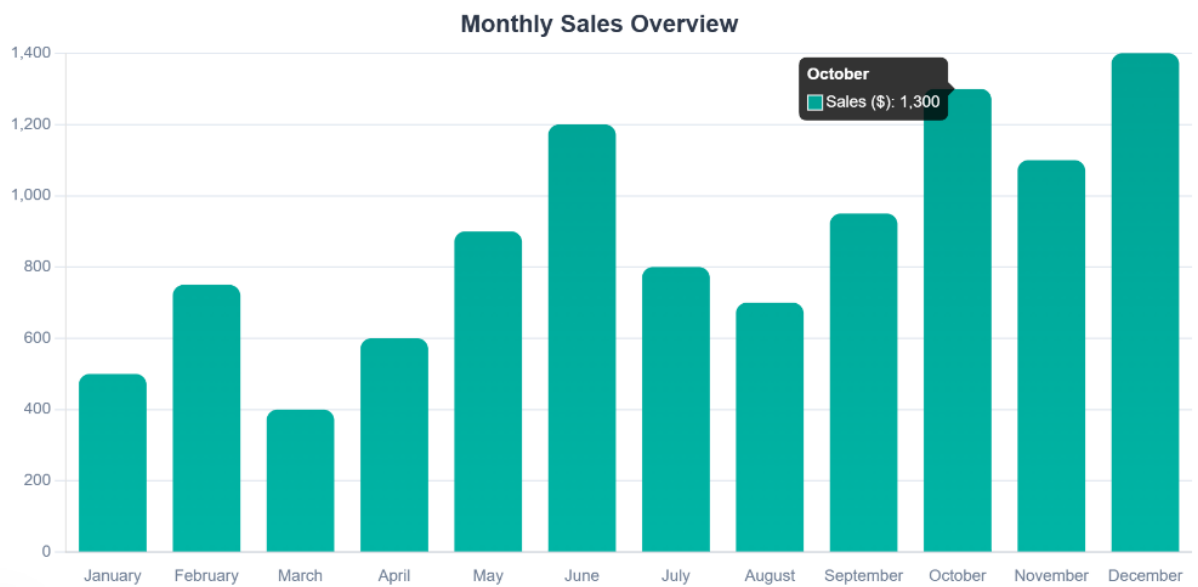
Scandi Dip Set
Added to cart!

Analytics Dashboard snippet.

Analytics Dashboard



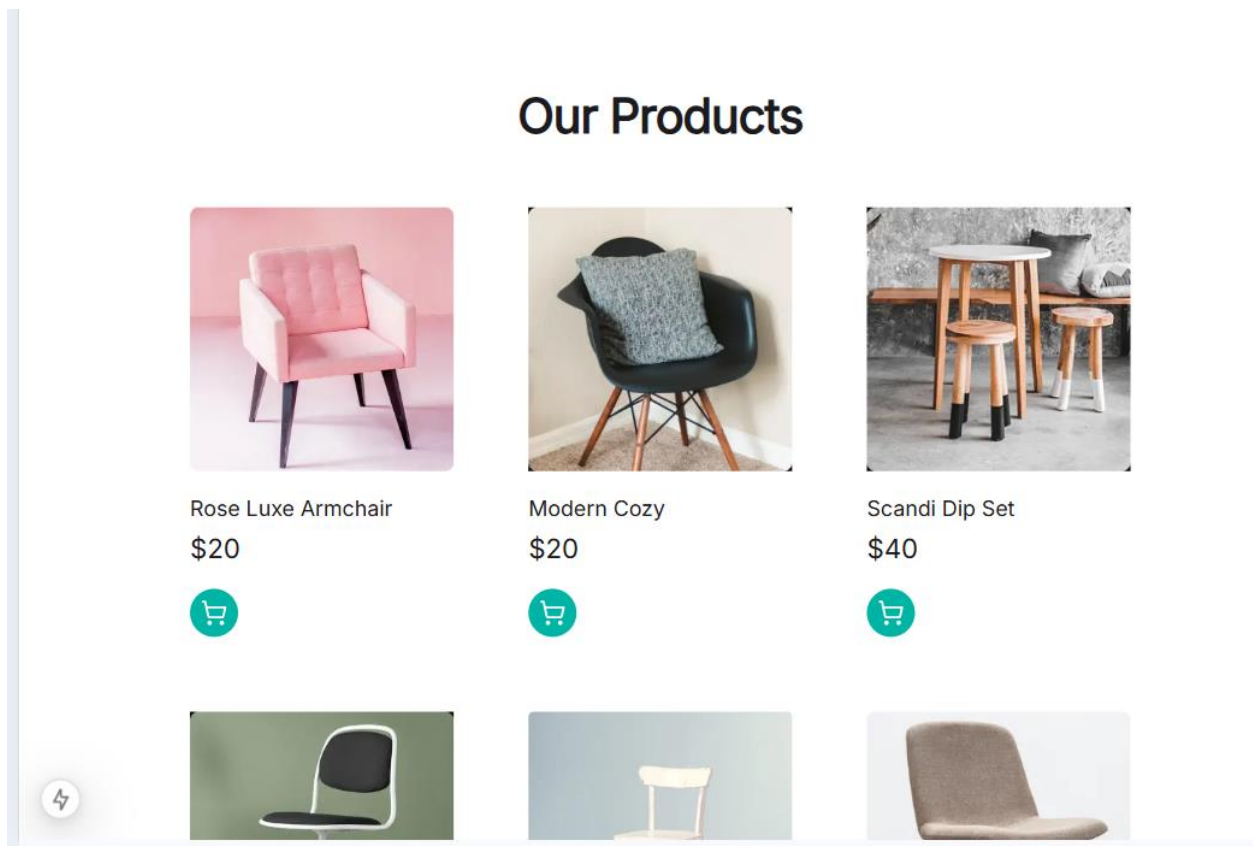
Sales Chart



Promotion and Discount snippet.

Apply Discount Code

Product page snippet.



Final checklist.

Frontend Component Development: ✓ Completed

Styling and Responsiveness: ✓ Completed

Code Quality: ✓ Completed

Documentation and Submission: ✓ Completed

Final Review: ✓ Completed