Day 4 - Dynamic Frontend Components – General E-Commerce

Introduction

The purpose of Day 4 is to build dynamic frontend components for a marketplace, focusing on displaying data fetched from a CMS or APIs. The components created should be modular, reusable, and scalable to ensure a responsive and efficient user experience. This document summarizes the steps taken to implement the dynamic components and outlines the challenges faced along with the solutions.

Key Learning Outcomes

1. Build dynamic frontend components to display data from Sanity CMS or APIs.
2. Implement reusable and modular components.
3. Understand and apply state management techniques.
4. Learn the importance of responsive design and UX/UI best practices.
5. Prepare for real-world client projects by replicating professional workflows.

Key Components Implemented

1. Product Listing Component: Built to dynamically display product data, including name, price, image, and stock status, in a grid layout.
2. Product Detail Component: Created individual product detail pages using dynamic routing to show detailed product information.
3. Category Component: Displayed product categories dynamically and enabled filtering by selected categories.
4. Search Bar: Implemented functionality to filter products by name or tags.
5. Cart Component: Managed cart items and tracked the quantity and total price.
6. Pagination: Added pagination to break large product lists into manageable pages.

Steps Taken

1. Setup: The Next.js project was connected to the API, and data fetching was tested for product listings and categories.
2. Components Implementation:
   a) Product Listing: Designed a reusable ProductCard component and dynamically rendered product data using props.
   b) Dynamic Routing: Implemented dynamic routing for individual product detail pages to render data based on the product ID.

c) Category Filters: Created a CategoryFilter component and implemented a filter function based on category selection.

d) Search Bar: Integrated a search bar that allows users to filter products by name or tags.

e) Pagination: Implemented pagination to allow the listing of products to be broken into multiple pages.

Challenges Faced and Solutions

1. Fetching Dynamic Data: Initially, there were issues with fetching dynamic data correctly from the API. This was solved by carefully reviewing the API calls and ensuring the correct endpoints were used.
2. Dynamic Routing: Handling dynamic routes for product detail pages was tricky at first. The solution was to use Next.js' built-in dynamic routing features and pass product IDs in the URL to fetch data.
3. Filtering and Search: Handling both category-based filtering and search functionality required managing multiple states. This was resolved by using React's `useState` for local state and ensuring that the search bar and filter components were synchronized.

Best Practices Followed

1. Reusable Components: Components like ProductCard, CategoryFilter, and SearchBar were designed to be reusable across pages, ensuring modularity and scalability.
2. State Management: React's `useState` and `useContext` were utilized for managing local and global states effectively.
3. Responsive Design: Tailwind CSS was used to ensure a responsive design across different screen sizes. Media queries were implemented for mobile-first layouts.
4. Performance Optimization: Implemented lazy loading for images and pagination to improve performance for large datasets.

Expected Output

By the end of Day 4, the following components were successfully implemented and tested:
1. A fully functional product listing page displaying dynamic data.
2. Individual product detail pages with dynamic routing.
3. Working category filters and search bar functionality.
4. Pagination and related products on the product detail pages.
5. Components styled to ensure responsiveness and professional appearance.

Conclusion

This document outlines the steps taken to build dynamic frontend components for a marketplace. The components were designed to be modular and reusable, ensuring scalability and performance. The development focused on applying best practices in state management, responsive design, and UI/UX to create a functional, user-friendly, and professional marketplace.