Day 4 - Dynamic Frontend Components -General E-Commerce

Project Overview

This is an e-commerce platform based on Next.js with some exciting functionalities:

- Search functionality by name and tags.
- Add to cart and remove from cart functionality.
- Fetching the product data based on Sanity CMS.
- Implemented a product detail page with dynamic routing.
- Handled state management using Context API.

Steps Used to Create and Integrate Components

1. Search Bar Functionality

Implementation:

- Designed a search bar component that enabled the user to find a product based on its name or tags.
- Applied `useState` for updating the input value and for real-time filtering.

Implementation:

- Integrate the search bar with the fetched list of products from Sanity CMS.
- Ensure that every change in the search bar automatically refreshes the shown list of products.

2. Add to Cart and Remove from Cart

Implementation:

- Created the `addToCart` and `removeFromCart` functions using the Context API.
- Stored cart items in the context state for access across the application.
- Managed updates to the cart using `useReducer` for better state management.

Integration:

- Added "Add to Cart" and "Remove from Cart" buttons on product cards and detail pages.
- Updated the UI to reflect changes in the cart state immediately.

3. Rendering Data from Sanity CMS

Setup:

- Set up the Sanity client to fetch data.
- Wrote GROQ queries to fetch data for products like names, tags, images, and descriptions.

Rendering:

- Rendered the fetched data in a responsive product grid.
- Used loading and error states.

4. Product Details Page with Dynamic Routing

Setup:

- Created individual product detail pages using Next.js dynamic routes (`[slug].tsx`).
- Retrieved specific product data from Sanity CMS based on the dynamic `slug` parameter.

Integration:

- Added links from product cards to their detail pages.
- Implemented the detail page to show all product information, including images, descriptions, and an "Add to Cart" button.

5. Context API for State Management

Implementation:

- Set up a Context API provider to manage global state for the cart and user interactions.
- Defined actions like `ADD_TO_CART` and `REMOVE_FROM_CART` in the reducer function.

Integration:

- Wrapped the application in the Context Provider to ensure global access.
- Consumed context in components to manage cart actions and reflect state changes.

Challenges Faced and Solutions Implemented

1. Problem: Sanity CMS Integration with Dynamic Routes

Problem: Data was being fetched for dynamic routes, and it was slow and undefined.
Solution: Using Next.js `getStaticProps` and `getStaticPaths` to pre-render pages, fetching data at build time.

2. Problem: State Across Multiple Components

Problem: Props were being passed through multiple layers, making it cumbersome.
Solution: Implemented Context API to centralize state management, making state sharing and updates easy.

3. Challenge: Search Bar Performance

Problem: Filtering a large dataset caused performance issues.
Solution: Optimized filtering logic using debouncing to reduce frequent re-renders.

Conclusion

Key project features, like search functionality and cart management and product detail pages to be dynamic in nature, are implemented and successfully integrated on the fourth day of the hackathon. Although this is the end of Day 4, this is where issues did not pop out to stop smooth user experience since optimized solutions would be applied right away.

The project can now stand poised for further development and deployment.