Mastering Dynamic Frontend Components

Elevate Your Marketplace Experience

Day 4 Report

Presented by: Abid Ali

Overview

The primary focus of today's session was building and integrating dynamic frontend components for the Comforty marketplace. This included developing:

- A fully functional product listing page.
- Individual product detail pages.
- Advanced category filters.
- Reviews and ratings.
- Add-to-cart and wish list functionalities.
- Authentication using Clerk.

Below is a detailed breakdown of the work completed.

Functional Deliverables

1. Product Listing Page with Dynamic Data

- **Description:** The product listing page dynamically fetches and displays product data from Sanity CMS or APIs. Each product is displayed in a card format showing its image, name, and price.
- **Implementation:**
  - Data fetching using APIs.
  - Responsive grid layout for product cards.
  - Dynamic rendering of product information.

2. Individual Product Detail Pages

- **Description:** Each product has a dedicated page that dynamically fetches and displays its details based on the product ID. Information such as name, description, price, and images is displayed.

- **Implementation:**
  - Dynamic routing using `[slug].tsx` in Next.js.
  - Accurate data rendering for individual products.

3. Advanced Category Filters

- **Description:** Users can refine product views by selecting categories. Filters dynamically update the product list based on the selected category.
- **Implementation:**
    - UI for seamless category selection.

4. Reviews and Ratings Component

- **Description:** Users can view and submit reviews for products. Average ratings and individual reviews are displayed dynamically. Users can also rate products on a scale of 1 to 5 and edit or delete their reviews.
- **Implementation:**
    - Review data storage and retrieval from Sanity CMS.
    - Editable and deletable reviews for authenticated users.

5. Add-to-Cart Functionality

- **Description:**
    - Users can add products to their cart directly from the product listing or detail pages.
    - A cart icon dynamically updates to show the number of items added.
    - Notifications for successful additions.
    - Quantity adjustment and item removal options.
- **Implementation:**
    - State management using React Context API.
    - State persistence via local storage.

6. Inventory Management

- **Description:** After an order is confirmed, the stock of the product is updated in Sanity CMS. The updated stock is displayed on the product detail page. Orders require user authentication.
- **Implementation:**
    - API integration to update stock levels.
    - Real-time stock reflection on product pages.

7. Authentication Using Clerk

- **Description:** Integrated Clerk for user authentication. Users can sign up, log in, and access protected routes seamlessly with pre-built UI components.
- **Implementation:**
    - User-friendly login/signup modals.
    - Secured access to restricted features.

Challenges Faced and Solutions

1. Dynamic Routing Issue

- **Challenge:**
  - Fetching and rendering product-specific details on individual product pages was initially challenging due to route parameter management.
- **Solution:**
  - Implemented dynamic routing using `[slug].tsx` in Next.js.
  - Ensured accurate and efficient data fetching for each product.

2. Performance Optimization

- **Challenge:**
  - Heavy components impacted initial load times.
- **Solution:**
  - Used dynamic imports for components like `ReviewsComponent` and `Checkout` to improve performance.

3. Responsive Design

- **Challenge:**
  - Ensuring a seamless user experience across devices.
- **Solution:**
  - Implemented responsive styling for all components.

4. Error Handling

- **Challenge:**
  - Handling API failures gracefully.
- **Solution:**
  - Added error boundaries and fallback UI to display user-friendly error messages.

Best Practices Followed

1. **Modular Components:**
   - Created reusable components like c`ardProduct` & `ReviewsComponent` to enhance maintainability and scalability.
2. **Responsive Design:**
   - Styled components to ensure compatibility across desktops, tablets, and mobile devices.
3. **Error Boundaries:**
   - Included fallback UI for API failures and unexpected errors.
4. **Code Splitting:**
   - Implemented dynamic imports for performance optimization.

Conclusion

The day's tasks were successfully completed, and all deliverables are fully functional. The project now features:

- Dynamic product listings with data fetched from Sanity CMS.
- Individual product detail pages with dynamic routing.
- Reviews and ratings with edit/delete functionality.
- Add-to-cart and wishlist features with state persistence.
- Inventory management to reflect updated stock after order confirmation.
- Secure authentication using Clerk.

The project's clean and modular design ensures scalability and maintainability, delivering a professional user experience.

Thank You!