Documentation for Day 4: Building Dynamic Frontend Components for Your Marketplace

Summary of Steps Taken

1. Setting Up the Project
- Framework: Used Next.js as the primary framework for its support of dynamic routing and server-side rendering.
- Styling: Tailwind CSS was integrated to ensure responsive and professional UI designs.
- Data Source: Configured Sanity CMS for fetching and managing dynamic data. Tested API connectivity and ensured the dataset was aligned with marketplace requirements.

2. Component Development
- Product Listing Component:
  - Rendered product data dynamically using a grid layout.
  - Implemented cards to display product details such as name, price, and stock status.
- Product Detail Component:
  - Created individual product pages with dynamic routing.
  - Included fields like description, price, and available sizes/colors.
- Category Component:
  - Displayed dynamic categories fetched from Sanity CMS.
  - Enabled filtering functionality for products by category.
- Search Bar:
  - Implemented a search bar to filter products by name or tags dynamically.

- Cart Component:
  - Added functionality to display cart items, quantities, and total price.
  - Used React Context API for state management.
- Wishlist Component:
  - Enabled users to save products using local storage.
- Checkout Flow Component:
  - Developed a multi-step form for billing, shipping, and payment details (mock implementation).
- Footer and Header Components:
  - Added consistent navigation and branding with responsive design.

3. Responsive Design
- Ensured mobile-first development with flexible grid layouts.
- Used Tailwind CSS utilities for styling and responsiveness.

4. State Management
- Used React Context API for global state management in components like Cart and Wishlist.
- Used useState for managing local component state.

5. Performance Optimization
- Implemented lazy loading for images to enhance loading speed.
- Used pagination to handle large datasets efficiently.

Challenges Faced

1. API Integration Issues

- Problem: Errors while fetching data from Sanity CMS.
- Solution: Verified API keys, corrected dataset configurations, and used error logging to debug the issue.

2. Dynamic Routing Challenges
- Problem: Routing to individual product pages initially failed.
- Solution: Added [id] as a dynamic segment in the Next.js routes and ensured valid IDs were passed during navigation.

3. State Management Complexities
- Problem: Managing cart and wishlist state across components.
- Solution: Implemented Context API for centralized state management.

4. Styling Consistency
- Problem: Inconsistent styling across devices.
- Solution: Used Tailwind CSS and tested the layout on various screen sizes for adjustments.

Solutions Implemented

1. Component Reusability
- Developed modular components like ProductCard and CategoryFilter to maintain scalability.

2. Error Handling
- Added try-catch blocks in API calls to handle errors gracefully and provide user feedback through toast notifications.

3. Best Practices

- Followed frontend best practices, including:

- Modular component design.

- Responsive UI development.

- Clear separation of concerns.

Conclusion

The project successfully met the objectives of Day 4 by creating dynamic, responsive, and reusable components. The implementation replicated real-world practices, preparing for scalable and professional marketplace solutions.