Hackathon Day 4

Building dynamic and responsive Marketpalce components

“THE COZYNEST”

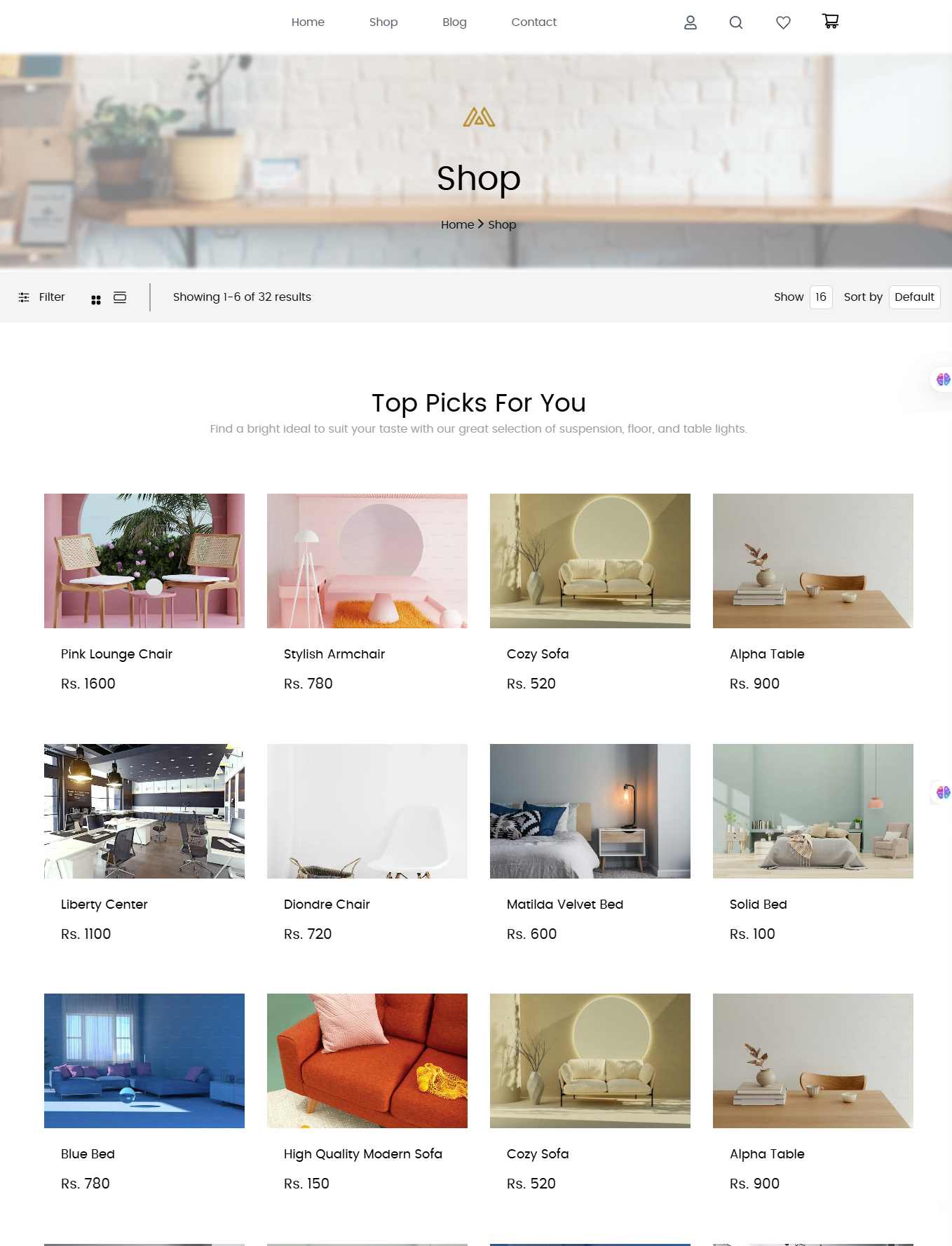
**Objective:** On Day 4, the focus is on designing and developing dynamic frontend components to display marketplace data fetched from Sanity CMS or APIs. This day emphasizes creating modular, reusable components and applying best practices for scalable and responsive web applications.

**Key Learning Outcomes:**

1. Build dynamic frontend components to display data from Sanity CMS or APIs.
2. Implement reusable and modular components.
3. Apply state management techniques.
4. Simulate real-world workflows for professional projects.

**Key Components to Build:**

1. **Product Listing Component:**
   * Render product data dynamically in a grid layout.
   * Include fields like Product Name, Price, Image, and Stock Status.



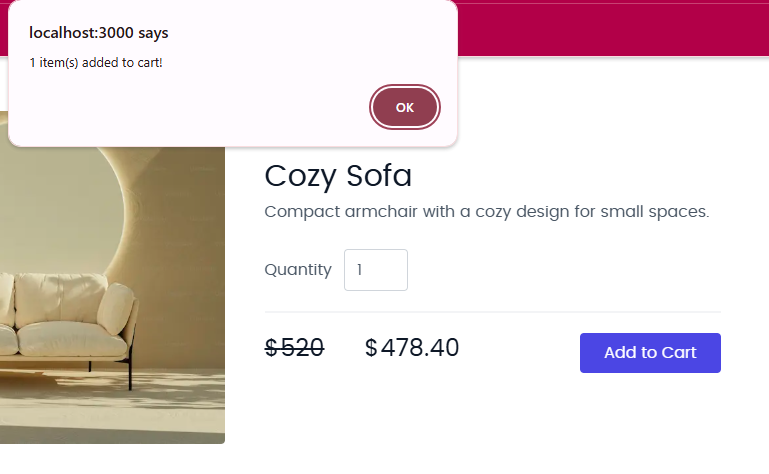
 **Product Detail Component:**

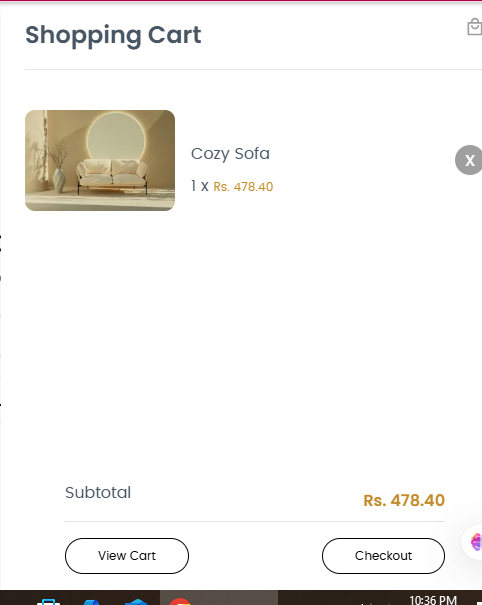
* Create individual product detail pages using Next.js dynamic routing.
* Display Product Description, Price, and Available Sizes/Colors.

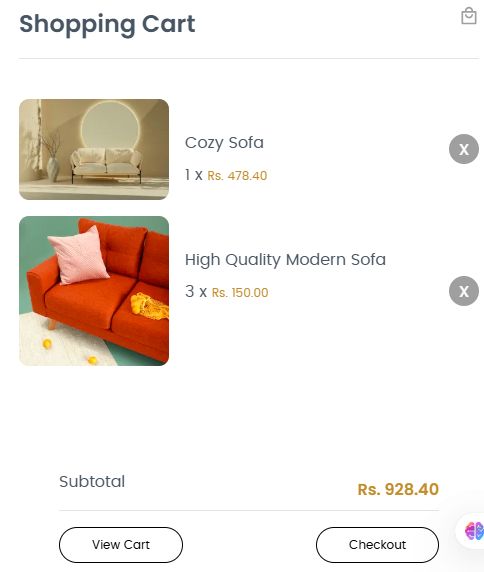


 **Cart Component:**

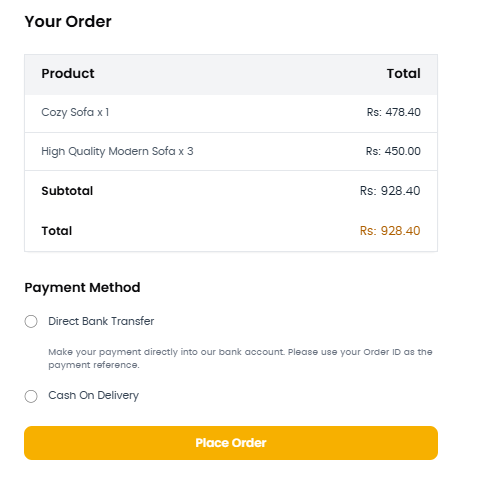
* Display added items, quantity, and total price.
* Use state management for cart functionality

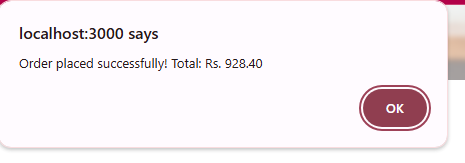






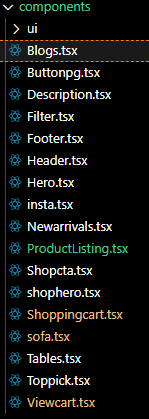
1. **Checkout Flow Component:**
   * Design a multi-step form for billing/shipping addresses and payment details (mock implementation).





**Frontend Best Practices:**

1. **Reusable Components:**
   * Build modular components like ProductCard and CategoryFilter.
   * Pass data via props for flexibility.
2. **State Management:**
   * Use React state or context for data management.
3. **Styling:**
   * Use modern libraries like Tailwind CSS or styled-components.
   * Ensure responsiveness for all device sizes.
4. **Performance Optimization:**
   * Implement lazy loading and pagination.
   * Optimize images and avoid unnecessary re-renders.



**Steps for Implementation:**

1. **Setup:**
   * Ensure Next.js project is connected to Sanity CMS or API.
   * Test data fetching to confirm availability.
2. **Component Development:**
   * Create the listed components as modular, reusable elements.
   * Use appropriate libraries and frameworks for styling and state management.