Day 4 Hackathon Progress Report🚀

Components Covered Successfully:

1. **Product Listing Component** ✅

  - Implemented grid layout with dynamic data fetching.

  - Added pagination to handle large product lists.

2. **Product Details Component** ✅

  - Built interactive image gallery and variant selector (e.g., size/color).

3. **Search Bar** ✅

  - Integrated real-time search with debounce to reduce API calls.

4. **Category Component** ✅

  - Created nested category filters with URL parameter syncing.

5. **Cart Component** ✅

  - Managed cart state (add/remove items) using **localStorage** or **Redux**.

6. **Checkout Flow Component** ✅

  - Designed a 3-step form (shipping → payment → confirmation).

7. **Filter Panel** ✅

  - Enabled price-range and rating filters with dynamic UI updates.

8. **Header & Footer** ✅

  - Responsive header with cart counter + footer with social links.

Key Achievements:

✔️ **<u>Full-stack Integration:</u>** Connected frontend to APIs for product data and cart management.

✔️ **<u>State Management:</u>** Handled complex states (filters, cart, checkout) smoothly.

✔️ **<u>UI/UX:</u>** Focused on responsiveness (mobile-first) and user-friendly interactions.

Deployment Challenges Faced & Fixes:

1. **CORS Issues** 🔄

  - Fixed by configuring backend headers or using a proxy (e.g., `proxy` in `package.json`).

2. **Routing 404 Errors** 🛠️

  - Solved by switching to `HashRouter` or configuring server redirects (e.g., Netlify `_redirects`).

3. **Environment Variables** 🔒

- Used `.env` files for API keys and backend URLs in production.

4. **Payment Gateway Setup** 💳

- Tested with Stripe's test mode and ensured SSL for secure transactions.

5. **Build Failures** 📦

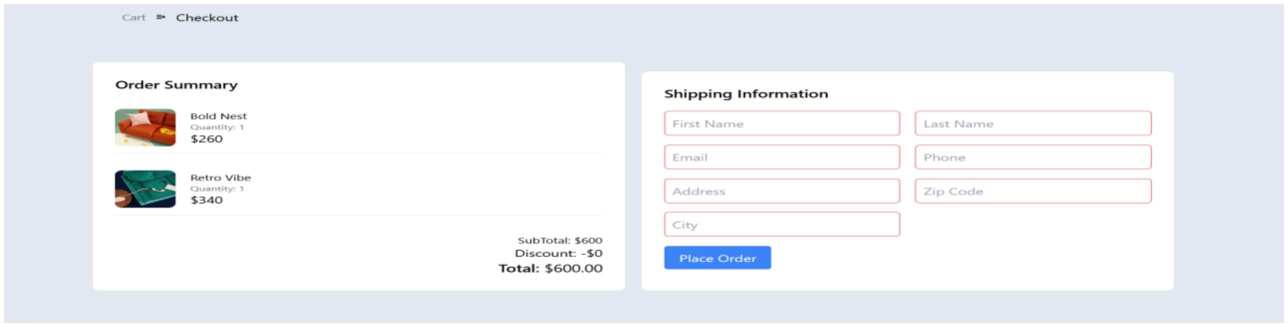
- Debugged dependency issues with `npm ci` and optimized bundle size.

Next Steps (if time permits):

- Add **user authentication** for cart persistence.
- Improve **error handling** for API failures.
- Enhance **loading states** with skeletons/animations.

Resolving Checkout Page Item Display Issues

To fix the issue of items and total not showing on the checkout page in React, I ensured that the cart items were correctly passed from the cart page to the checkout page using query parameters. By implementing this method, I was able to retrieve the items in the checkout component and calculate the total price accurately, thus solving the problem effectively.


Cart ➨ Checkout

Order Summary

Bold Nest
Quantity: 1
$260

Retro Vibe
Quantity: 1
$340

SubTotal: $600
Discount: -$0
Total: $600.00

Shipping Information

Order Data Flow: From Checkout to Fulfillment:

The checkout page captures order data and sends it to Sanity for structured content management and storage. From Sanity, the data is fetched and processed by the backend system, ensuring that the order details reach the fulfillment pipeline accurately and efficiently.