Day 4 - Dynamic Frontend Components - COMFORTY

1. Introduction

Day 4 of the hackathon focused on building dynamic frontend components to display and interact with the data imported into Sanity CMS on Day 3. The aim was to create a scalable, responsive, and user-friendly interface for the furniture marketplace.

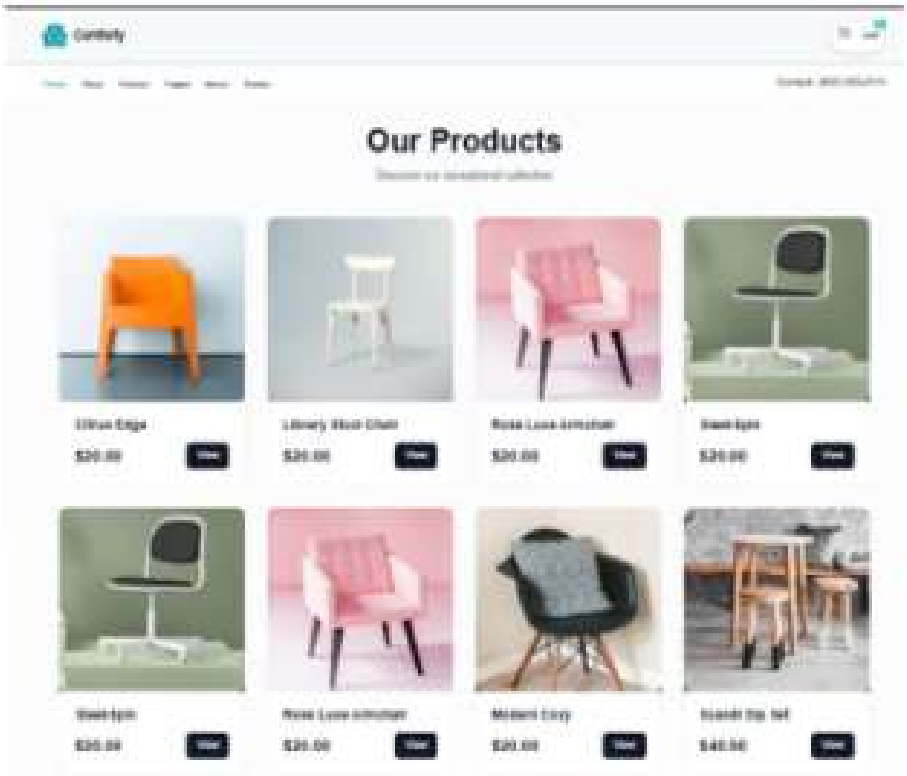
---

2. Key Components Implemented

Product Page

- **Purpose**: To display a dynamic list of products fetched from Sanity CMS.
- **Implementation**:
    - Fetched product data using Sanity's GROQ queries and displayed it in a grid layout.
    - Rendered product cards showing the name, price, image, and availability status.
    - Utilized reusable components for consistency and scalability.
- **Features**:
    - Responsive design for optimal viewing across devices.
    - Lazy loading for improved performance.

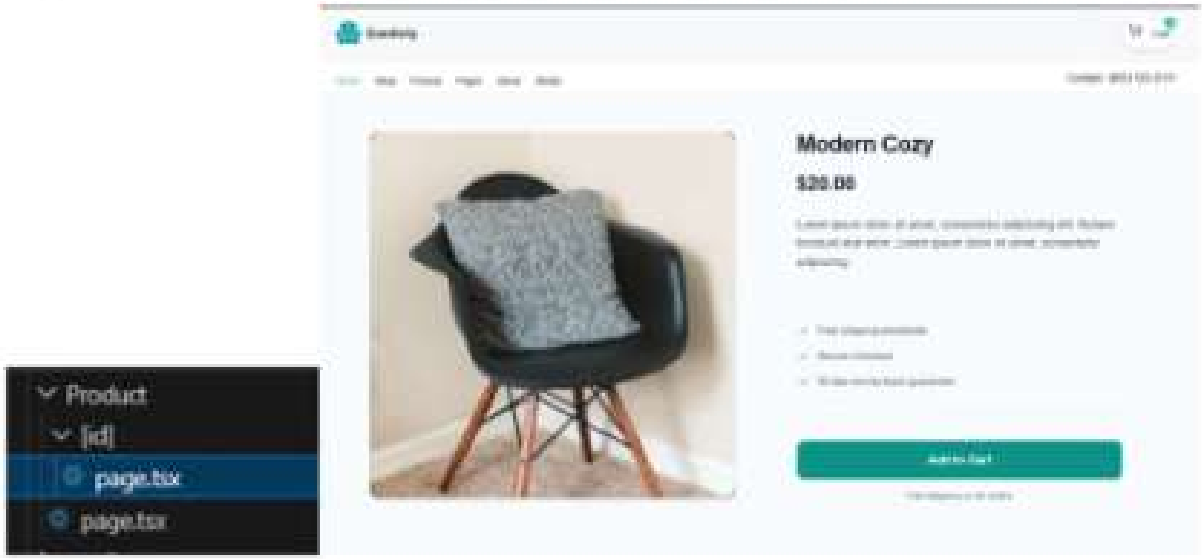
Snippets of Product Listing Page and its Code:



Product Detail Page

- **Purpose**: To provide detailed information about a specific product.
- **Implementation**:
  - Used Next.js dynamic routing (pages/product/[id].tsx) to create individual product pages.
  - Fetched and displayed detailed product data, including:
    - Name, price, description, images, and stock availability.
  - Integrated user-friendly navigation to return to the product listing.
- **Features**:
  - Dynamic URL-based navigation.
  - Clean and informative UI for an enhanced user experience.

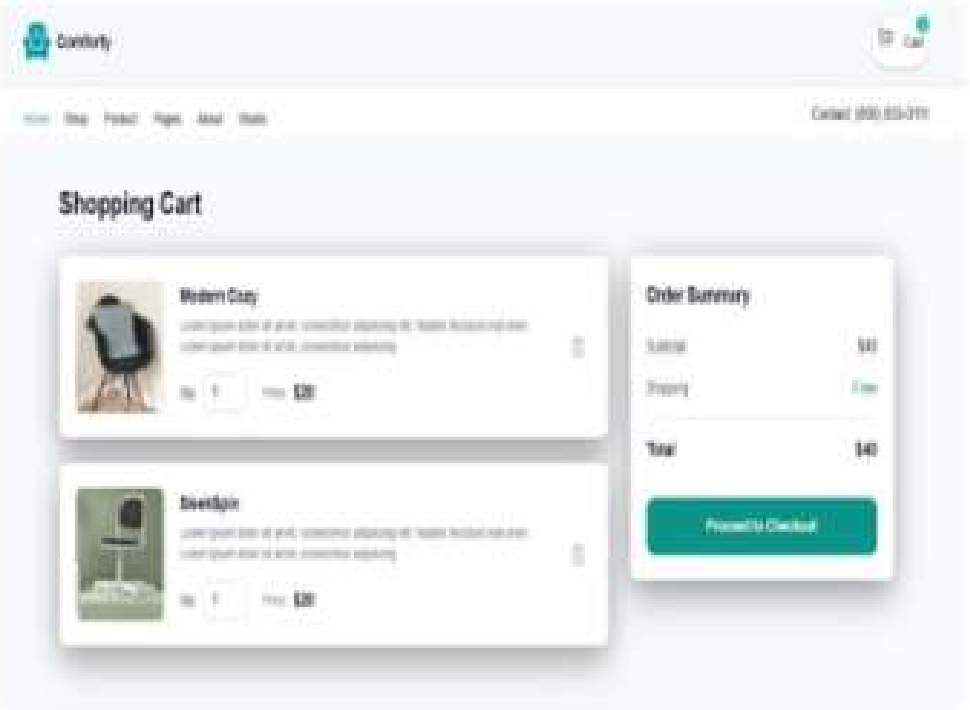
Snippets of Product Detailed Page and Dynamic Routing:



Cart Functionality

- **Purpose**: To allow users to add products to their cart and view selected items.
- **Implementation**:
    - Created a CartContext using React Context API for global state management.
    - Implemented "Add to Cart" functionality on the product detail page.
    - Cart features included:
        - Dynamic item count in the header.
        - View, update, and remove items from the cart.
    - Persisted cart state to local storage to retain data between sessions.

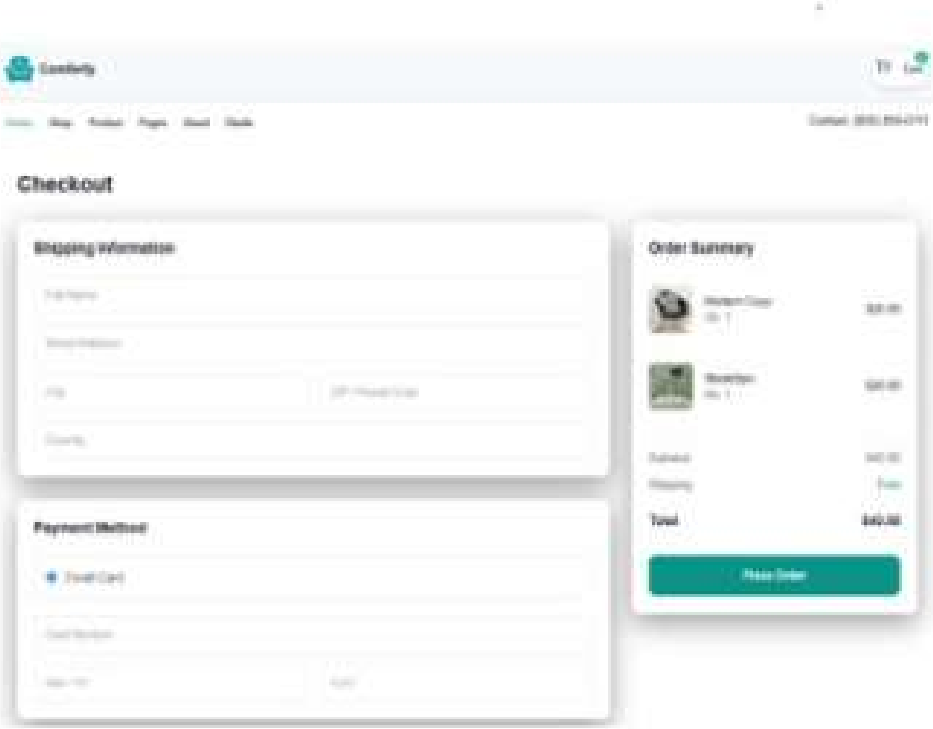
Snippets of Cart Page:



Checkout Page

- **Purpose**: To finalize the purchase process.
- **Implementation**:
    - Designed a multi-step checkout form to collect:
        - Billing and shipping details.
        - Payment information (mock implementation).
    - Displayed a summary of the cart with the total price.
- **Features**:
    - Clean and intuitive UI for user convenience.
    - Error handling for incomplete or invalid inputs.

Snippets of CheckOut Page:



3. Challenges and Solutions

- **API Integration with Sanity CMS**:
  - **Challenge**: Ensuring accurate data fetching and rendering for the product pages.
  - **Solution**: Verified the GROQ queries and implemented error handling to manage API response issues.
- **State Management**:
  - **Challenge**: Managing global cart state efficiently.
  - **Solution**: Utilized Context API for simplicity and local storage for persistence.
- **Dynamic Routing**:
  - **Challenge**: Dynamically generating pages for each product using Next.js.
  - **Solution**: Leveraged the getStaticPaths and getStaticProps functions to pre-render pages at build time.

---

4. Best Practices Followed

- Modular and reusable component design for scalability.
- Responsive design using Tailwind CSS to ensure compatibility across devices.
- Optimized data fetching with Sanity GROQ and React query hooks.
- Proper error handling for robust user interactions.

---

6. Project Link

https://giaic-market-place-e-commerce-hackathon.vercel.app/

---

7. Conclusion

Day 4 was focused on transforming static data into an interactive and functional user interface. By completing the product pages, cart functionality, and checkout flow, the project demonstrates a scalable approach to building an eCommerce platform.
