

Mastering Dynamic Frontend Components

*ELEVATE YOUR MARKETPLACE
EXPERIENCE ON DAY 4*

PRESENT BY: Harmain Bashir

MENTOR : SIR HAMZAH SYED

Overview:

Today's task focused on building and integrating dynamic frontend components for the Comforty marketplace. The goal was to create a fully functional product listing page, individual product detail pages, advanced category filters, and additional features like related products, reviews and ratings, and add-to-cart functionality. wishlist functionality, inventory management and Authentication was also integrated using Clerk. Below is a detailed report of the work completed.

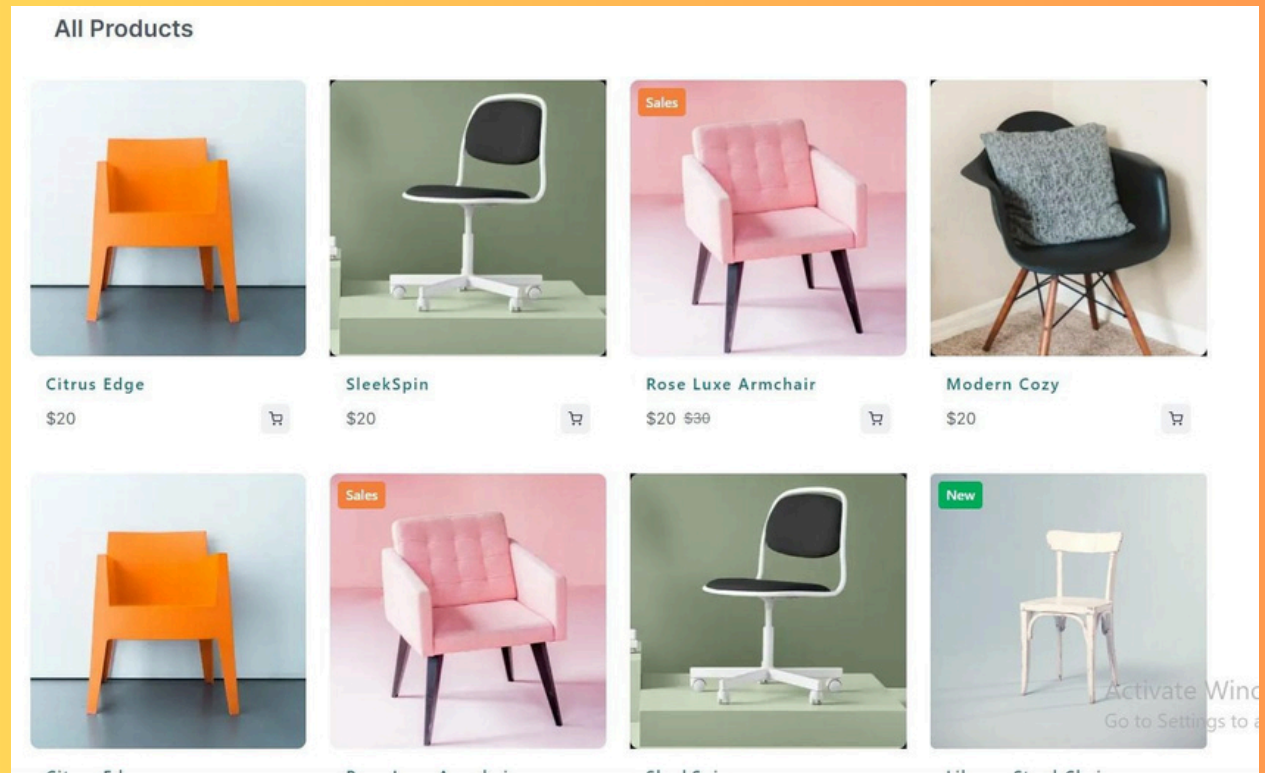
Functional Deliverables:

1. Product Listing Page with Dynamic Data:

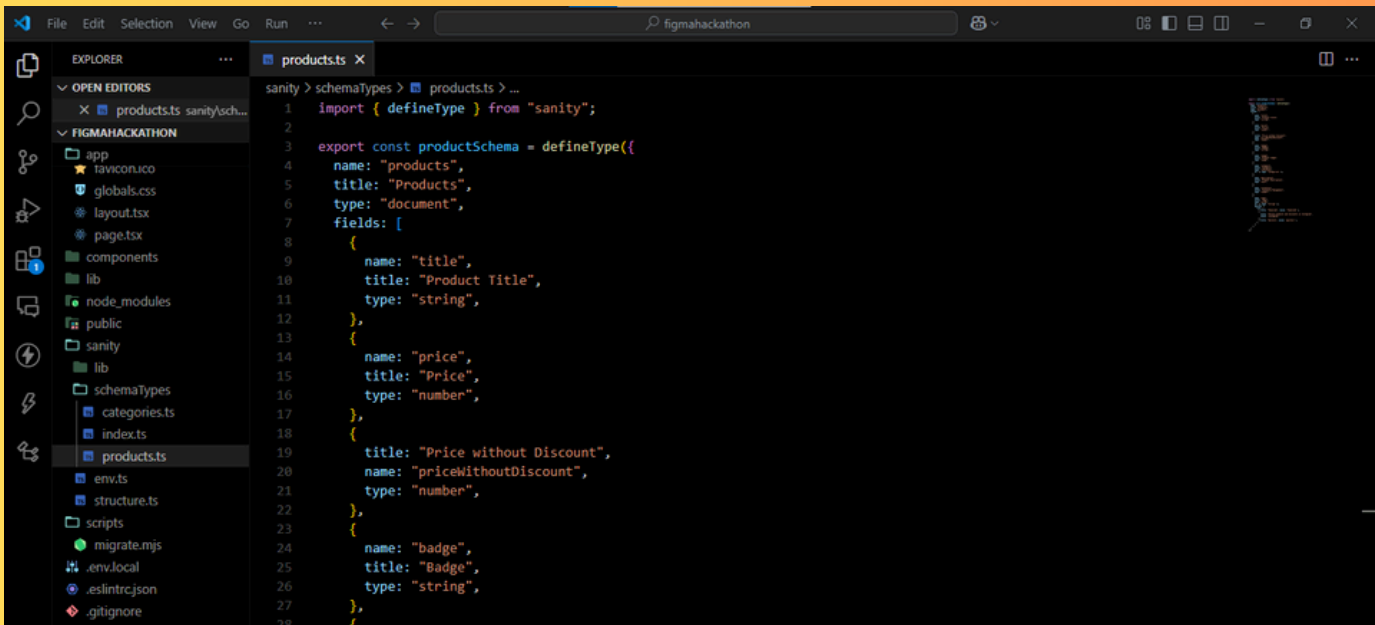
- *Description:*

The product listing page dynamically fetches and displays product data from Sanity CMS or APIs. Each product is displayed in a card format with its image, name, and price.

Caption: Product listing page displaying dynamic data.



Code Deliverables:

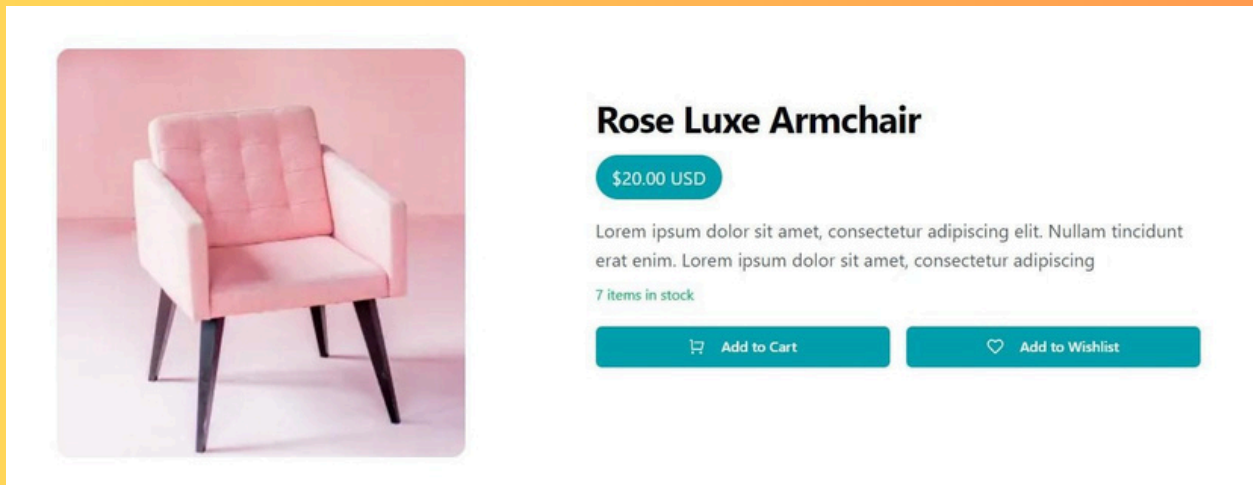


2. Individual Product Detail Pages:

- *Description:*

Implemented using dynamic routing ([id].ts). Each product detail page accurately renders data based on the product ID, including the product name, description, price, and image.

Caption: Individual product detail page with dynamic routing.



Code Deliverables:

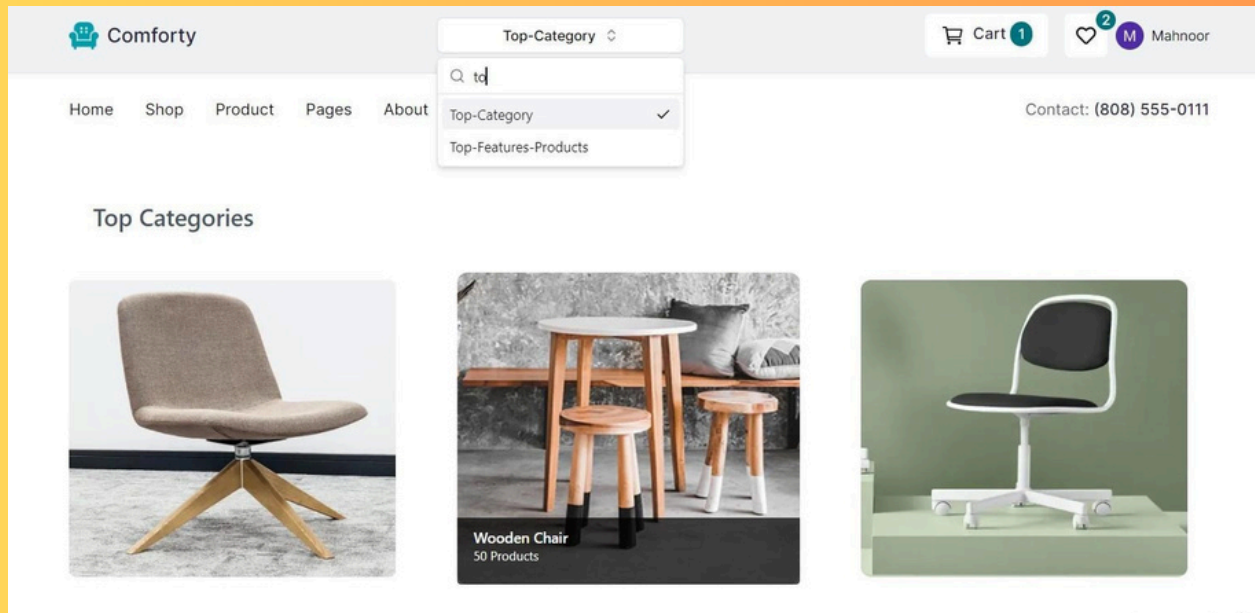
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'src', 'app', and 'admin'. The code editor shows the content of 'productDetailServer.tsx', which includes imports for 'client' and 'ProductDetailClient', and a function 'ProductDetailServer' that fetches data from an API and returns a 'ProductDetail' object.

3. Advanced Category Filters:

- *Description:*

Users can refine product views by selecting categories. The filters dynamically update the product list based on the selected category.

Caption: Category filters applied to refine product views



Code Deliverables:

```

24  const Products = [
42    {
43      value: "Our Products",
44      label: "Our-Products",
45      path: "/our-Products",
46    },
47  ];
48
49  export function ComboboxDemo() {
50    const [open, setOpen] = React.useState(false);
51    const [value, setValue] = React.useState("");
52
53    return (
54      <Popover open={open} onOpenChange={setOpen}>
55        <PopoverTrigger asChild>
56          <Button
57            variant="outline"
58            role="combobox"
59            aria-expanded={open}
60            className="w-[250px] px-6"
61          >
62            {value
63              ? Products.find((product) => product.value === value)?.Label
64              : "Select Categories..."}
65            <ChevronsUpDown className="opacity-50" />
66          </Button>
67          <PopoverContent className="w-[250px] p-0">
68            <Command>

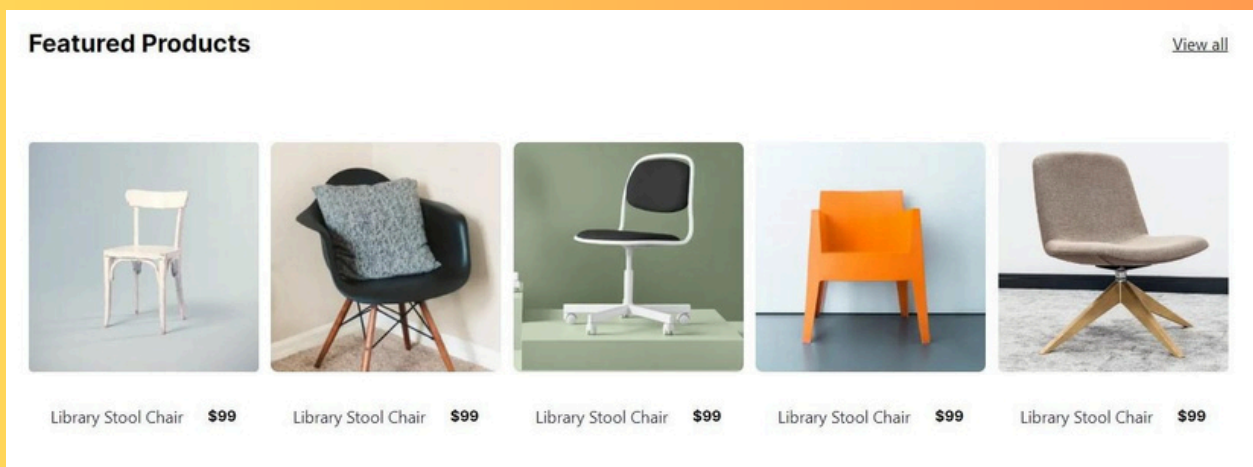
```

4. Related Products:

- *Description:*

Related products are displayed on individual product detail pages to enhance user engagement. These products are dynamically fetched based on the current product's category or tags.

Caption: Related products section on the product detail page.



5. Reviews and Ratings Component:

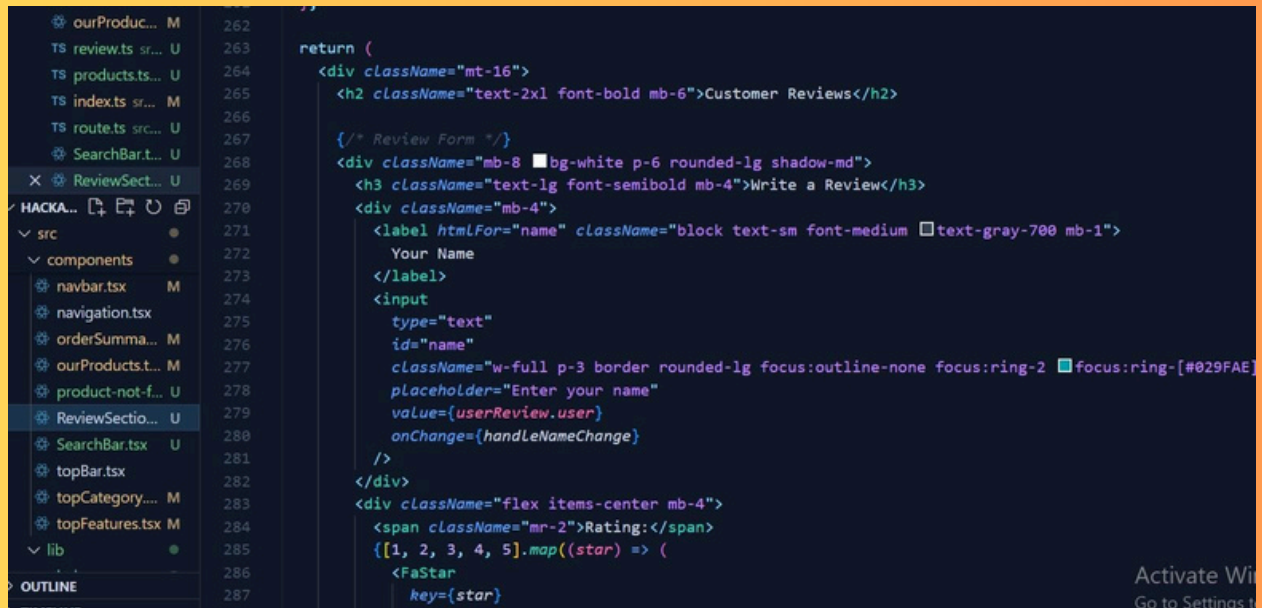
● Description:

Users can view and submit reviews for products. Average ratings and individual reviews are displayed dynamically. The component also allows users to rate products on a scale of 1 to 5. Reviews are stored in Sanity CMS and displayed for everyone on the product detail page. Users can easily edit or delete their reviews.

The screenshot displays the 'Customer Reviews' section of a product detail page. At the top, a green notification banner reads 'Thank you for your review!'. Below this is a 'Write a Review' form with a 'Your Name' input field, a 'Rating' section with five stars, and a text area for the review. A 'Submit Review' button is at the bottom of the form. Below the form, a sample review is shown with a 5-star rating, the name 'Mahnoor_Muhammad_Fareed', and the text 'This is the best product I have received. 🍷 ✨'. At the bottom of the review are 'Edit' and 'Delete' buttons. The date '1/20/2025' is displayed to the right of the review. In the bottom right corner, there is a watermark that says 'Activate Win Go to Settings to'.

Caption: Reviews and rating component on the product detail page.

Code Deliverables:



6. Add to Cart Functionality:

- *Description:*

Users can add products to their cart directly from the product listing or detail pages. The cart icon updates dynamically to reflect the number of items added. The cart state is managed using React's Context API and persisted in local storage. When a product is added to the cart, a notification is shown at the top of the page. Users can also adjust the quantity of products in the cart and remove items easily.

Caption: Add to cart button and cart icon with item count.

✓ Free shipping on all orders over \$50

Eng ▾ Faqs ⓘ Need Help

Comforty


Item successfully added to your cart!

Cart 1

2 M Mahnoor

Home Shop Product Pages About

Contact: (808) 555-0111



Rose Luxe Armchair

\$20.00 USD

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing

7 items in stock

Add to Cart


Add to Wishlist

Activate Windows

Home Shop Product Pages About

Contact: (808) 555-0111

Bag



Rose Luxe Armchair

♥

🗑

7 items in stock

-

1

+

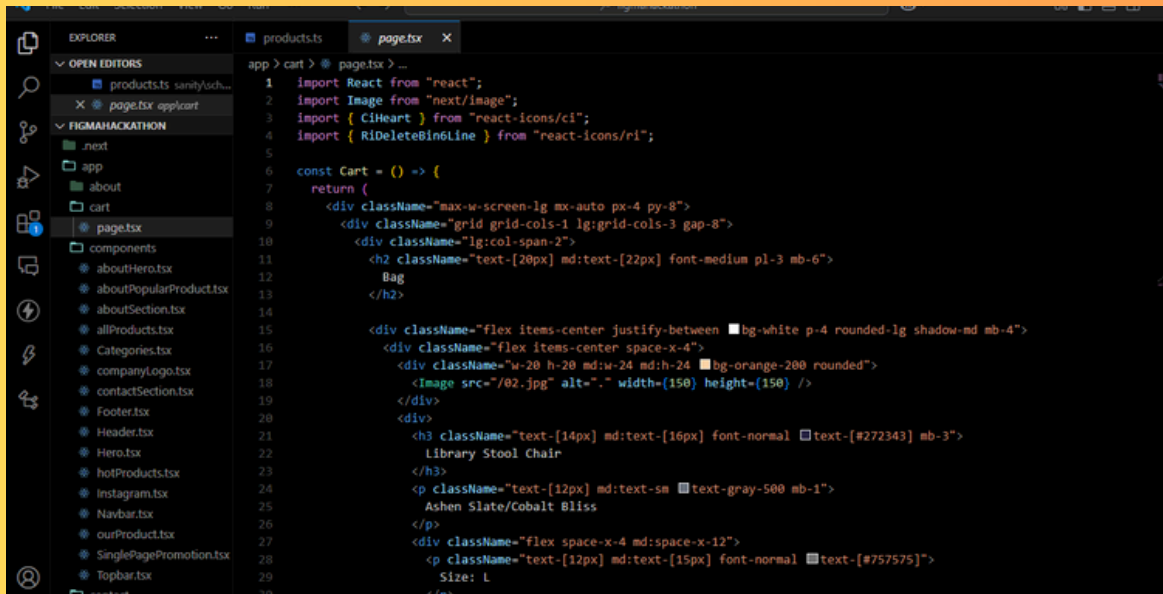
\$20.00

Summary

Subtotal	\$20.00
Estimated Delivery & Handling	Free
Total	\$20.00

Checkout

Code Deliverables:



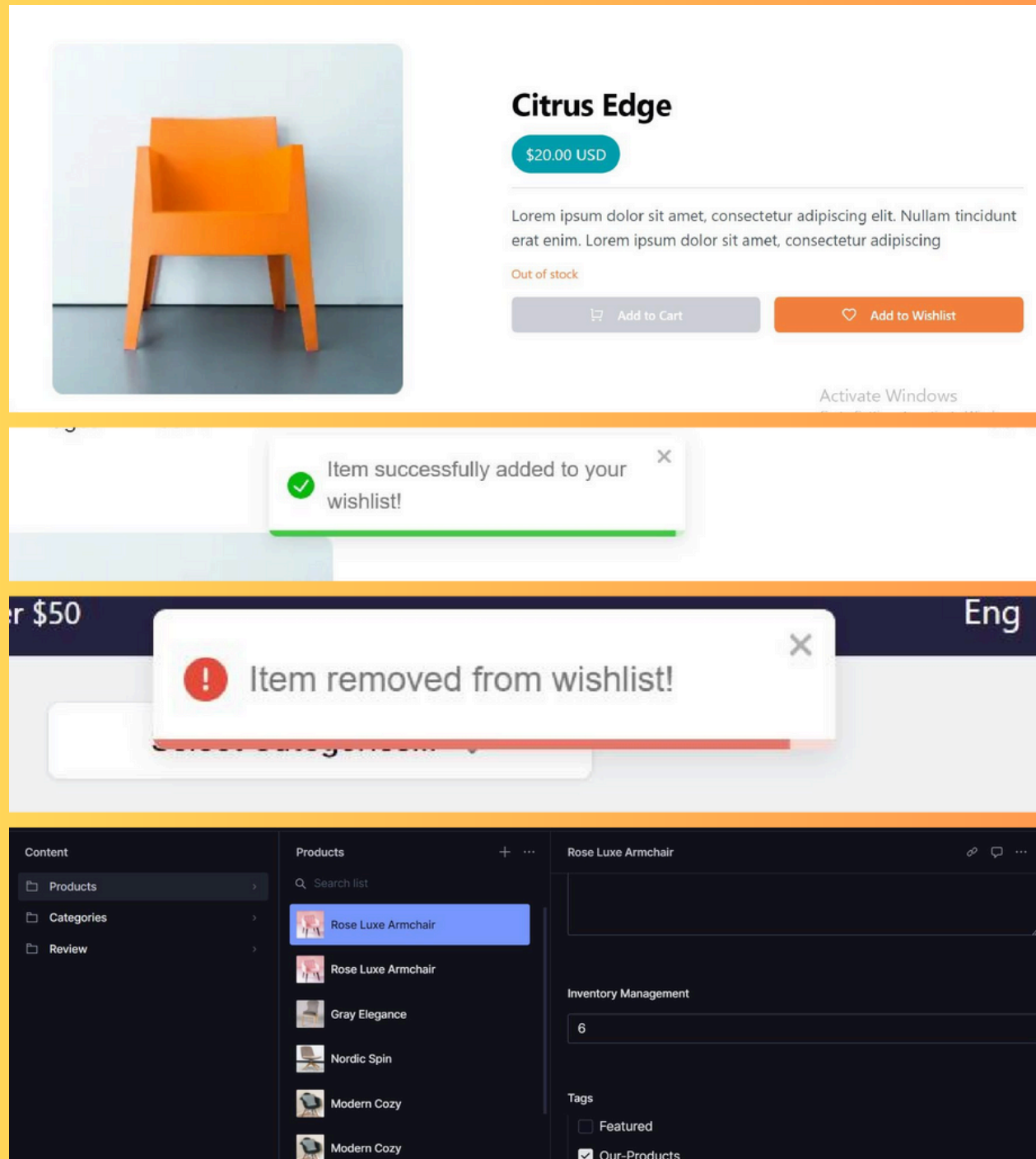
```
1 import React from "react";
2 import Image from "next/image";
3 import { CiHeart } from "react-icons/ci";
4 import { RiDeleteBin6Line } from "react-icons/ri";
5
6 const Cart = () => {
7   return (
8     <div className="max-w-screen-lg mx-auto px-4 py-8">
9       <div className="grid grid-cols-1 lg:grid-cols-3 gap-8">
10         <div className="lg:col-span-2">
11           <h2 className="text-[20px] md:text-[22px] font-medium pl-3 mb-6">
12             Bag
13           </h2>
14
15           <div className="flex items-center justify-between bg-white p-4 rounded-lg shadow-md mb-4">
16             <div className="flex items-center space-x-4">
17               <div className="w-20 h-20 md:w-24 md:h-24 bg-orange-200 rounded">
18                 <Image src="/02.jpg" alt="." width={150} height={150} />
19               </div>
20               <div>
21                 <h3 className="text-[14px] md:text-[16px] font-normal text-[#272343] mb-3">
22                   Library Stool Chair
23                 </h3>
24                 <p className="text-[12px] md:text-sm text-gray-500 mb-1">
25                   Ashen Slate/Cobalt Bliss
26                 </p>
27               </div>
28             <div className="flex space-x-4 md:space-x-12">
29               <p className="text-[12px] md:text-[15px] font-normal text-[#757575]">
30                 Size: L
31               </p>
32             </div>
33           </div>
34         </div>
35       </div>
36     </div>
37   );
38 }
```

7. Inventory Management:

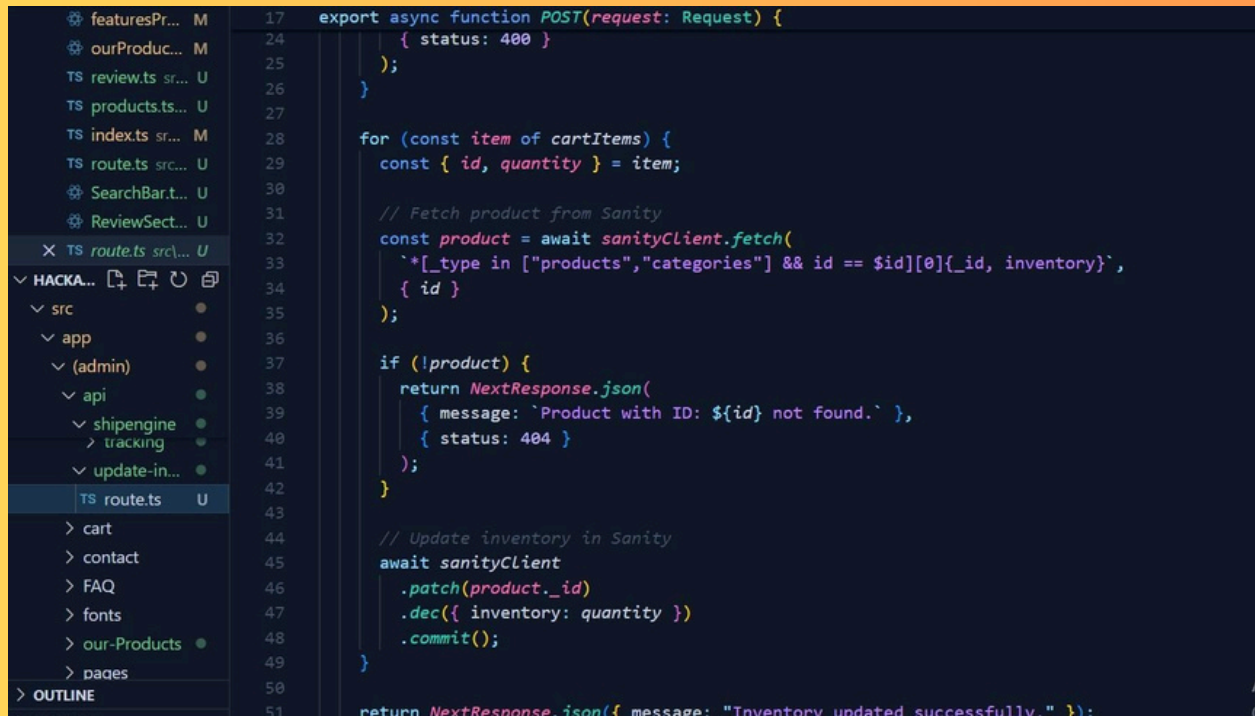
● Description:

After confirming an order, the stock of the product is decreased in Sanity CMS, and the updated stock is reflected on the product detail page. Users cannot confirm orders unless they are authorized (logged in).

Caption: Updated stock after order confirmation.



Code Deliverables:



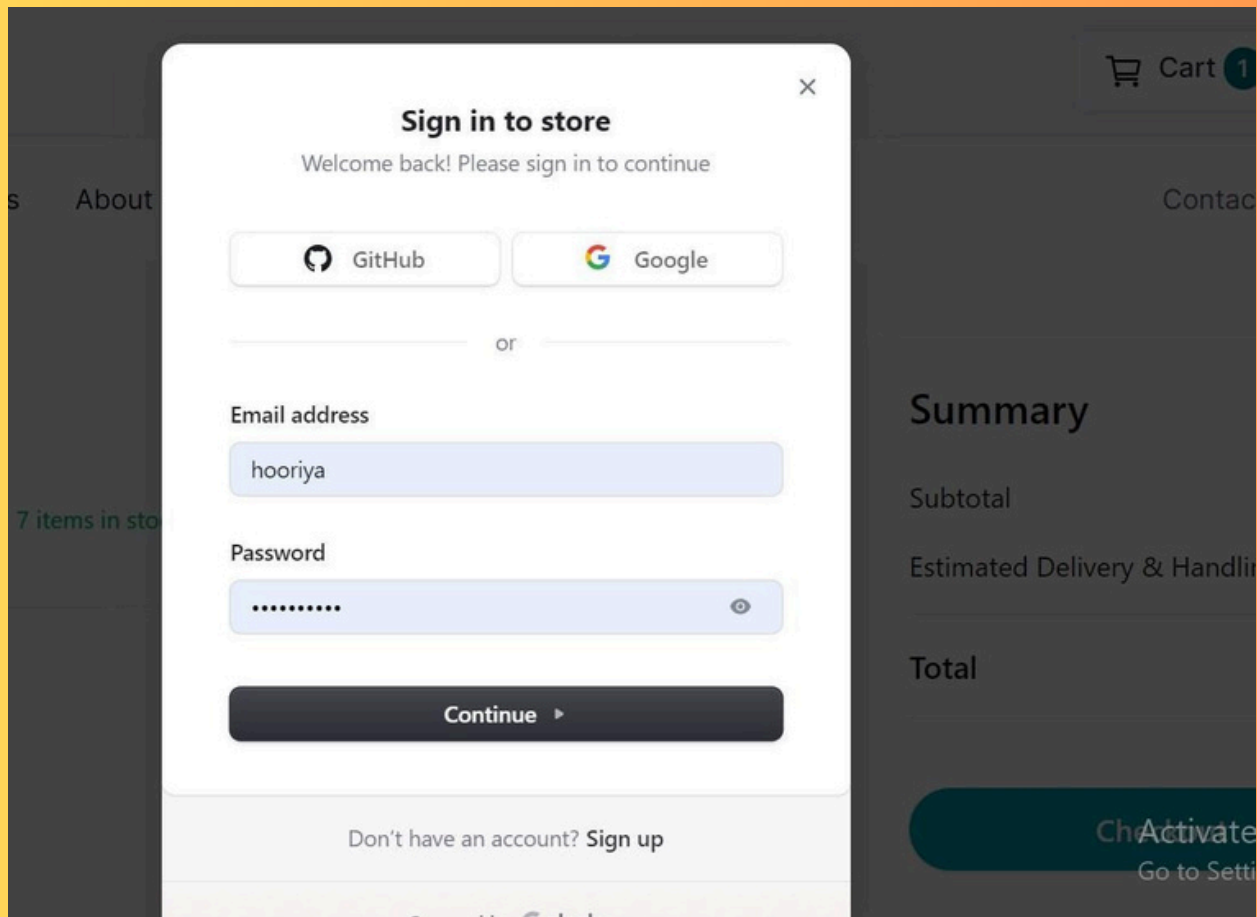
```
17 export async function POST(request: Request) {
24   { status: 400 }
25   );
26 }
27
28 for (const item of cartItems) {
29   const { id, quantity } = item;
30
31   // Fetch product from Sanity
32   const product = await sanityClient.fetch(
33     `*[_type in ["products","categories"] && id == ${id}][_id, inventory]`,
34     { id }
35   );
36
37   if (!product) {
38     return NextResponse.json(
39       { message: `Product with ID: ${id} not found.` },
40       { status: 404 }
41     );
42   }
43
44   // Update inventory in Sanity
45   await sanityClient
46     .patch(product._id)
47     .dec({ inventory: quantity })
48     .commit();
49 }
50
51 return NextResponse.json({ message: "Inventory updated successfully." });
```

8. Authentication using Clerk:

● Description:

Integrated Clerk for user authentication. Users can sign up, log in, and access protected routes. Clerk provides a seamless authentication experience with pre-built UI components.

Caption: Clerk authentication modal for user login/signup



Code Deliverables:

Challenges Faced

Dynamic Routing

Issue: Initially, fetching data for dynamic routes posed challenges, especially when trying to render product-specific details on individual product pages. The main issue was ensuring that the correct product data was fetched and displayed based on the dynamic route parameters (e.g., product ID).

Solution: After exploring various approaches, the issue was resolved by implementing dynamic routing in Next.js. By structuring the product detail pages using the `[id].ts` file naming convention, the application dynamically fetches and renders the correct product data based on the route parameters. This approach ensures that each product detail page is unique and displays accurate information without requiring hardcoded routes.

Best Practices Followed:

1. Modular Components:

Description:

Created reusable components like `ProductCard`, `SearchBar`, and `ReviewsComponent` to ensure scalability and maintainability.

Benefit: This approach reduces code duplication and makes it easier to update or extend functionality in the future.

2. Responsive Design:

Description:

Ensured all components are styled to be responsive across devices, including desktops, tablets, and mobile phones.

***Benefit:** Provides a consistent and professional user experience regardless of the device used.*

3. Code Splitting:

Description:

Used dynamic imports for heavy components to improve performance. For example, the ReviewsComponent and Checkout components are loaded only when needed.

***Benefit:** Reduces the initial load time of the application and improves overall performance.*

4. Error Handling:

Description:

Added error boundaries and fallback UI for API failures and unexpected errors. For example, if the product data fails to load, a fallback message is displayed to the user.

***Benefit:** Enhances user experience by gracefully handling errors and providing meaningful feedback.*

Conclusion:

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

- *Dynamic product listings with data fetched from Sanity CMS.*
- *Individual product detail pages using dynamic routing.*
- *Advanced category filters and a real-time search bar for seamless product discovery.*
- *Reviews and ratings with edit and delete functionality for authenticated users.*
- *Add-to-cart and wishlist features with state persistence in local storage.*
- *Inventory management to update product stock after order confirmation.*
- *Authentication using Clerk to ensure secure access to protected routes.*

The codebase is well-organized, scalable, and follows best practices, making it easy to maintain and expand in the future. By addressing challenges and focusing on clean, modular design, the project delivers a smooth and professional user experience.

Thank You!