PRESENT BY: Hooriya M.Fareed

MENTOR : SIR HAMZAH SYED

Overview:

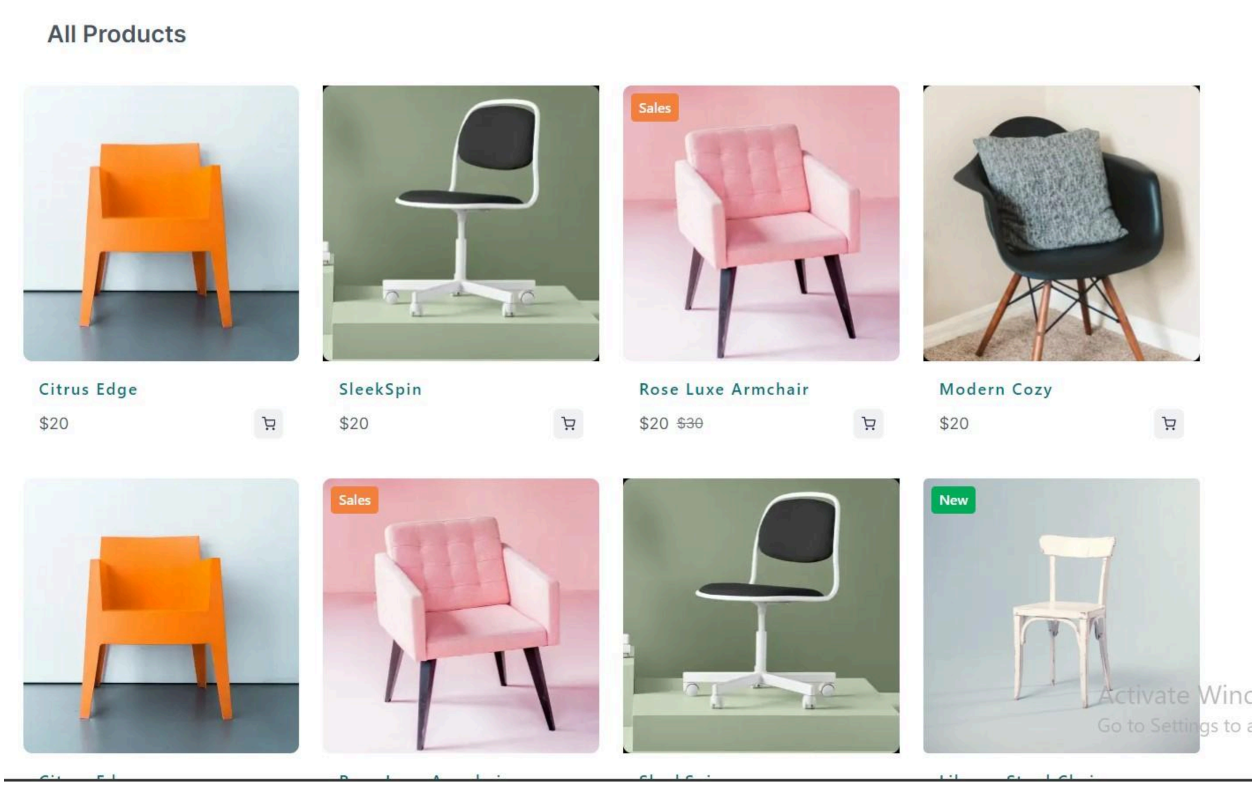
Today's task focused on building and integrating dynamic frontend components for the Comforty marketplace. The goal was to create a fully functional product listing page, individual product detail pages, advanced category filters, and additional features like related products, reviews and ratings, and 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.*



All Products

Citrus Edge
$20

SleekSpin
$20

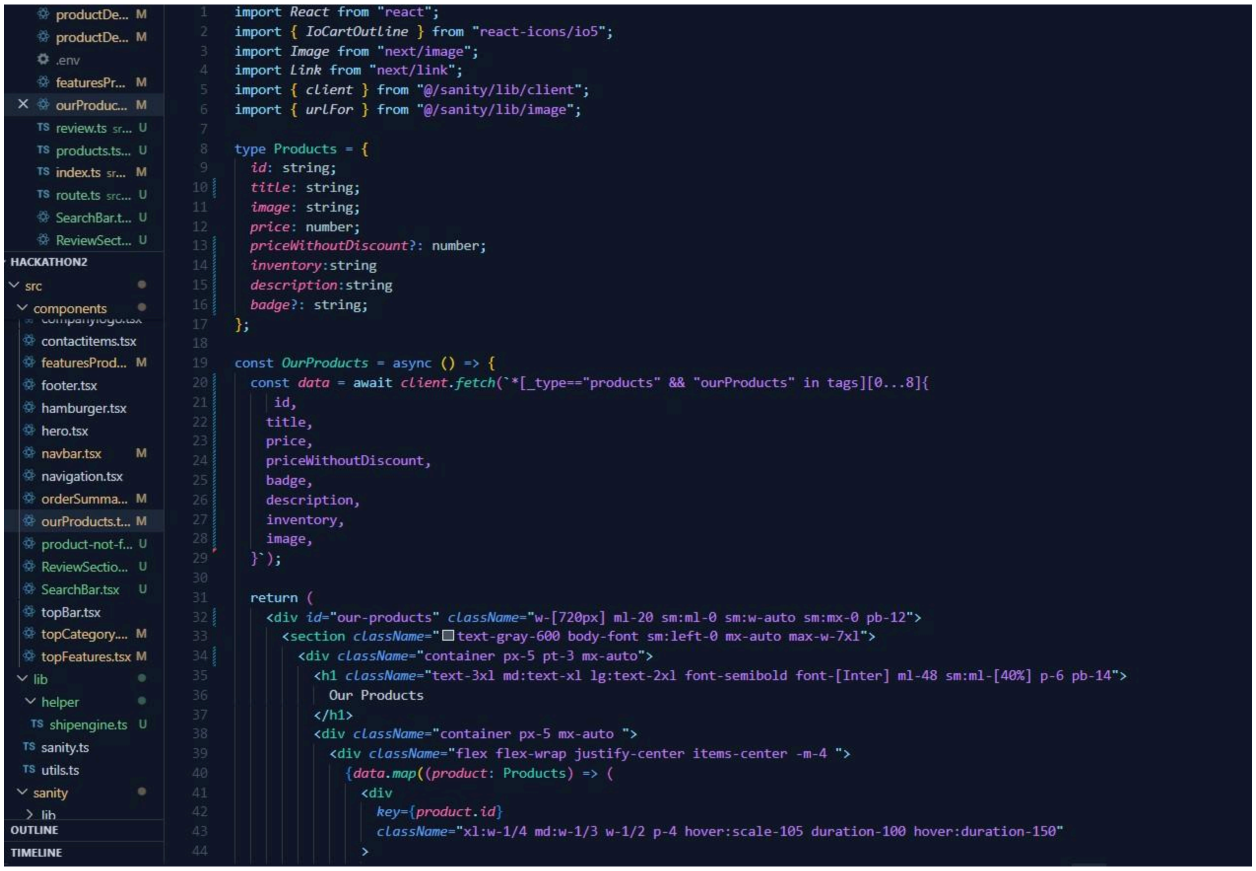
Sales
Rose Luxe Armchair
$20 ~~$30~~

Modern Cozy
$20

Sales

New

Code Deliverables:

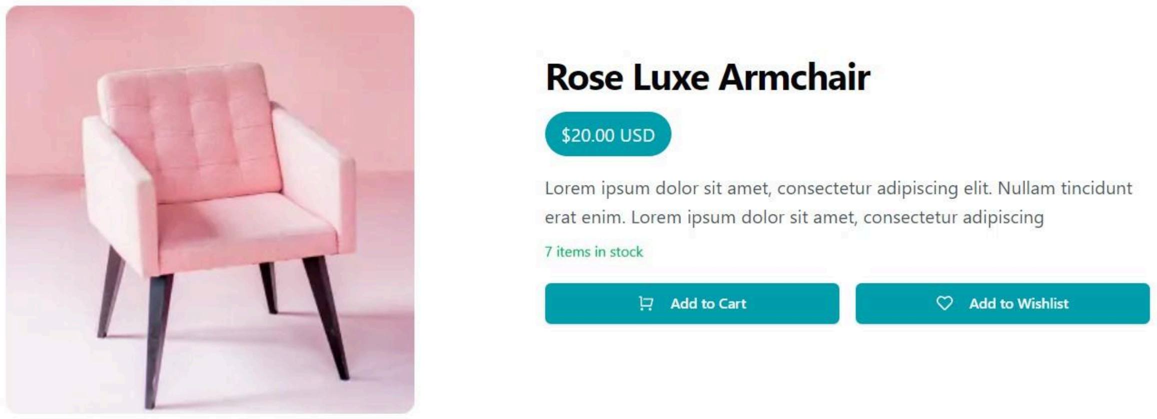


```
import React from "react";
import { IoCartOutline } from "react-icons/io5";
import Image from "next/image";
import Link from "next/link";
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";

type Products = {
  id: string;
  title: string;
  image: string;
  price: number;
  priceWithoutDiscount?: number;
  inventory:string
  description:string
  badge?: string;
};

const OurProducts = async () => {
  const data = await client.fetch(`*[_type=="products" && "ourProducts" in tags][0...8]{
     id,
    title,
    price,
    priceWithoutDiscount,
    badge,
    description,
    inventory,
    image,
  }`);

  return (
    <div id="our-products" className="w-[720px] ml-20 sm:ml-0 sm:w-auto sm:mx-0 pb-12">
      <section className="text-gray-600 body-font sm:left-0 mx-auto max-w-7xl">
        <div className="container px-5 pt-3 mx-auto">
          <h1 className="text-3xl md:text-xl lg:text-2xl font-semibold font-[Inter] ml-48 sm:ml-[40%] p-6 pb-14">
            Our Products
          </h1>
          <div className="container px-5 mx-auto ">
            <div className="flex flex-wrap justify-center items-center -m-4 ">
              {data.map((product: Products) => (
                <div
                  key={product.id}
                  className="xl:w-1/4 md:w-1/3 w-1/2 p-4 hover:scale-105 duration-100 hover:duration-150"
                >
```

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.*



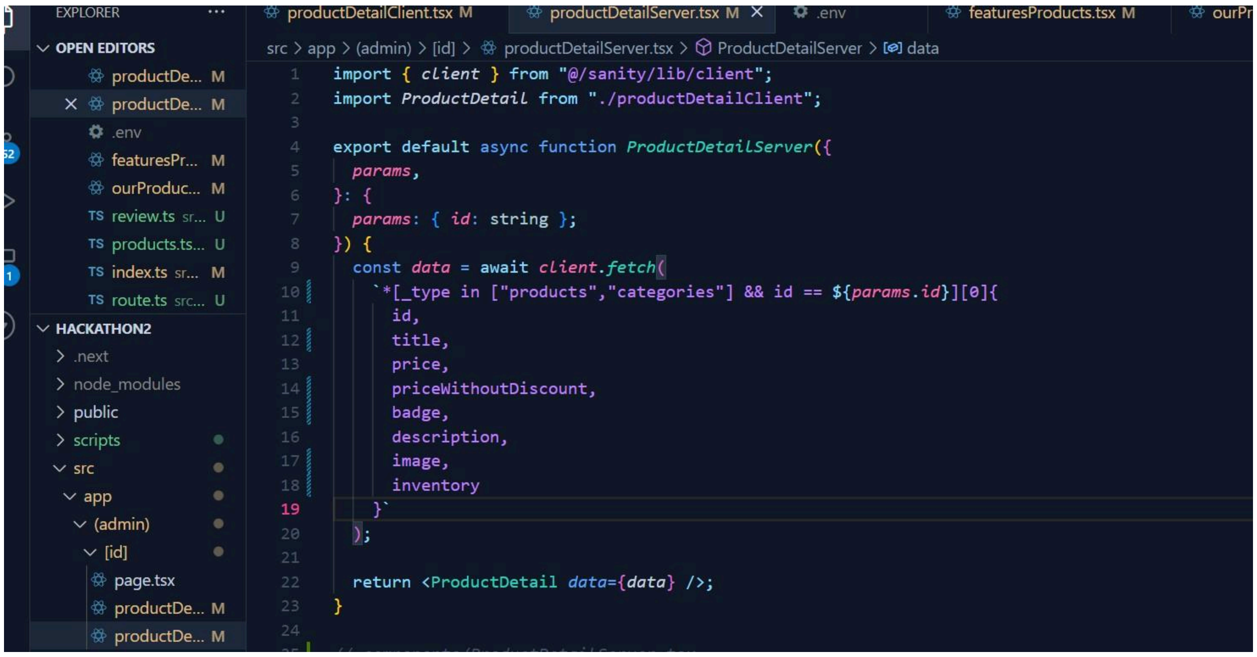
Rose Luxe Armchair

$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

Code Deliverables:



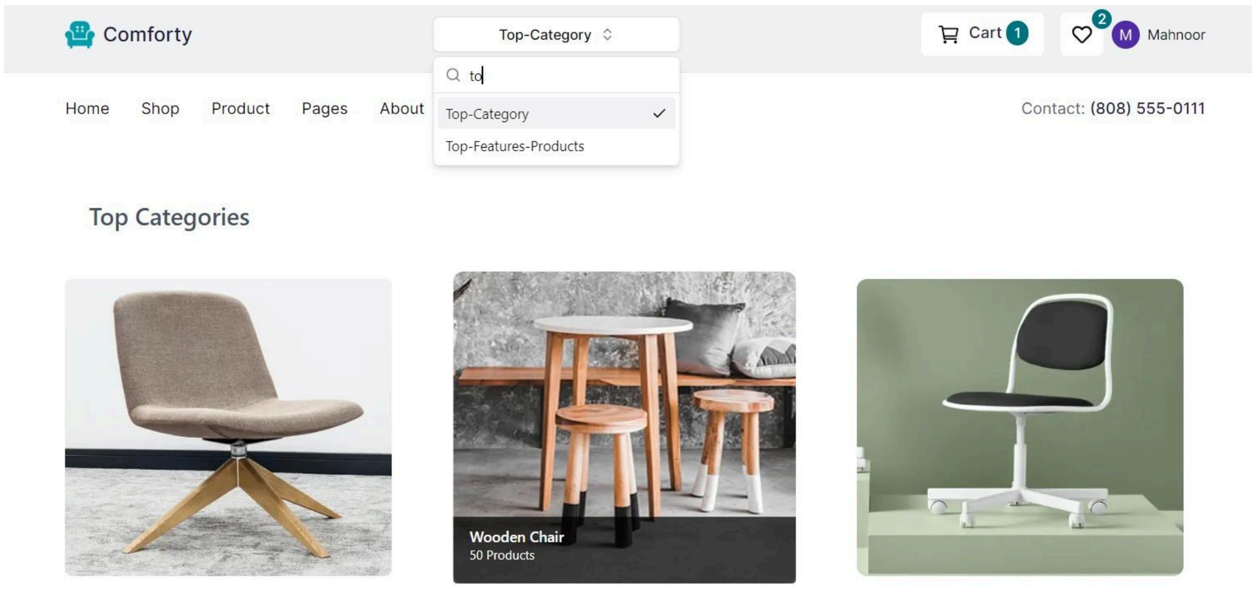
```
import { client } from "@/sanity/lib/client";
import ProductDetail from "./productDetailClient";

export default async function ProductDetailServer({
  params,
}: {
  params: { id: string };
}) {
  const data = await client.fetch(
    `*[_type in ["products","categories"] && id == ${params.id}][0]{
      id,
      title,
      price,
      priceWithoutDiscount,
      badge,
      description,
      image,
      inventory
    }`
  );

  return <ProductDetail data={data} />;
}
```

3. Advanced Category Filters:

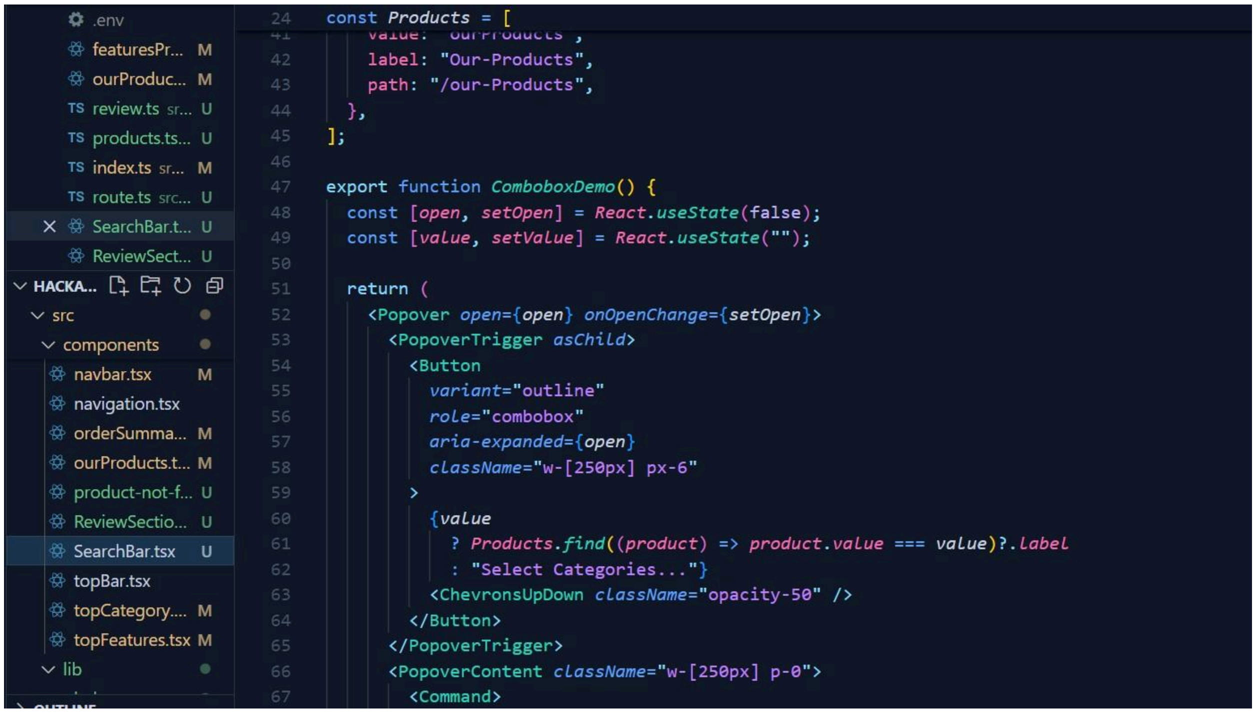
- ***Description:***

  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:

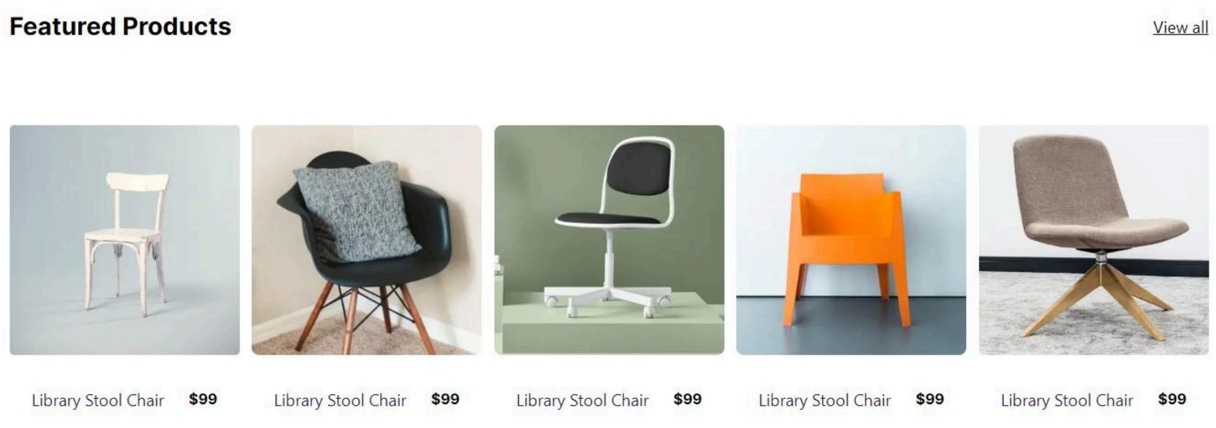


```
const Products = [
    value: "ourProducts",
    label: "Our-Products",
    path: "/our-Products",
  },
];

export function ComboboxDemo() {
  const [open, setOpen] = React.useState(false);
  const [value, setValue] = React.useState("");

  return (
    <Popover open={open} onOpenChange={setOpen}>
      <PopoverTrigger asChild>
        <Button
          variant="outline"
          role="combobox"
          aria-expanded={open}
          className="w-[250px] px-6"
        >
          {value
            ? Products.find((product) => product.value === value)?.label
            : "Select Categories..."}
          <ChevronsUpDown className="opacity-50" />
        </Button>
      </PopoverTrigger>
      <PopoverContent className="w-[250px] p-0">
        <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.

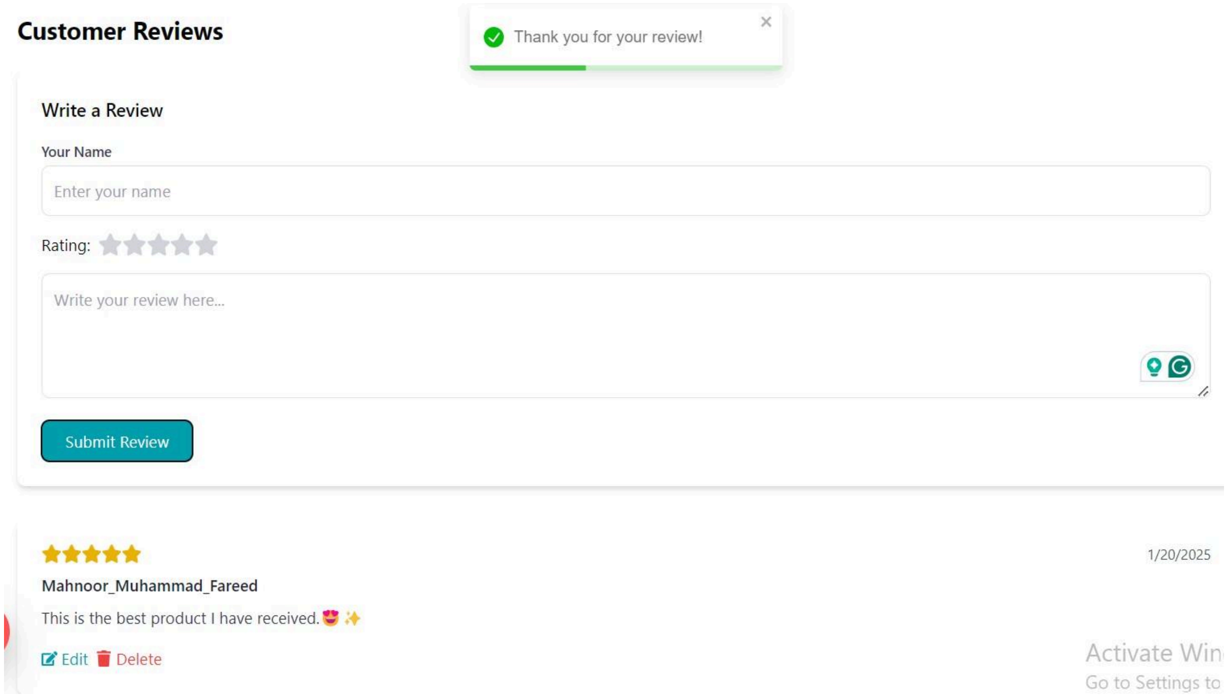
Caption: *Related products section on the product detail page.*



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.



Customer Reviews

Thank you for your review!

Write a Review

Your Name

Enter your name

Rating:

Write your review here...

Submit Review

1/20/2025

Mahnoor_Muhammad_Fareed

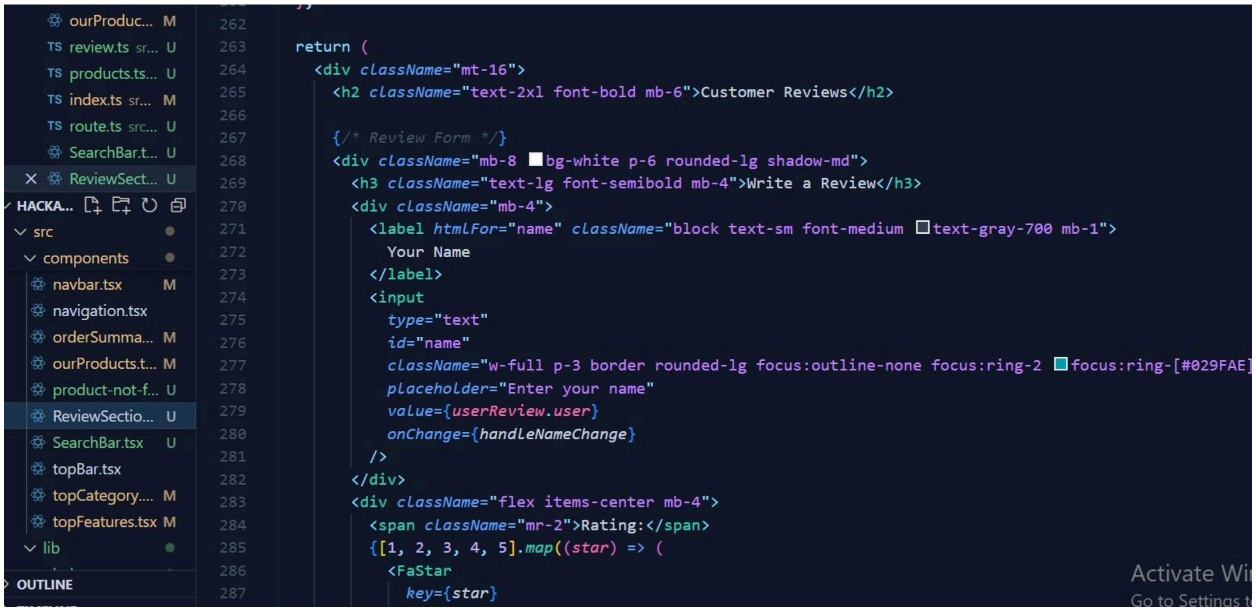
This is the best product I have received.😍✨

Edit Delete

Activate Win
Go to Settings to

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

Code Deliverables:



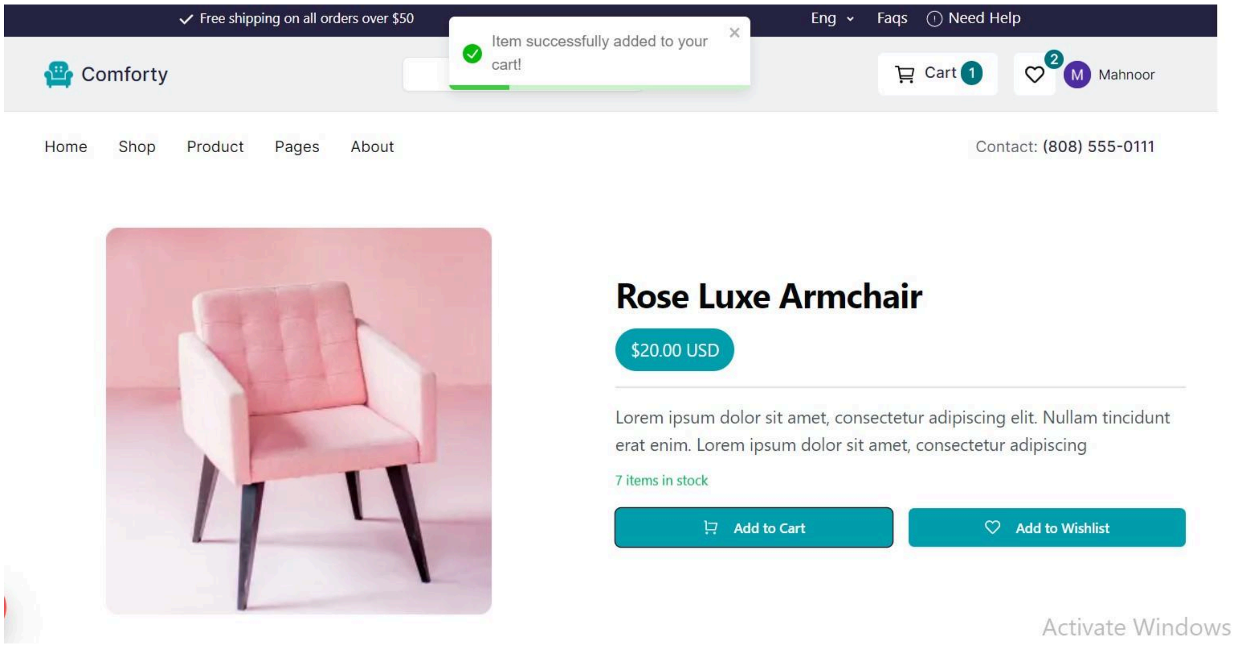
```
    return (
      <div className="mt-16">
        <h2 className="text-2xl font-bold mb-6">Customer Reviews</h2>

        {/* Review Form */}
        <div className="mb-8 bg-white p-6 rounded-lg shadow-md">
          <h3 className="text-lg font-semibold mb-4">Write a Review</h3>
          <div className="mb-4">
            <label htmlFor="name" className="block text-sm font-medium text-gray-700 mb-1">
              Your Name
            </label>
            <input
              type="text"
              id="name"
              className="w-full p-3 border rounded-lg focus:outline-none focus:ring-2 focus:ring-[#029FAE]
              placeholder="Enter your name"
              value={userReview.user}
              onChange={handleNameChange}
            />
          </div>
          <div className="flex items-center mb-4">
            <span className="mr-2">Rating:</span>
            {[1, 2, 3, 4, 5].map((star) => (
              <FaStar
                key={star}
```

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

Item successfully added to your cart!

Eng Faqs Need Help

Comforty

Cart 1 2 M Mahnoor

Home Shop Product Pages About

Contact: (808) 555-0111

Rose Luxe Armchair

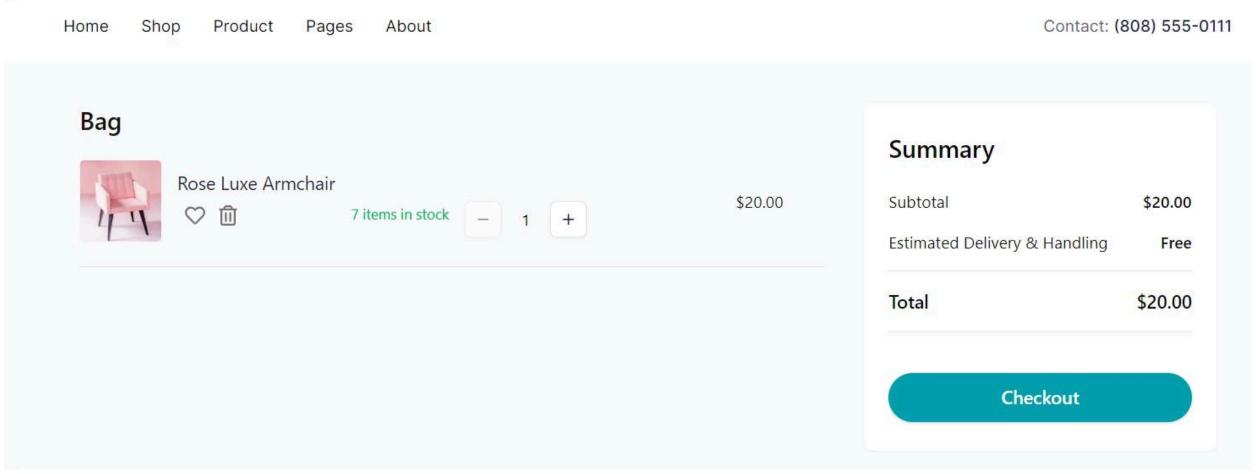
$20.00 USD

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

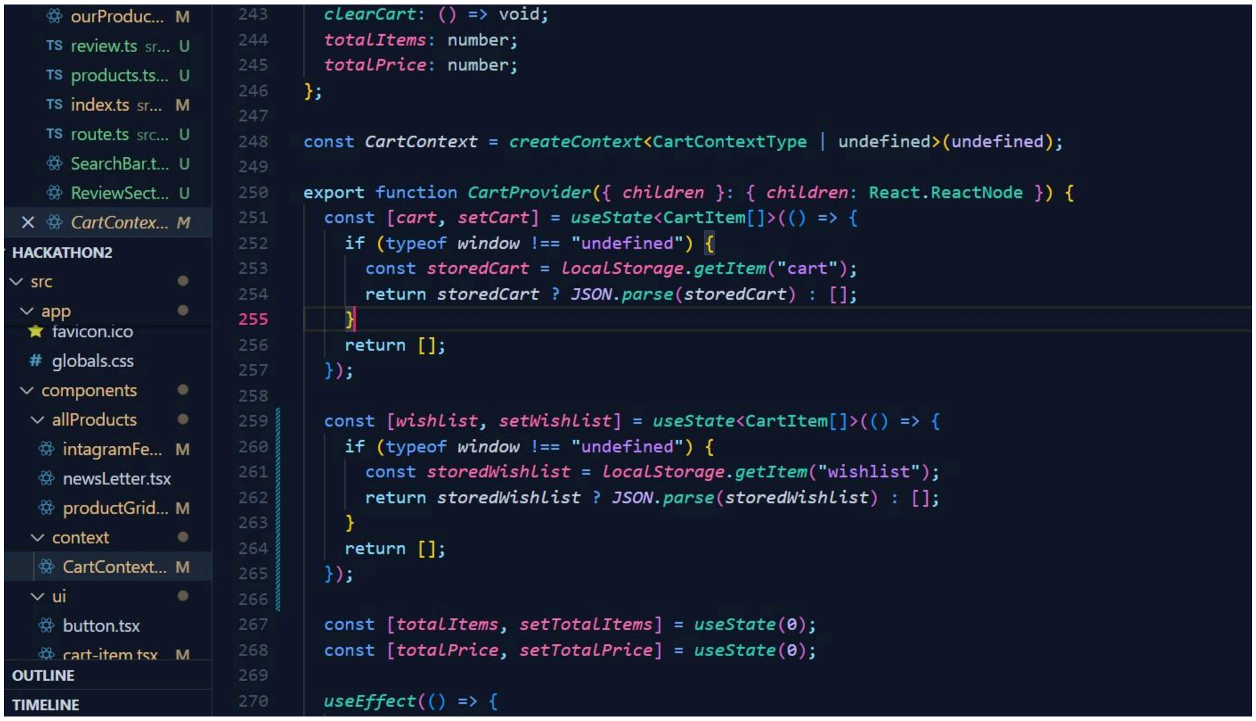
7 items in stock – 1 + $20.00

Summary

Subtotal	$20.00
Estimated Delivery & Handling	Free
Total	$20.00

Checkout

Code Deliverables:



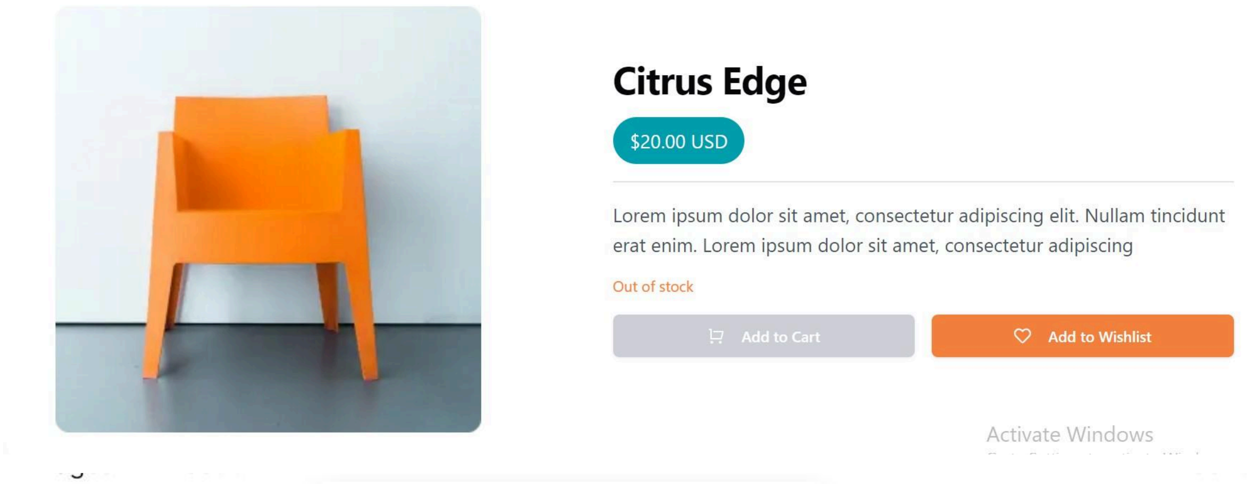
```
  clearCart: () => void;
  totalItems: number;
  totalPrice: number;
};

const CartContext = createContext<CartContextType | undefined>(undefined);

export function CartProvider({ children }: { children: React.ReactNode }) {
  const [cart, setCart] = useState<CartItem[]>(() => {
    if (typeof window !== "undefined") {
      const storedCart = localStorage.getItem("cart");
      return storedCart ? JSON.parse(storedCart) : [];
    }
    return [];
  });

  const [wishlist, setWishlist] = useState<CartItem[]>(() => {
    if (typeof window !== "undefined") {
      const storedWishlist = localStorage.getItem("wishlist");
      return storedWishlist ? JSON.parse(storedWishlist) : [];
    }
    return [];
  });

  const [totalItems, setTotalItems] = useState(0);
  const [totalPrice, setTotalPrice] = useState(0);

  useEffect(() => {
```

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).

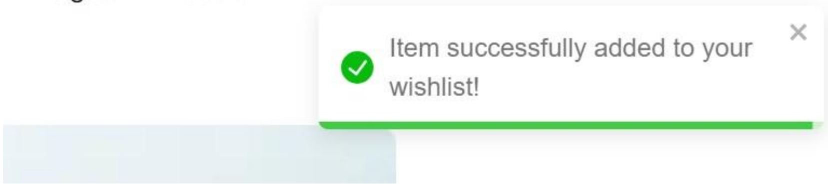
Caption*: Updated stock after order confirmation.*

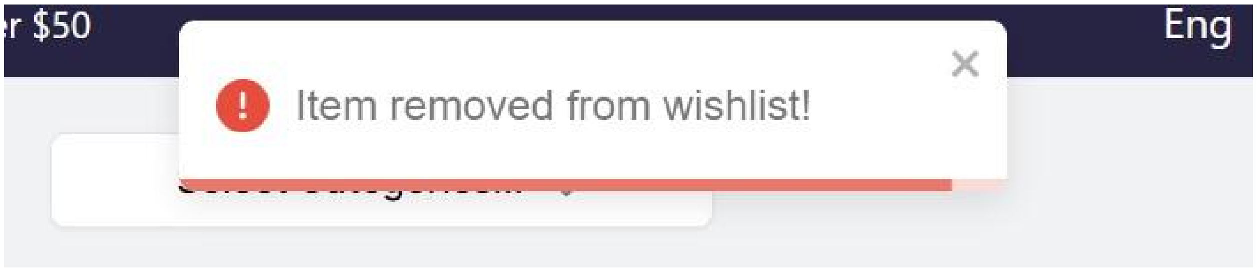

Citrus Edge

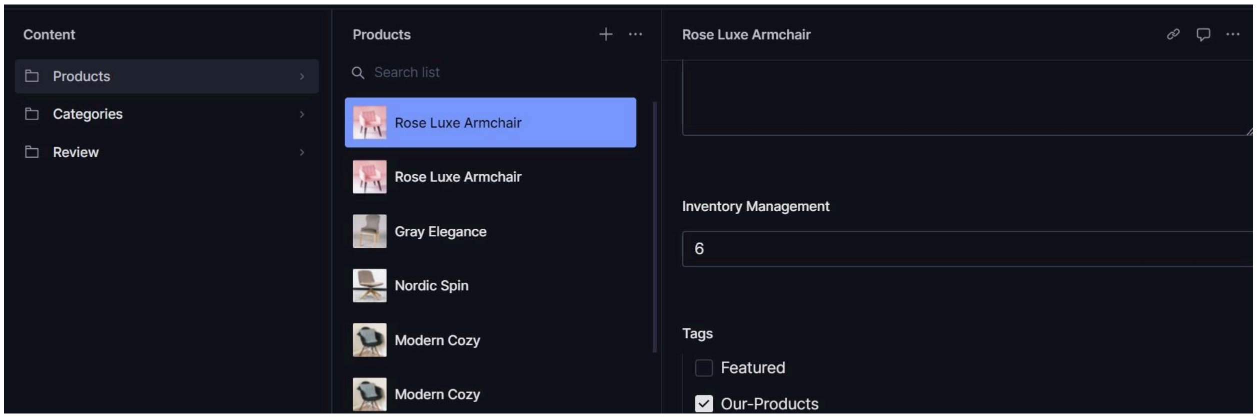
$20.00 USD

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

Out of stock


Item successfully added to your wishlist!


Item removed from wishlist!



Rose Luxe Armchair
Rose Luxe Armchair
Gray Elegance
Nordic Spin
Modern Cozy
Modern Cozy

Rose Luxe Armchair

Inventory Management

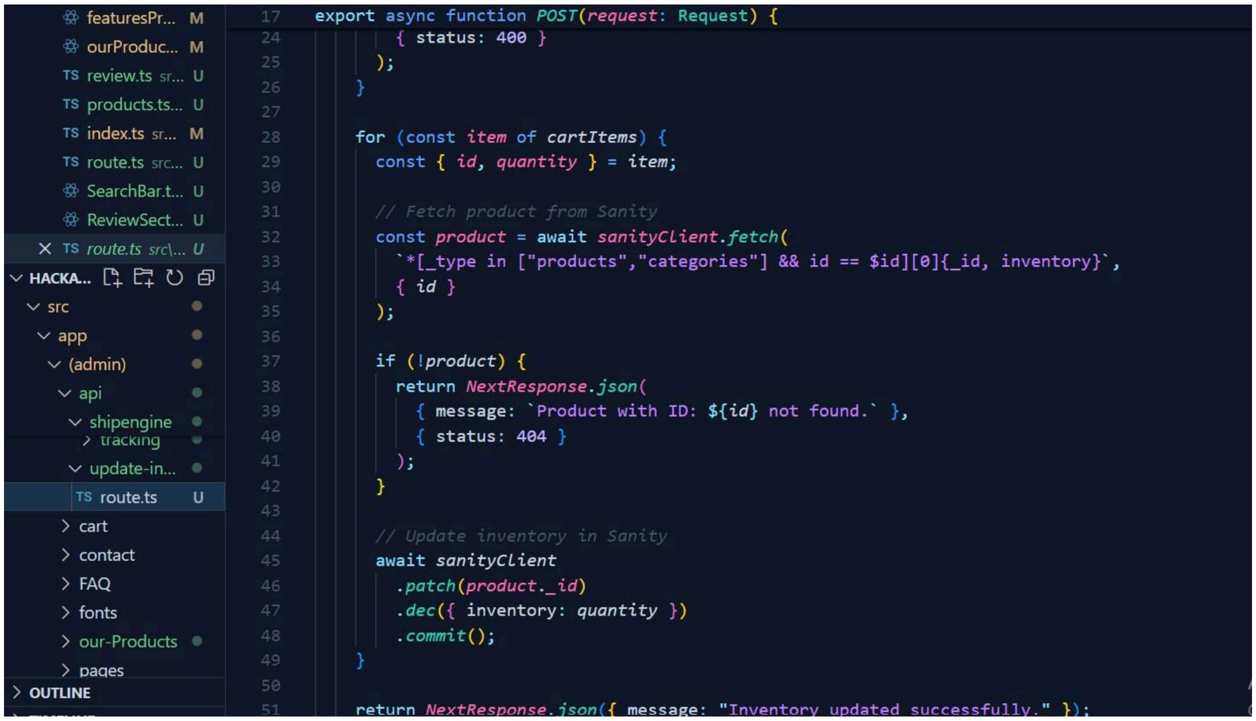
6

Tags

Featured

[✓ Our-Products]

Code Deliverables:



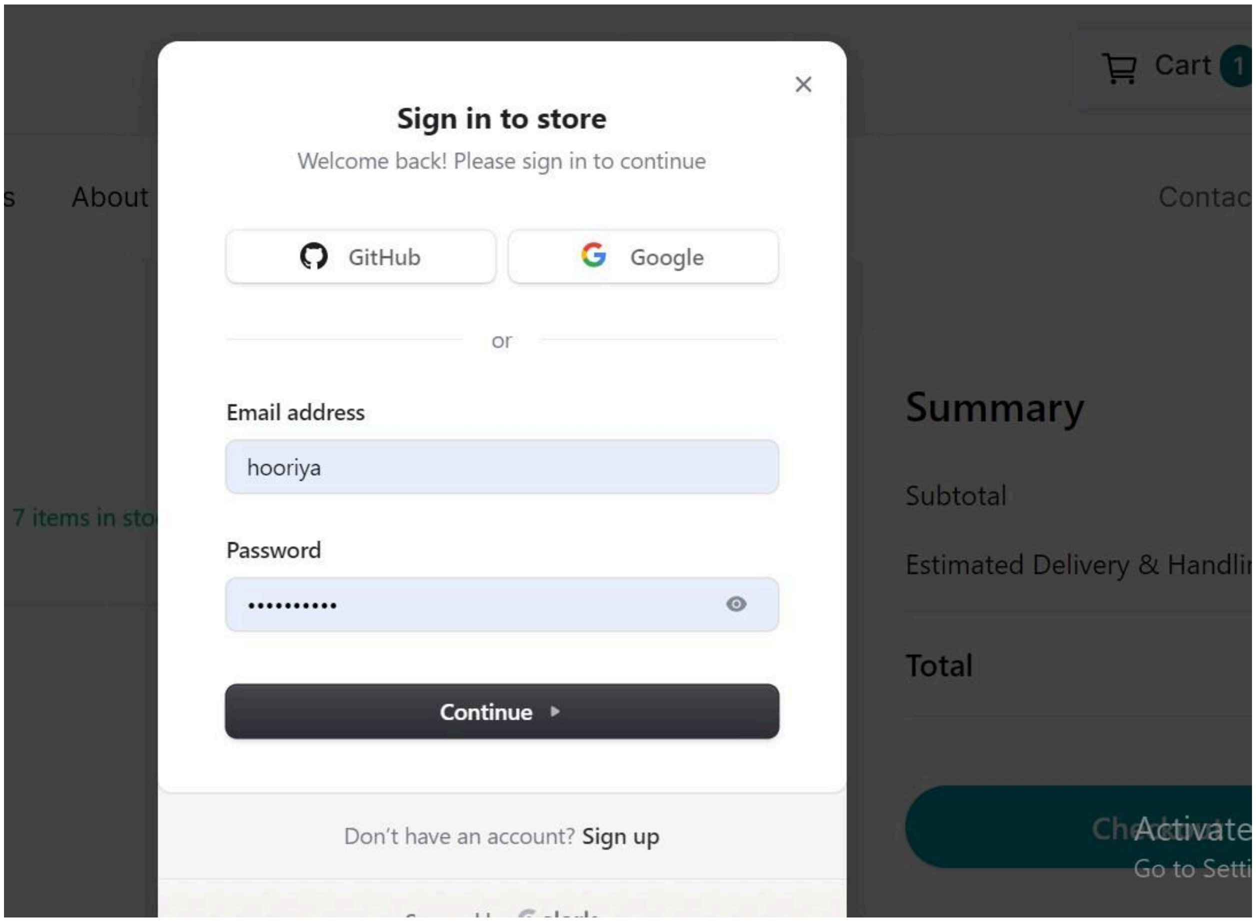
```
export async function POST(request: Request) {
        { status: 400 }
      );
    }

    for (const item of cartItems) {
      const { id, quantity } = item;

      // Fetch product from Sanity
      const product = await sanityClient.fetch(
        `*[_type in ["products","categories"] && id == $id][0]{_id, inventory}`,
        { id }
      );

      if (!product) {
        return NextResponse.json(
          { message: `Product with ID: ${id} not found.` },
          { status: 404 }
        );
      }

      // Update inventory in Sanity
      await sanityClient
        .patch(product._id)
        .dec({ inventory: quantity })
        .commit();
    }

    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*



Sign in to store

Welcome back! Please sign in to continue

GitHub Google

or

Email address

hooriya

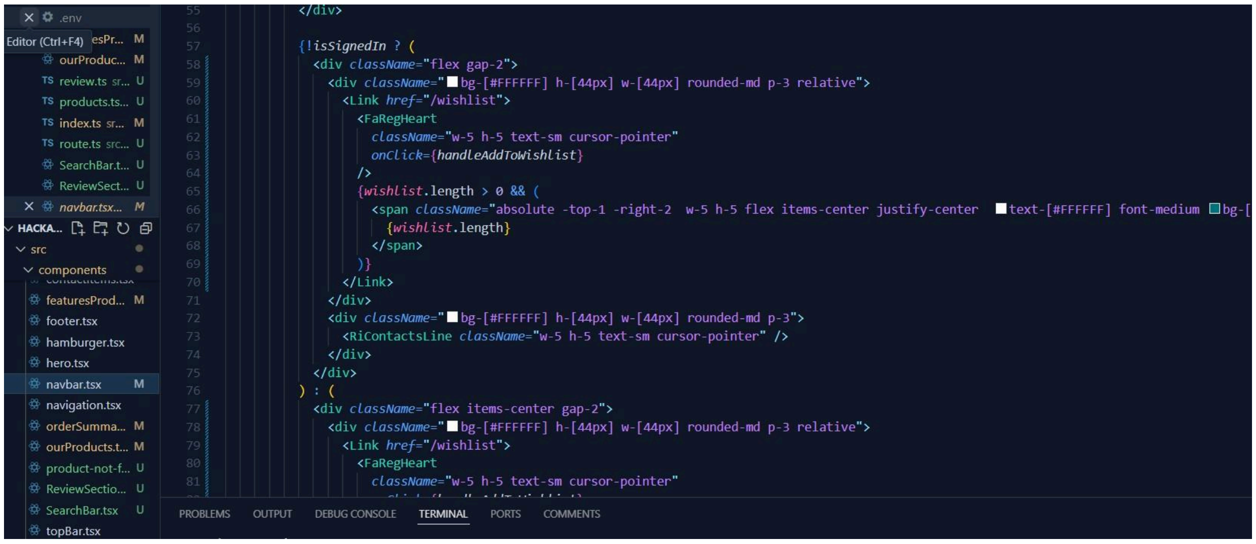
Password

••••••••••

Continue ▸

Don't have an account? Sign up

Code Deliverables:



```
      </div>

      {!isSignedIn ? (
        <div className="flex gap-2">
          <div className="bg-[#FFFFFF] h-[44px] w-[44px] rounded-md p-3 relative">
            <Link href="/wishlist">
              <FaRegHeart
                className="w-5 h-5 text-sm cursor-pointer"
                onClick={handleAddToWishlist}
              />
              {wishlist.length > 0 && (
                <span className="absolute -top-1 -right-2  w-5 h-5 flex items-center justify-center  text-[#FFFFFF] font-medium bg-[
                  {wishlist.length}
                </span>
              )}
            </Link>
          </div>
          <div className="bg-[#FFFFFF] h-[44px] w-[44px] rounded-md p-3">
            <RiContactsLine className="w-5 h-5 text-sm cursor-pointer" />
          </div>
        </div>
      ) : (
        <div className="flex items-center gap-2">
          <div className="bg-[#FFFFFF] h-[44px] w-[44px] rounded-md p-3 relative">
            <Link href="/wishlist">
              <FaRegHeart
                className="w-5 h-5 text-sm cursor-pointer"
```

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.