DAY 4 - BUILDING DYNAMIC FRONTEND COMPONENTS FOR MY FURNITURE E-COMMERCE WEBSITE

Objective: On Day 4, I focused on designing and developing dynamic frontend components to display furniture products fetched from Sanity CMS or APIs. This step emphasizes modular, reusable component design and real-world practices for building scalable and responsive web applications.

Key Learning Outcomes:

1. Build dynamic frontend components to display furniture product data from Sanity CMS or APIs.
2. Implement reusable and modular components.
3. Understand and apply state management techniques.
4. Learn the importance of responsive design and UX/UI best practices.
5. Prepare for real-world client projects by replicating professional workflows.

Key Components Built:

1. **Product Listing Component:**
    - Dynamically render furniture products in a grid layout.
    - Include fields like Product Name, Price, Image, and Stock Status.
    - Example layout: cards displaying product details.
2. **Product Detail Component:**
    - Create individual product detail pages using dynamic routing in Next.js.
    - Include detailed fields such as Product Description, Price, and Available Sizes or Colors.
3. **Category Component:**
    - Display dynamically fetched categories.
    - Enable filtering of products by selected categories.
4. **Search Bar:**
    - Implement search functionality to filter furniture products by name or tags.
5. **Cart Component:**
    - Display added items, quantity, and total price.
    - Use state management for tracking cart items.
6. **Wishlist Component:**
    - Allow users to save furniture items for future reference.
    - Use local storage or a global state management tool to persist data.
7. **Checkout Flow Component:**
    - Create a multi-step form for checkout, including fields for:
        - Billing and shipping address
        - Payment details (mock implementation)
8. **User Profile Component:**
    - Display user-specific details like:
        - Name, email, and saved addresses.
        - Order history with links to individual orders.

9. **Reviews and Ratings Component:**
    - Allow users to view and submit reviews for furniture products.
    - Display average ratings and individual reviews dynamically.
10. **Pagination Component:**

- Break down large product lists into manageable pages.
- Implement previous and next buttons or numbered pagination.

11. **Filter Panel Component:**

- Provide advanced filtering options, such as:
    - Price range sliders
    - Brand selection
    - Availability toggles (e.g., "In Stock" only)

12. **Related Products Component:**

- Suggest similar or complementary furniture items on the product detail page.
- Fetch data based on tags, categories, or customer behaviors.

13. **Footer and Header Components:**

- Build consistent navigation and branding elements.
- Include links to key pages (e.g., Home, About, Contact).
- Ensure responsiveness and accessibility.

14. **Notifications Component:**

- Show real-time alerts for actions like adding to cart, errors, or successful transactions.

Performance Optimization:

- Use modern styling libraries like Tailwind CSS or styled-components.
- Ensure designs are responsive for mobile and desktop views.
- Implement lazy loading for images.
- Use pagination or infinite scrolling for large datasets.

Steps for Implementation:

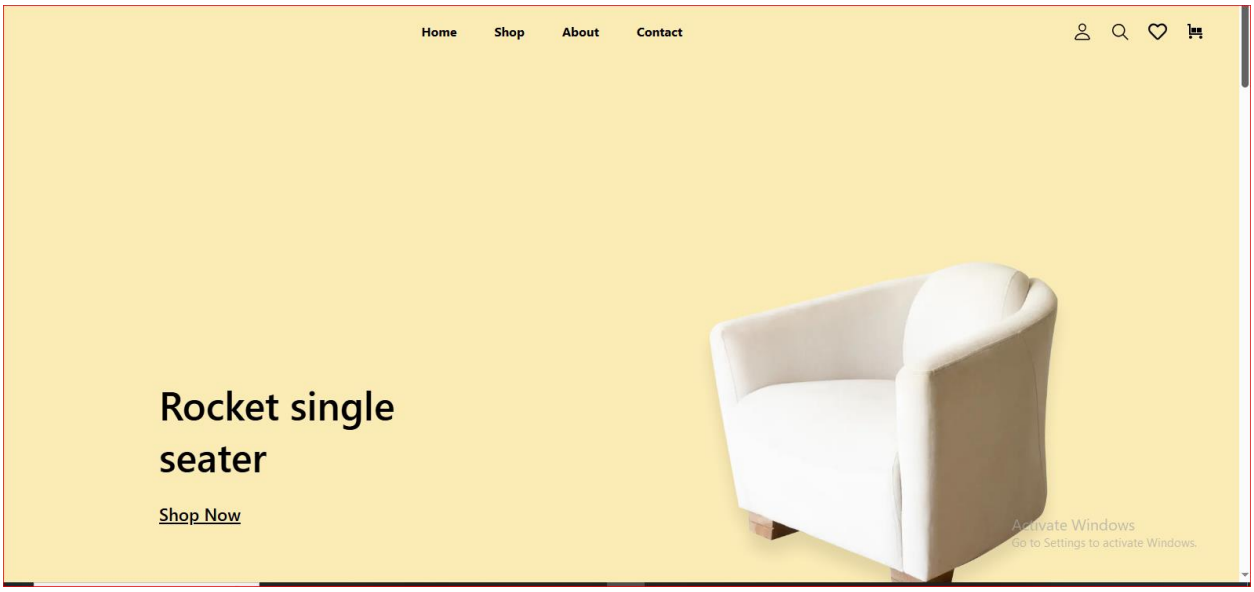
1. **Setup:**
    - Ensure the Next.js project is connected to Sanity CMS or the chosen API.
    - Test data fetching to confirm availability.
2. **Build Components:**
    - Develop and integrate all the key components mentioned above.
    - Ensure modular and reusable structures for future scalability.

Expected Output:

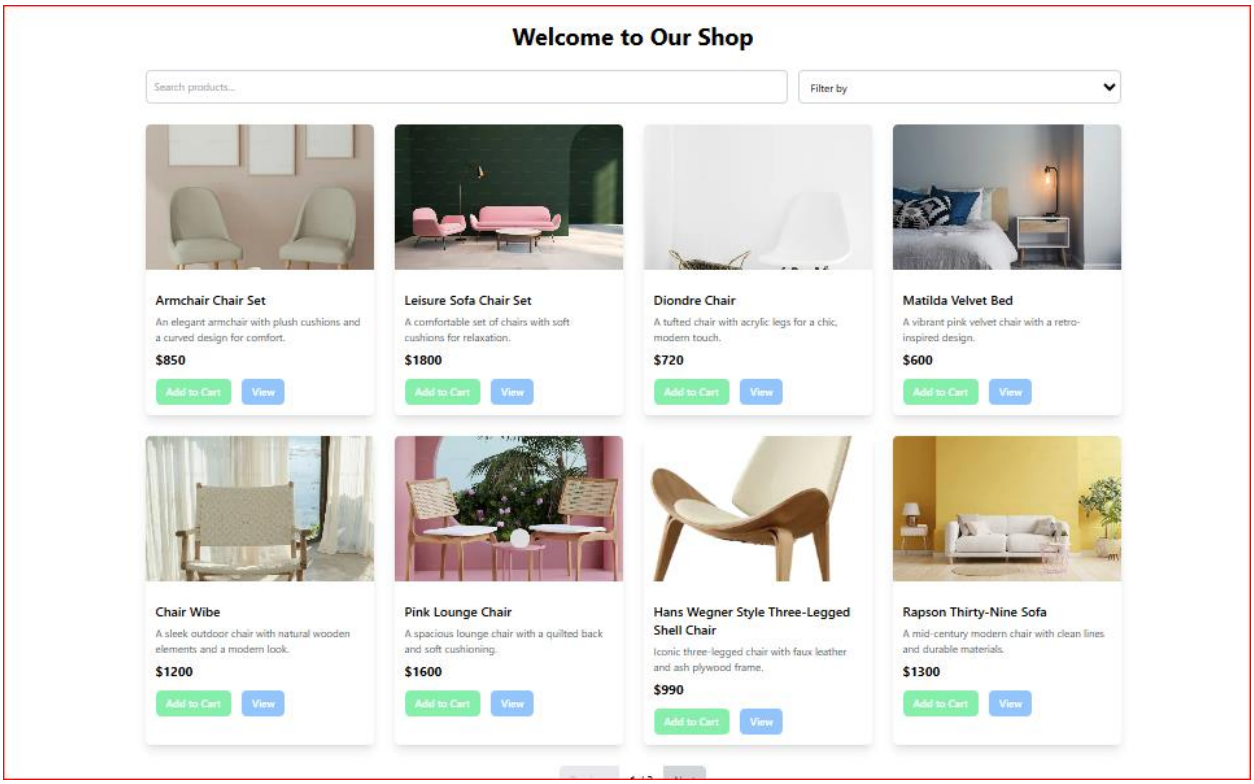
By the end of Day 4, my furniture e-commerce website has:

1. A fully functional product listing page displaying dynamic data from Sanity CMS or APIs.
2. Individual product detail pages implemented using dynamic routing.
3. Advanced category filters to refine and segment product views dynamically.
4. A search bar that effectively filters products by name or tags.
5. Additional features like pagination and related products on detail pages.
6. Components styled to ensure responsiveness and a professional look across devices.
7. Modular and reusable components for future scalability.

My home page

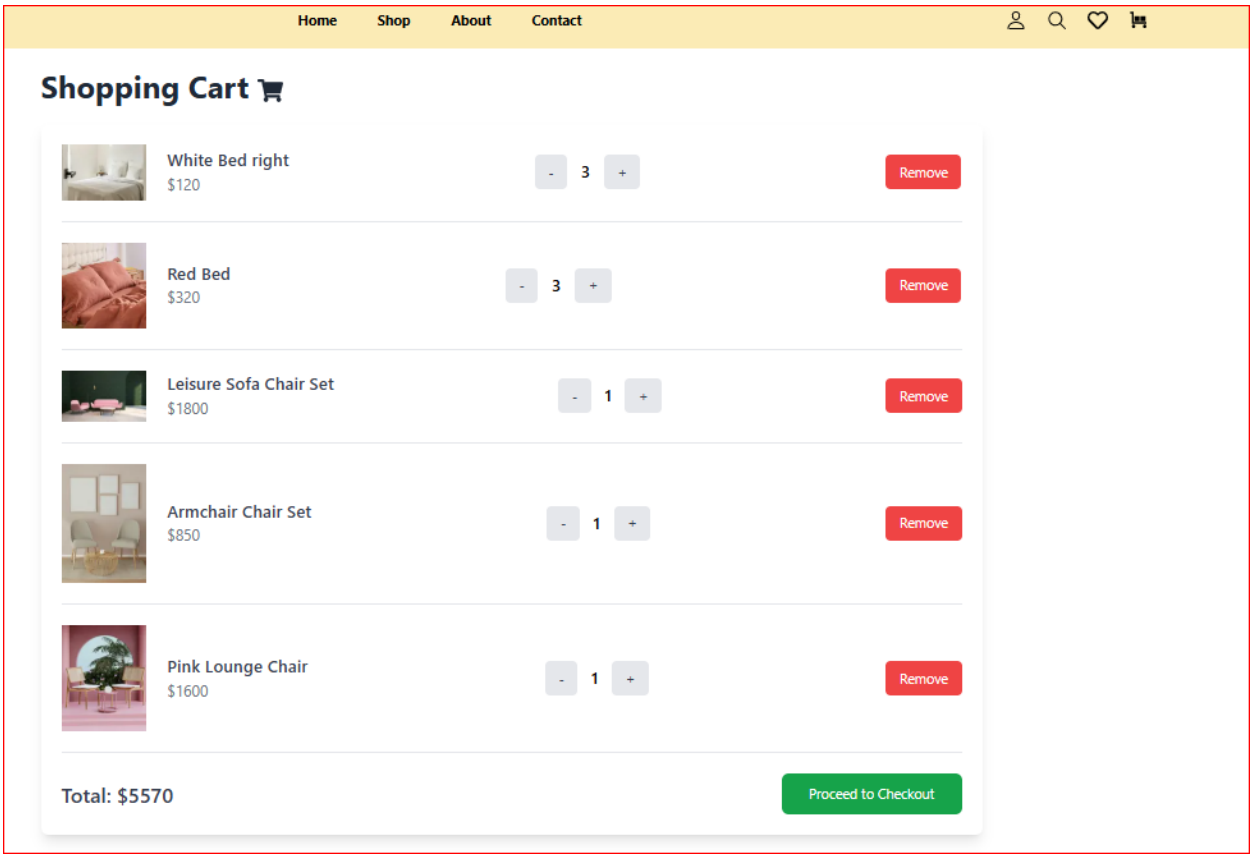


Product listing



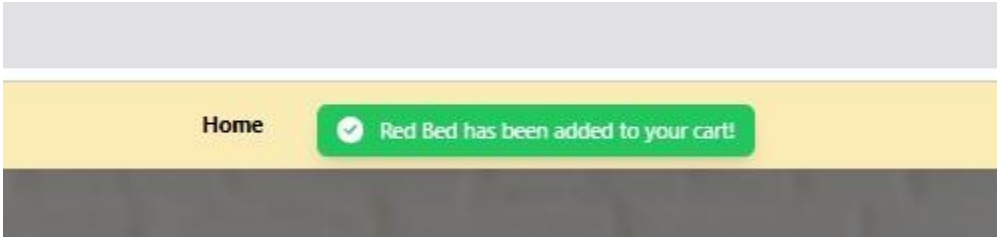
```python
"use client";

import { useRouter } from "next/navigation";
import { useState, useEffect } from "react";
import { client } from "@/sanity/lib/client";

import Image from "next/image";
import { useCart } from "@/app/context/CartContext";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faCheckCircle } from "@fortawesome/free-solid-svg-icons";

interface Product {
  _id: string;
  name: string;
  price: number;
  description: string;
  image: { asset: { url: string } }; // Assuming Sanity image structure
  slug: { current: string };
}

const ShopPage = () => {
  const router = useRouter();
  const { addToCart } = useCart();

  const query = `*[_type == "product" && defined(slug.current)] {
    _id,
    name,
    price,
    description,
     image { asset -> { url } },
    slug
  }`;

  const [products, setProducts] = useState<Product[]>([]);
  const [filteredProducts, setFilteredProducts] = useState<Product[]>([]);
  const [searchQuery, setSearchQuery] = useState("");
  const [selectedFilter, setSelectedFilter] = useState("");
  const [loading, setLoading] = useState(true);
  const [notification, setNotification] = useState<string | null>(null);
  const [isVisible, setIsVisible] = useState(false);

  const [currentPage, setCurrentPage] = useState(1);
  const productsPerPage = 8;

  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const products = await client.fetch(query);
        setProducts(products);
        setFilteredProducts(products);
      } catch (error) {
        console.error("Failed to fetch products:", error);
      } finally {
        setLoading(false);
      }
    };
    fetchProducts();
  }, [query]);

  const handleSearch = (e: React.ChangeEvent<HTMLInputElement>) => {
    const query = e.target.value.toLowerCase();
    setSearchQuery(query);

    const filtered = products.filter(
      (product) =>
        product.name.toLowerCase().includes(query) ||
        (product.description &&
          product.description.toLowerCase().includes(query))
    );
    setFilteredProducts(filtered);
  };

  const handleFilterChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
    const selected = e.target.value;
    setSelectedFilter(selected);

    if (selected === "price-low-high") {
      setFilteredProducts([...products].sort((a, b) => a.price - b.price));
    } else if (selected === "price-high-low") {
      setFilteredProducts([...products].sort((a, b) => b.price - a.price));
    } else {
      setFilteredProducts(products);
    }
  };

  const handleAddToCart = (product: Product) => {
    const item = {
      id: product._id
```

Shopping cart

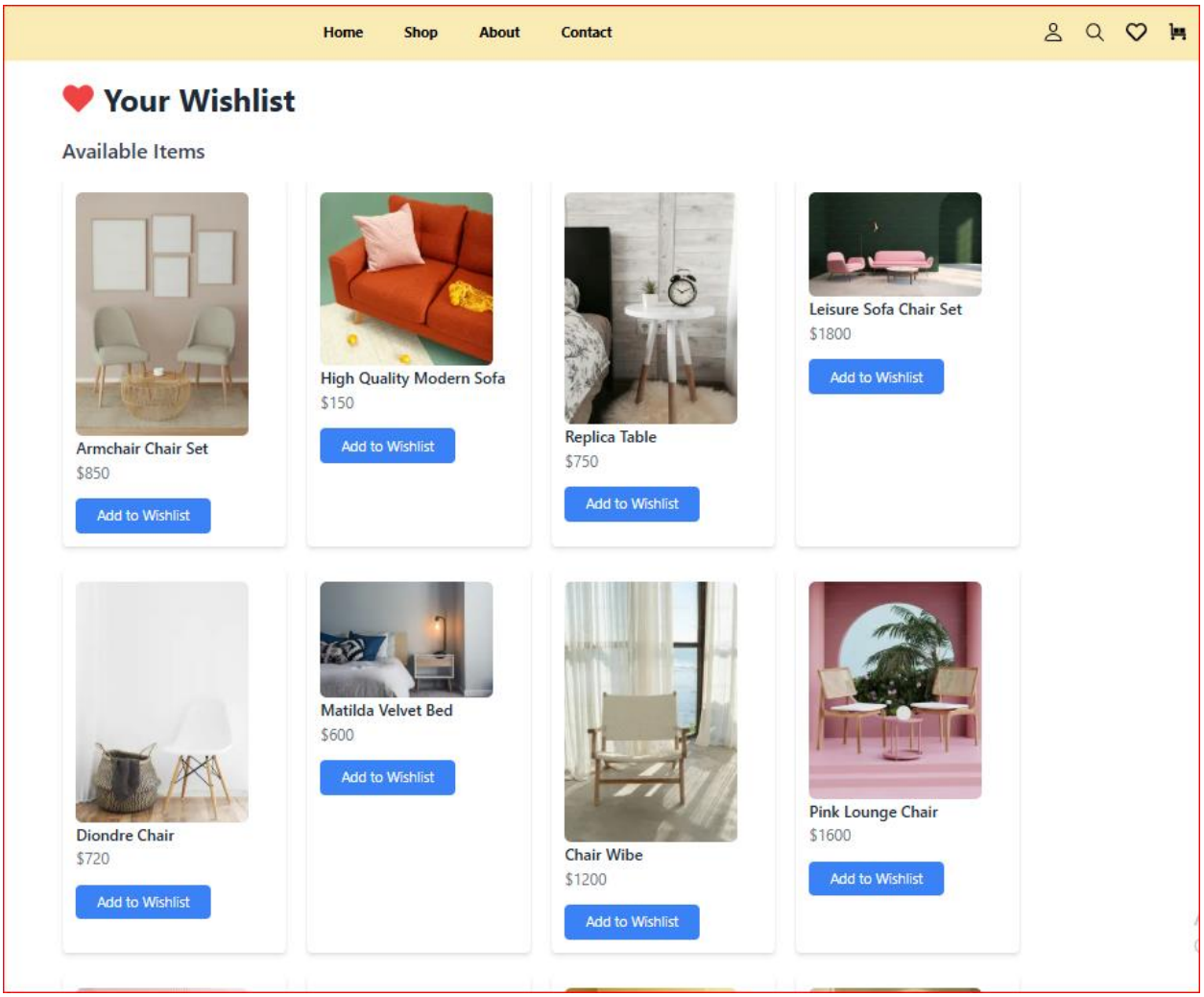


```python
"use client";

import { useCart } from "@/app/context/CartContext";
import { FaShoppingCart } from "react-icons/fa";
import { useRouter } from "next/navigation";
import { urlFor } from "@/sanity/lib/image";
import { useEffect } from "react";
import Image from "next/image";

const CartPage = () => {
 const { cart,  setCart } = useCart();
 const router = useRouter();

 useEffect(() => {
   const storedCart = localStorage.getItem("cart");
   if (storedCart) {
     setCart(JSON.parse(storedCart));
   }
 }, [setCart]);

 const handleCheckout = () => {
   localStorage.removeItem("cart");
   router.push("/checkout");
 };

 // Handle increase quantity
 const handleIncreaseQuantity = (id: string) => {
   const updatedCart = cart.map(item => {
     if (item.id === id) {
       console.log(`Increasing quantity for`, item.name, item.quantity); // Debugging line
       return { ...item, quantity: item.quantity + 1 };
     }
     return item;
   });
   setCart(updatedCart);  // Update the state
   localStorage.setItem("cart", JSON.stringify(updatedCart));  // Update localStorage
 };

 // Handle decrease quantity
 const handleDecreaseQuantity = (id: string, quantity: number) => {
   if (quantity === 1) {
     handleRemoveItem(id);  // Remove item if quantity is 1
   } else {
     const updatedCart = cart.map(item => {
       if (item.id === id) {
         console.log(`Decreasing quantity for`, item.name, item.quantity); // Debugging line
         return { ...item, quantity: item.quantity - 1 };
       }
       return item;
     });
     setCart(updatedCart);  // Update the state
     localStorage.setItem("cart", JSON.stringify(updatedCart));  // Update localStorage
   }
 };

 // Remove item from the cart
 const handleRemoveItem = (id: string) => {
   const updatedCart = cart.filter(item => item.id !== id);  // Remove item from cart
   setCart(updatedCart);  // Update state
   localStorage.setItem("cart", JSON.stringify(updatedCart));  // Update localStorage
 };

 return (
   <div className="max-w-6xl mx-auto p-6">
     <h1 className="text-4xl font-bold text-gray-800 mb-6">
       Shopping Cart <FaShoppingCart className="inline-block text-3xl" />
     </h1>

     {cart.length === 0 ? (
       <p className="text-gray-500 text-center text-lg">Your cart is empty 😔</p>
     ) : (
       <div className="bg-white shadow-lg rounded-lg p-6">
         {cart.map((item) => (

           <div key={item.id} className="flex items-center justify-between border-b pb-6 mb-6">
             <div className="flex items-center gap-6">

               {item.image && (
                 <Image
                   src={typeof item.image === "string" ? item.image : urlFor({ ...item.image, _type: "image", asset: { ...item.image.asset, _type: "reference" } }).url()}
                   alt={item.name}
                   width={100}
                   height={100}
                 />
               )}
               <div>
                 <p className="text-xl font-semibold text-gray-700">{item.name}</p>
                 <p className="text-gray-500 text-lg">${item.price}</p>
               </div>
             </div>

             <div className="flex items-center gap-4">
               <button
                 onClick={() => handleDecreaseQuantity(item.id, item.quantity)}
                 className="bg-gray-200 hover:bg-gray-300 text-gray-700 px-4 py-2 rounded-md"
               >
                 -
               </button>
               <span className="text-xl font-semibold">{item.quantity}</span>
               <button
                 onClick={() => handleIncreaseQuantity(item.id)}
                 className="bg-gray-200 hover:bg-gray-300 text-gray-700 px-4 py-2 rounded-md"
               >
                 +
               </button>
             </div>

             <button
               onClick={() => handleRemoveItem(item.id)}
               className="bg-red-500 hover:bg-red-600 text-white px-4 py-2 rounded-md"
             >
               Remove
             </button>
           </div>
         ))}

         <div className="flex justify-between items-center mt-6">
           <p class="text-2xl font-semibold text-gray-700">
             Total: ${cart.reduce((total, item) => total + item.price * item.quantity, 0)}
           </p>
           <button onClick={handleCheckout} class="bg-green-600 hover:bg-green-700 text-white px-8 py-3 rounded-lg">
             Proceed to Checkout
           </button>
         </div>
       </div>
     )}
   </div>
 );
};

export default CartPage;
```

Add cart Notitification

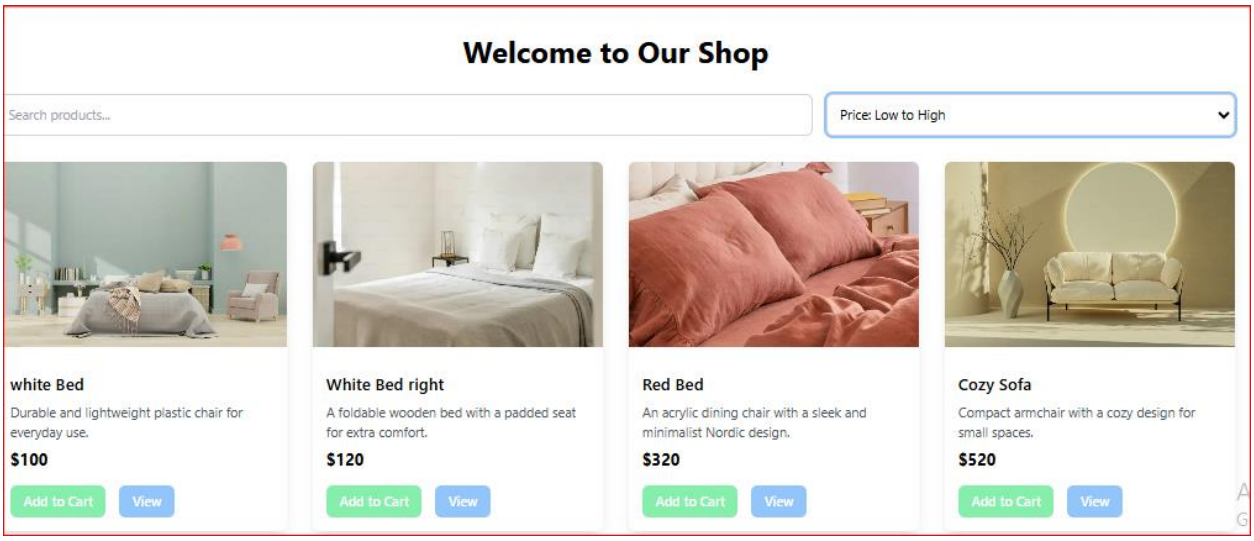


Wishlist

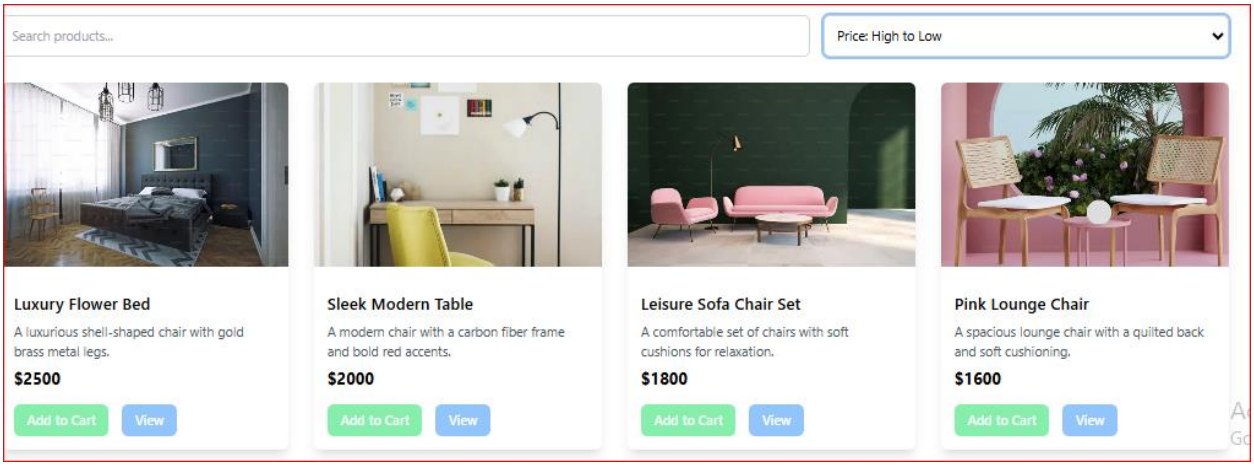


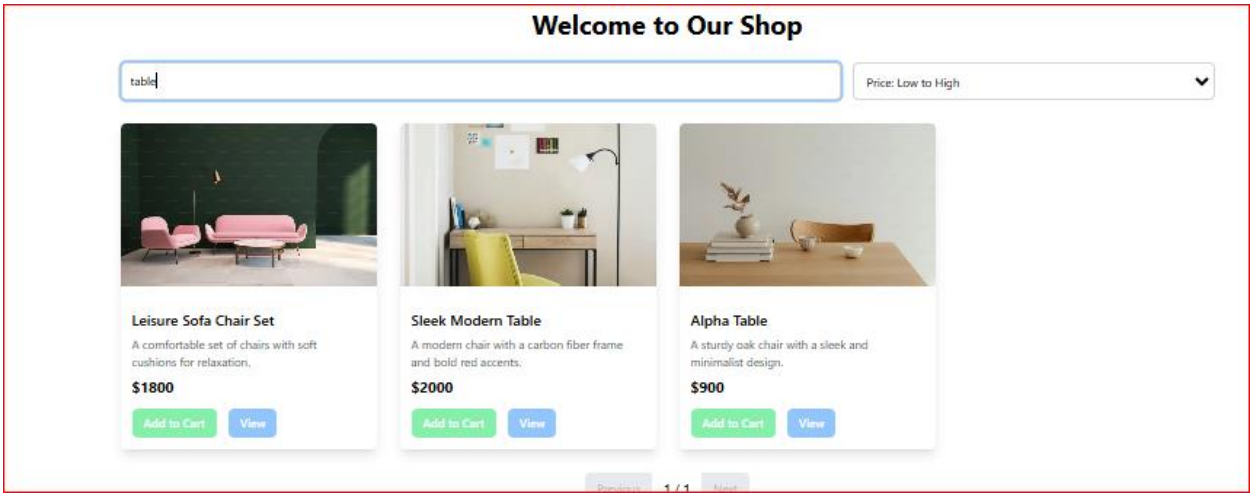
```

"use client";

import { useState, useEffect } from "react";
import { FaHeart } from "react-icons/fa";
import { client } from "@/sanity/lib/client"; // Sanity client import
import { urlFor } from "@/sanity/lib/image"; // Sanity image helper
import Image from "next/image";
interface ImageAsset {
  _type: string;
  asset: {
    _ref: string;
    _type: string;
  };
}

interface WishlistItem {
  _id: string; // Sanity document ID
  name: string;
  price: number;
  image: ImageAsset; // More specific type for Sanity image reference
}

const WishlistPage = () => {
  const [wishlist, setWishlist] = useState<WishlistItem[]>([]);  // Initial empty wishlist state
  const [items, setItems] = useState<WishlistItem[]>([]);

  // Load available items from Sanity and wishlist from localStorage
  useEffect(() => {
    // Fetch items from Sanity
    client.fetch<WishlistItem[]>(`*[_type == "product"]`).then((data: WishlistItem[]) => {
      setItems(data); // Set fetched items for selection
    }).catch((err: Error) => {
      console.error("Error fetching items: ", err);
    });

    // Load wishlist from localStorage only if there are items
    const storedWishlist = localStorage.getItem("wishlist");
    if (storedWishlist) {
      setWishlist(JSON.parse(storedWishlist)); // Update state with items in localStorage
    }
  }, []);

  // Add item to wishlist
  const addToWishlist = (item: WishlistItem) => {
    const currentWishlist = JSON.parse(localStorage.getItem('wishlist') || '[]');

    // Check if item already exists in wishlist
    const isItemInWishlist = currentWishlist.some((wishlistItem: WishlistItem) => wishlistItem._id === item._id);

    if (!isItemInWishlist) {
      const updatedWishlist = [...currentWishlist, item];
      localStorage.setItem('wishlist', JSON.stringify(updatedWishlist)); // Update localStorage
      setWishlist(updatedWishlist); // Update state to re-render the wishlist
    }
  };

  // Remove item from wishlist
  const removeFromWishlist = (id: string) => {
    const updatedWishlist = wishlist.filter(item => item._id !== id);
    setWishlist(updatedWishlist);
    localStorage.setItem("wishlist", JSON.stringify(updatedWishlist)); // Update localStorage
  };

  return (
    <div className="max-w-6xl mx-auto p-6">
      <h1 className="text-4xl font-bold text-gray-800 mb-6 flex items-center">
        <FaHeart className="text-red-500 mr-3" />
        Your Wishlist
      </h1>

      {/* Item Selection Section */}
      <h2 className="text-2xl font-semibold text-gray-700 mb-4">Available Items</h2>
      <div className="grid grid-cols-2 sm:grid-cols-3 md:grid-cols-4 gap-6">
        {items.map((item) => (
          <div key={item._id} className="bg-white shadow-md rounded-md p-4">
            <Image
  src={urlFor(item.image)?.url() || "/placeholder.jpg"} // Use optimized Sanity image
  alt={item.name}
  width={200}
  height={200}
```

Filtration




```
const filtered = products.filter(
    (product) =>
      product.name.toLowerCase().includes(query) ||
      (product.description &&
        product.description.toLowerCase().includes(query))
  );
  setFilteredProducts(filtered);
};

const handleFilterChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
  const selected = e.target.value;
  setSelectedFilter(selected);

  if (selected === "price-low-high") {
    setFilteredProducts([...products].sort((a, b) => a.price - b.price));
  } else if (selected === "price-high-low") {
    setFilteredProducts([...products].sort((a, b) => b.price - a.price));
  } else {
    setFilteredProducts(products);
  }
};
```


Search products...
Price: High to Low
Luxury Flower Bed
A luxurious shell-shaped chair with gold brass metal legs.
$2500
Add to Cart
View
Sleek Modern Table
A modern chair with a carbon fiber frame and bold red accents.
$2000
Add to Cart
View
Leisure Sofa Chair Set
A comfortable set of chairs with soft cushions for relaxation.
$1800
Add to Cart
View
Pink Lounge Chair
A spacious lounge chair with a quilted back and soft cushioning.
$1600
Add to Cart
View


Welcome to Our Shop
table
Price: Low to High
Leisure Sofa Chair Set
A comfortable set of chairs with soft cushions for relaxation.
$1800
Add to Cart
View
Sleek Modern Table
A modern chair with a carbon fiber frame and bold red accents.
$2000
Add to Cart
View
Alpha Table
A sturdy oak chair with a sleek and minimalist design.
$900
Add to Cart
View
1 / 1

Search bar

Welcome to Our Shop

Search products...

Filter by

Welcome to Our Shop

table

Price: Low to High

Leisure Sofa Chair Set

A comfortable set of chairs with soft cushions for relaxation.

$1800

Add to Cart View

Sleek Modern Table

A modern chair with a carbon fiber frame and bold red accents.

$2000

Add to Cart View

Alpha Table

A sturdy oak chair with a sleek and minimalist design.

$900

Add to Cart View

Previous 1 / 1 Next

Welcome to Our Shop

sofa

Filter by

Leisure Sofa Chair Set

A comfortable set of chairs with soft cushions for relaxation.

$1800

Add to Cart View

Rapson Thirty-Nine Sofa

A mid-century modern chair with clean lines and durable materials.

$1300

Add to Cart View

Cozy Sofa

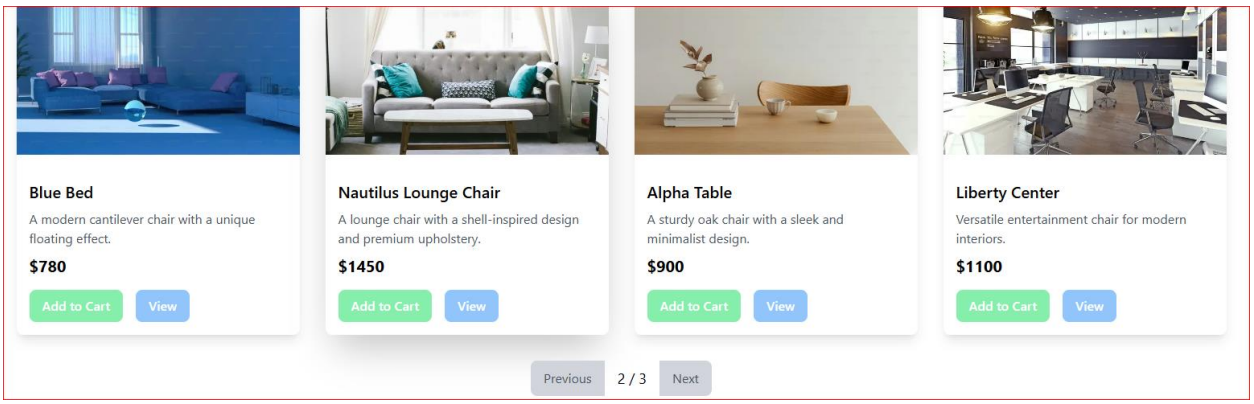
Compact armchair with a cozy design for small spaces.

$520

Add to Cart View

Previous 1 / 1 Next

Pagination



Daynamic Routing

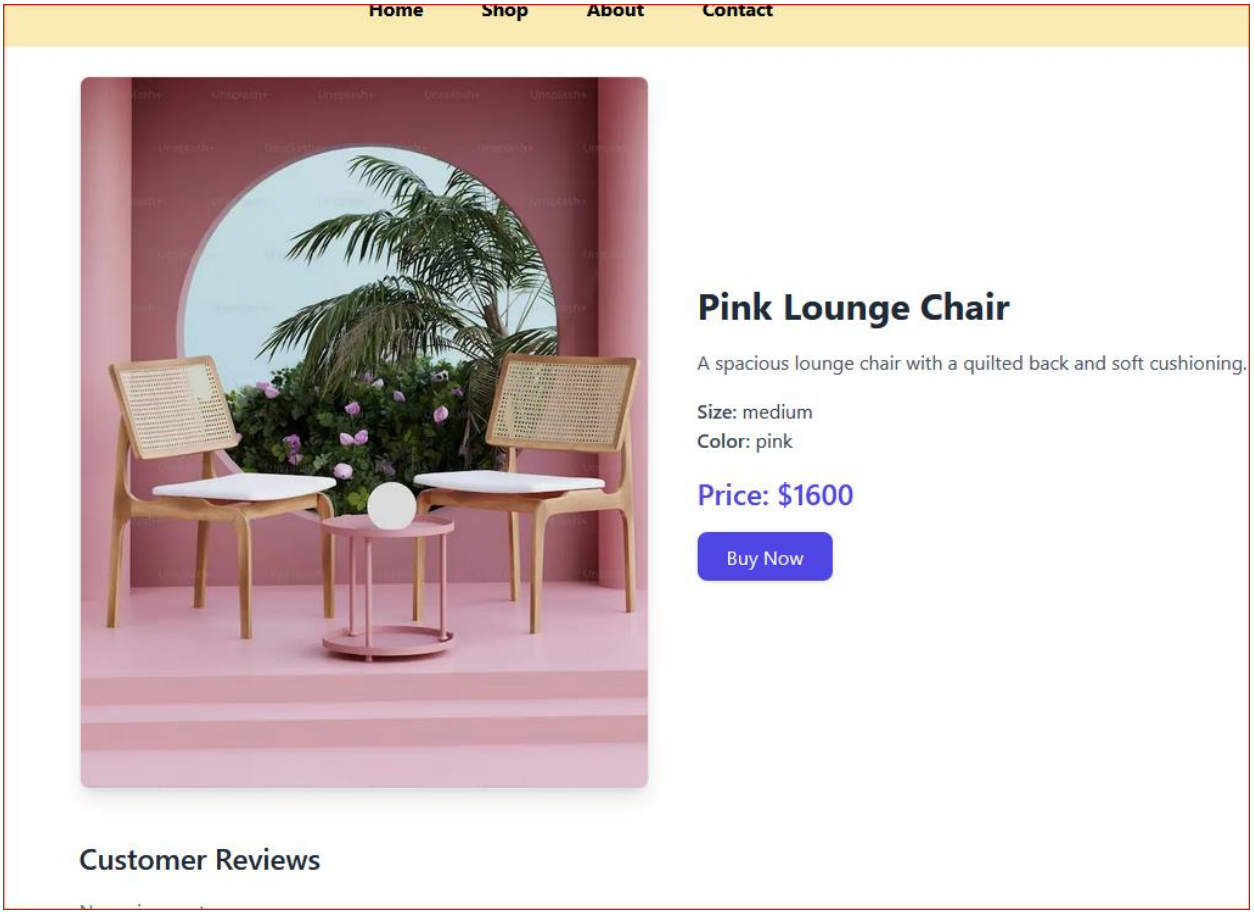
Product 1

Pink Lounge Chair

A spacious lounge chair with a quilted back and soft cushioning.

$1600

Add to Cart View


Home
Shop
About
Contact
Pink Lounge Chair
A spacious lounge chair with a quilted back and soft cushioning.
Size: medium
Color: pink
Price: $1600
Buy Now
Customer Reviews

Product 2

Cozy Sofa

Compact armchair with a cozy design for small spaces.

$520

Add to Cart View

Home Shop About Contact

Cozy Sofa

Compact armchair with a cozy design for small spaces.

Size:
Color:

Price: $520

Buy Now

Customer Reviews

No reviews yet.

Home Shop About Contact

Nautilus Lounge Chair

A lounge chair with a shell-inspired design and premium upholstery.

Size: Medium
Color:

Price: $1450

Buy Now

Customer Reviews

No reviews yet.

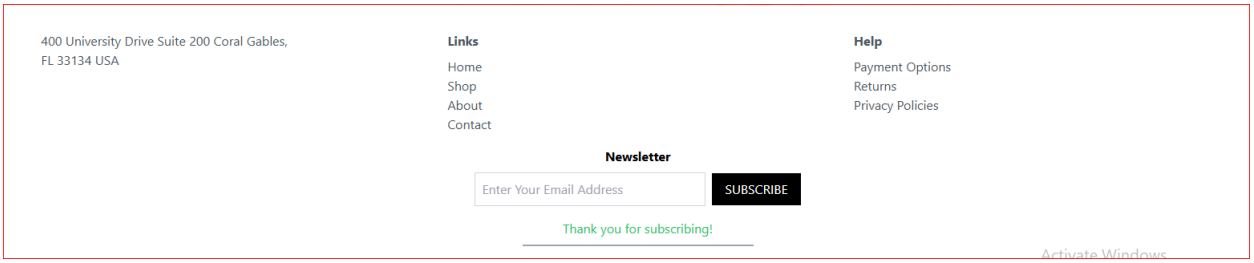
Nautilus Lounge Chair

A lounge chair with a shell-inspired design and premium upholstery.

$1450

Add to Cart View

Footer



✅ Self-Validation Checklist for Day 4

Frontend Component Development:

✓

Styling and Responsiveness:

✓

Code Quality:

✓

Documentation and Submission:

✓