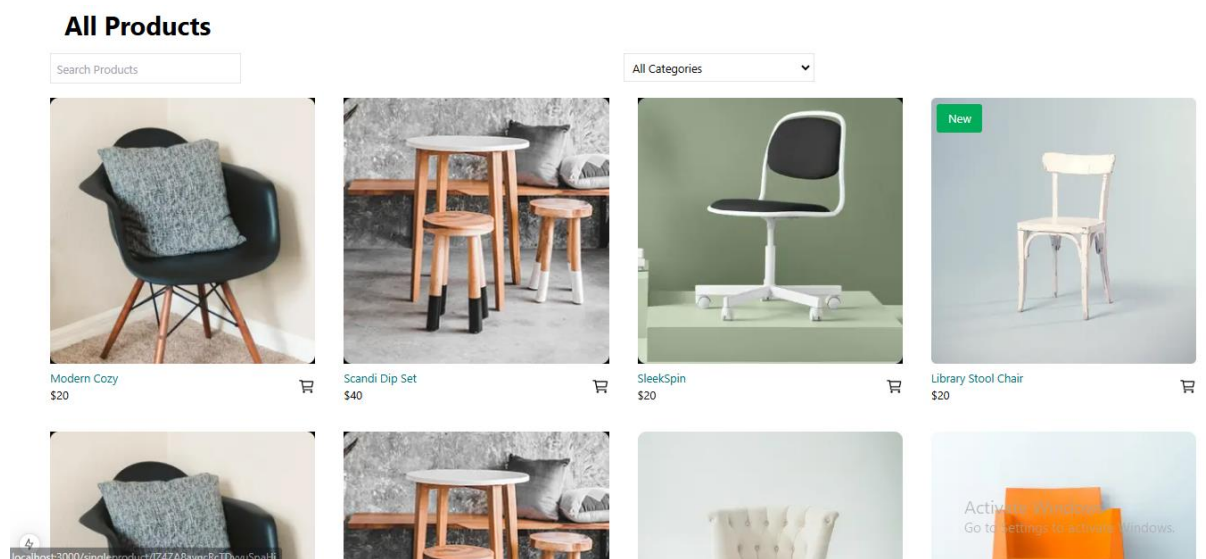
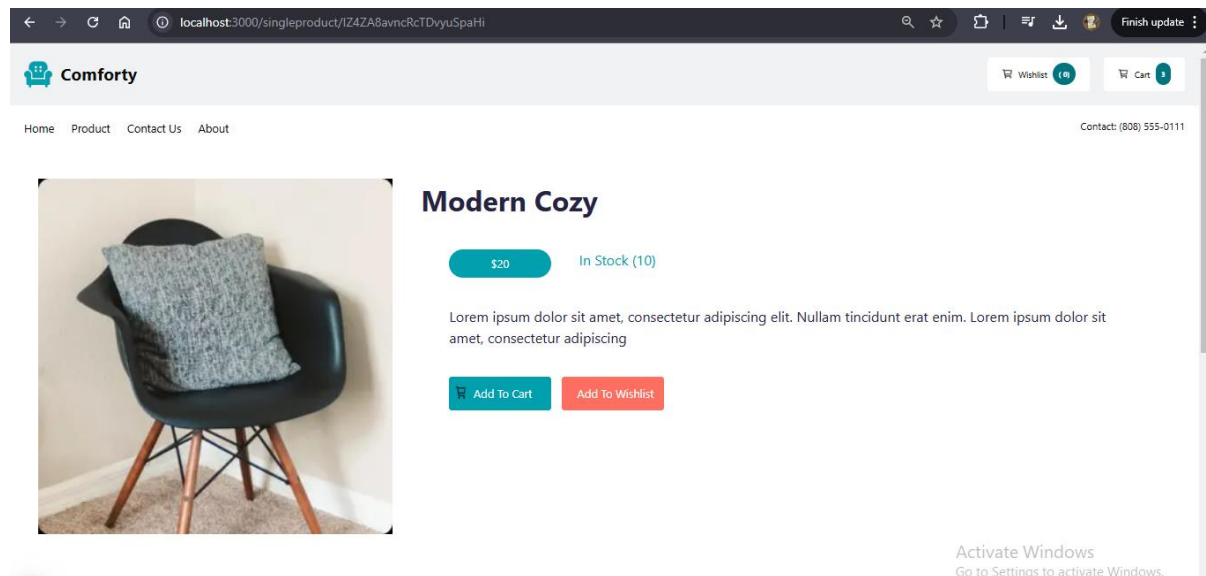


Day 4

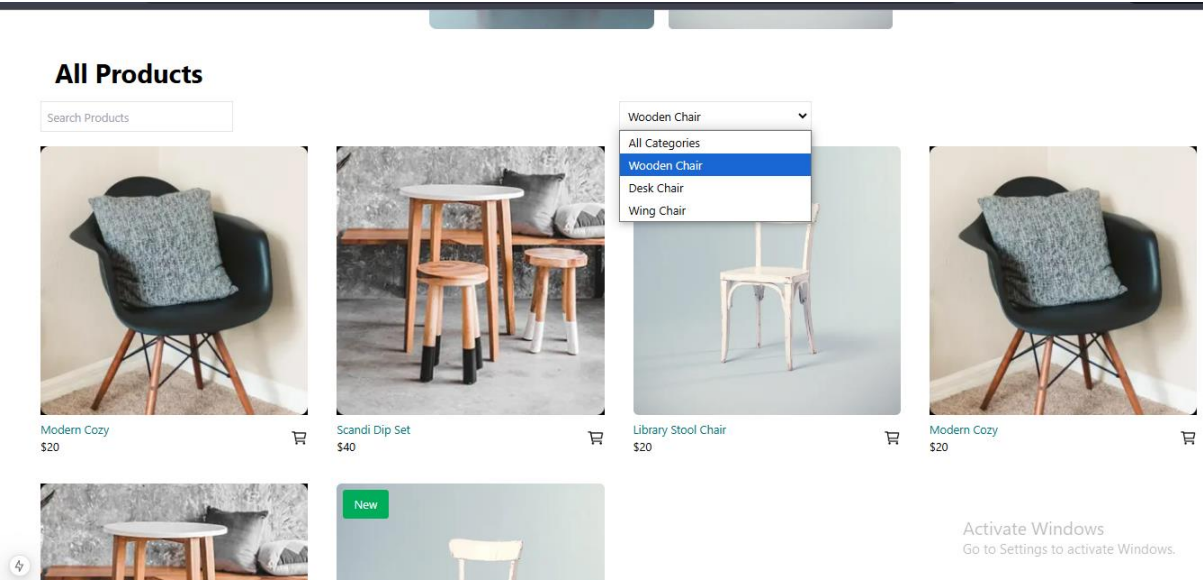
1. The product listing page with dynamic data.



2. Individual product detail pages with accurate routing and data rendering.



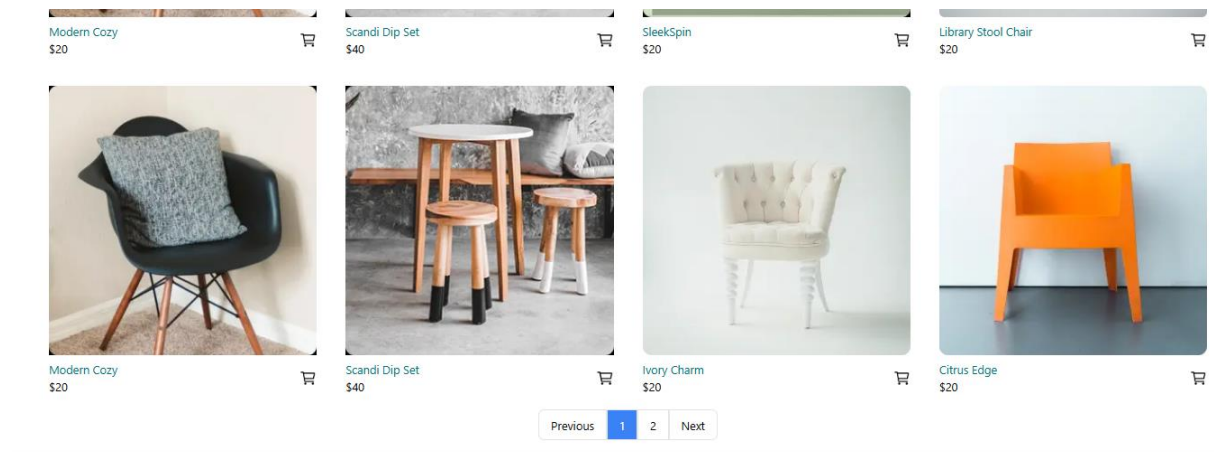
3. Working category filters.



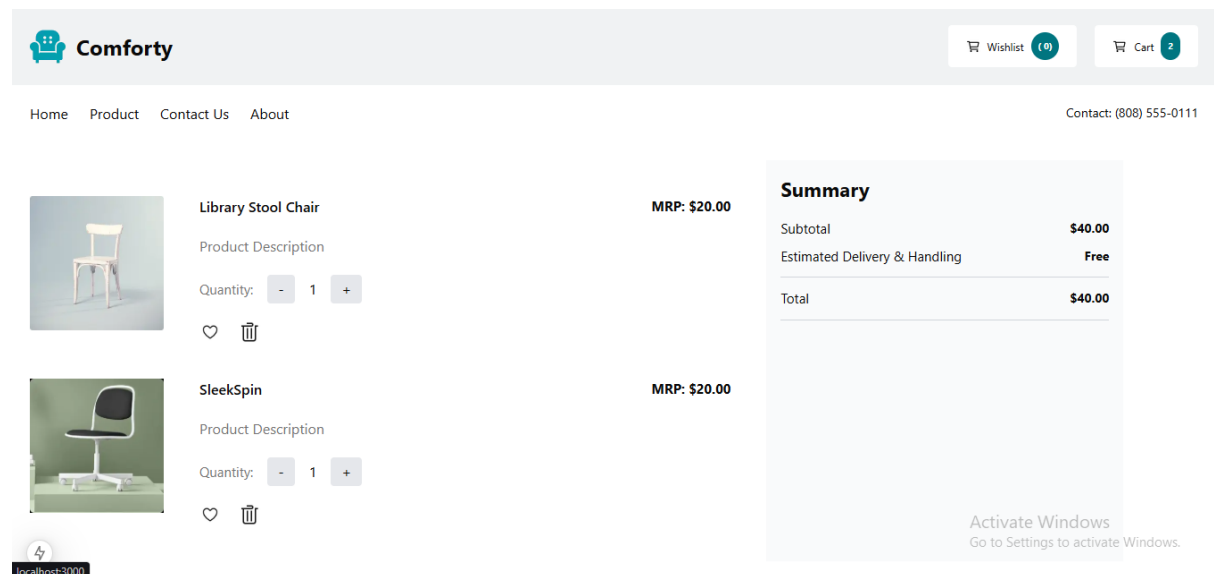
3. Working search bar



3. Working Pagination



// Add to card



❖ **Code Deliverables:**

1. **Code snippets for key components.**

For dynamic data:

```

export default function Shop({ params }: { params: { id: string } }) {
  const [product, setProduct] = useState<Product | null>(null);
  const [wishlist, setWishlist] = useState<Product[]>([]);

  const { addItem } = useShoppingCart();

  // Fetch product data
  useEffect(() => {
    const fetchProduct = async () => {
      const productData = await client.fetch(
        `*[_type == "products" && _id == $id][0]{
          title,
          price,
          description,
          "imageUrl": image.asset->url,
          inventory
        }`,
        { id: params.id }
      );
      setProduct(productData);
    };

    fetchProduct();
  });
}

```

For filter code snippet:

```

20 const AllProductsSection = () => {
28   useEffect(() => {
47     };
48
49     fetchAllProducts();
50   }, []);
51
52   // Handle the search input change
53   const handleSearch = (e: React.ChangeEvent<HTMLInputElement>) => {
54     setSearchQuery(e.target.value);
55   };
56
57   // Handle category filter change
58   const handleCategoryChange = (e: React.ChangeEvent<HTMLSelectElement>) => {
59     setSelectedCategory(e.target.value);
60   };
61
62   // Filter the products based on search query and selected category
63   const filteredProducts = allProducts.filter((product) => {
64     const matchesSearchQuery = product.title.toLowerCase().includes(searchQuery.toLowerCase());
65     const matchesCategory = selectedCategory ? product.category?.title === selectedCategory : true;
66     return matchesSearchQuery && matchesCategory;
67   });
68
69   // Get current products to display for the current page
70   const indexOfLastProduct = currentPage * productsPerPage;
71   const indexOfFirstProduct = indexOfLastProduct - productsPerPage;
72   const currentProducts = filteredProducts.slice(indexOfFirstProduct, indexOfLastProduct);
73
74   // Get unique categories from the products
75   const categories = Array.from(
76     new Set(allProducts.map((product) => product.category?.title))

```

Steps Taken to Build and Integrate Components

1. **Backend Configuration:**
 - Set up **Sanity CMS** to manage product data efficiently.
2. **Dynamic Data Retrieval:**
 - Integrated API calls in **Next.js** to fetch product data.
3. **Reusable Component Development:**
 - Built key reusable components:
 - **ProductCard:** Displays product details.
 - **ProductList:** Dynamically renders product collections.
 - **Filter:** Enables category-based filtering.
4. **Filter Integration:**
 - Implemented category filters for a seamless and intuitive user experience

Challenges and Solutions

1. **Filter Functionality:**
 - **Challenge:** Product lists didn't update dynamically with selected categories.
 - **Solution:** Optimized filter logic to handle category changes smoothly and efficiently.
2. **Cart Management:**
 - **Challenge:** Managing the "Add to Cart" feature across the app.
 - **Solution:** Leveraged **"USE-SHOPPING-CART"** for streamlined state management and consistent functionality.
3. **Responsive Design:**
 - **Challenge:** Achieving a responsive interface across different devices.
 - **Solution:** Used Tailwind Css and conducted extensive testing to ensure compatibility on all screen sizes.