

# **Hackathon Day-04**

---

*DAY 4 - BUILDING DYNAMIC*

*FRONTEND COMPONENTS*

*FOR YOUR MARKETPLACE*

---

## **Furniture E-Commerce Website**

### Dynamic Information Short Summery

This project focuses on creating a **Furniture E-Commerce Website** by leveraging Sanity CMS for dynamic data fetching and API integration. The website will feature reusable frontend components, including product listing, detail pages, a category filter, a search bar, and pagination. Dynamic routing in Next.js will enable seamless navigation, while responsive design ensures optimal user experience across devices. The goal is to build a scalable and professional marketplace platform.

More Detail 

### Sanity And Api Data Integration ( Product Listing and Pagiinations )

This section highlights the implementation of dynamic product listing and pagination using **Sanity CMS** and **API integration**.

- **Product Listing:**
  - Displays furniture items dynamically fetched from the database, including name, price, and image.
  - Cards are styled for a clean and modern layout.
- **Pagination:**
  - Implements user-friendly navigation to browse multiple pages of product data.
  - Supports seamless transition between pages using **state management** and **dynamic routing** in **Next.js**.



Replica Table

Rs. 750



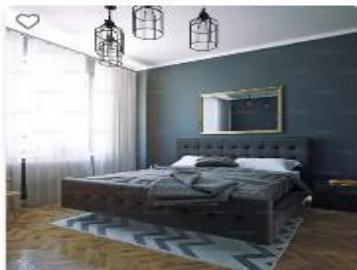
Liberty Center

Rs. 1100



Diondre Chair

Rs. 720



Luxury Flower Bed

Rs. 2500



High Quality Modern Sofa

Rs. 150



Armchair Chair Set

Rs. 850



Pink Lounge Chair

Rs. 1600



Rapson Thirty-Nine Sofa

Rs. 1300

< Previous

1

2

3

Next >



Red Bed

Rs. 320



Nautilus Lounge Chair

Rs. 1450



Armchair Chair Set

Rs. 850



Sleek Modern Table

Rs. 2000



Nautilus Lounge Chair

Rs. 1450



Alpha Table

Rs. 900



Diondre Chair

Rs. 720



Matilda Velvet Bed

Rs. 600

< Previous

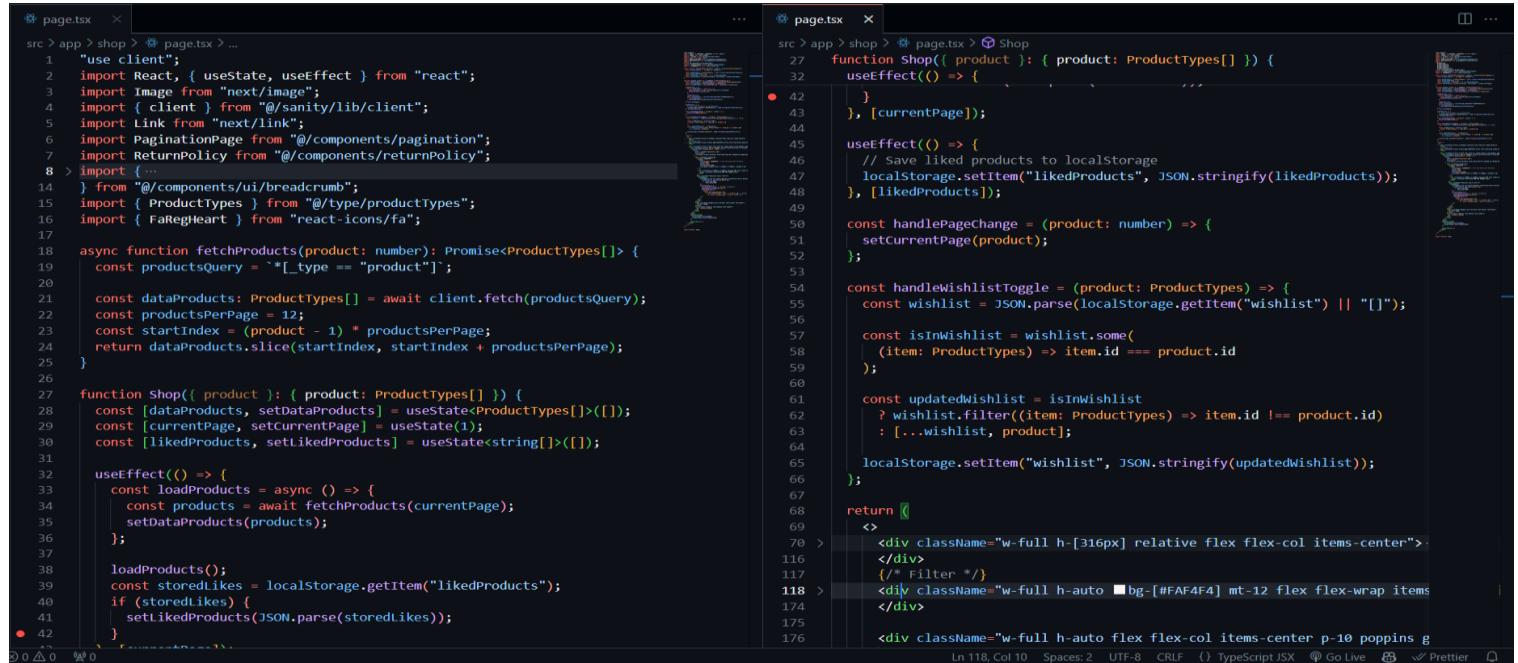
1

2

3

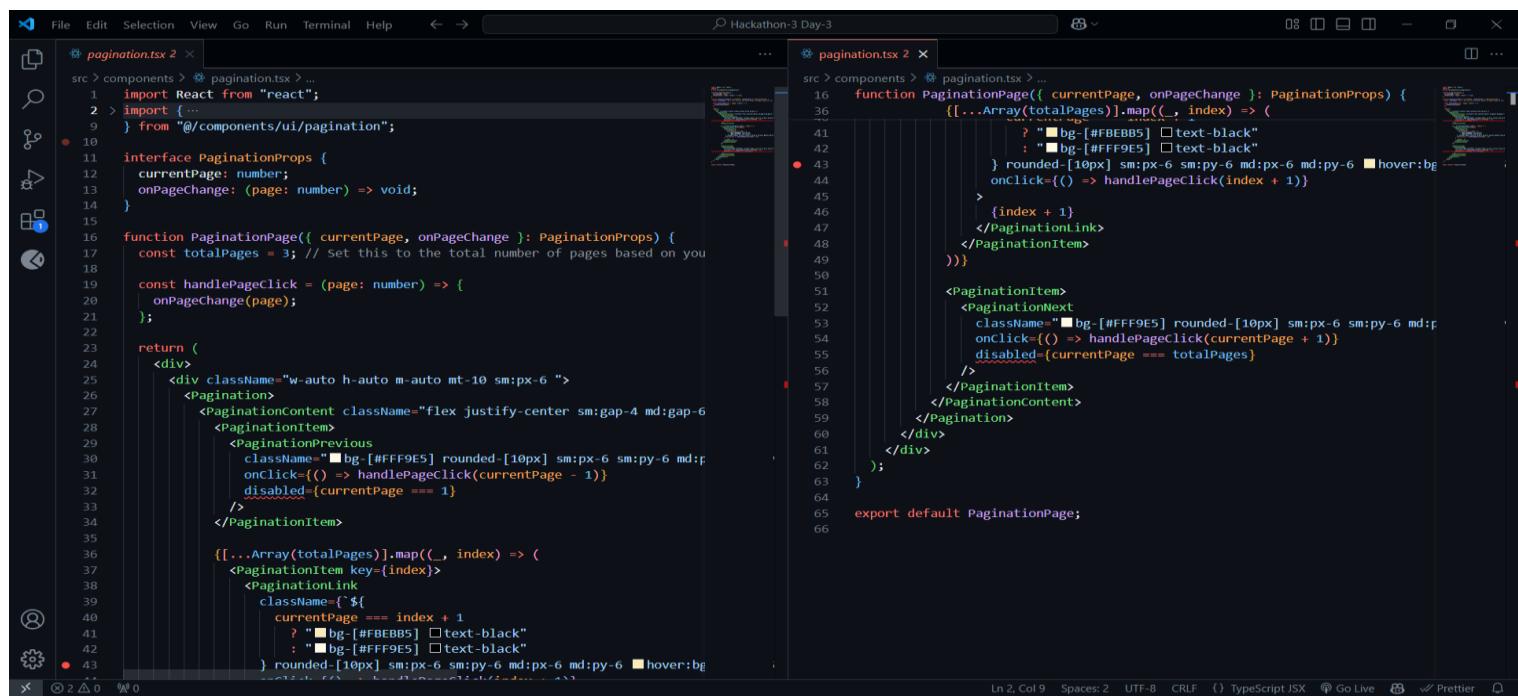
Next >

## Sanity And API Data Integration ( Product Listing ( Code ) )



```
src > app > shop > page.tsx > ...
1 "use client";
2 import React, { useState, useEffect } from "react";
3 import Image from "next/image";
4 import { client } from "@sanity/lib/client";
5 import Link from "next/link";
6 import PaginationPage from "@components/pagination";
7 import ReturnPolicy from "@components/returnPolicy";
8 > import { ... }
9 from "@components/ui/breadcrumb";
10 import { ProductTypes } from "@type/productTypes";
11 import { FaRegHeart } from "react-icons/fa";
12
13 async function fetchProducts(product: number): Promise<ProductTypes[]> {
14   const productsQuery = `*[_type == "product"]`;
15
16   const dataProducts: ProductTypes[] = await client.fetch(productsQuery);
17   const productsPerPage = 12;
18   const startIndex = (product - 1) * productsPerPage;
19   return dataProducts.slice(startIndex, startIndex + productsPerPage);
20 }
21
22 function Shop({ product }: { product: ProductTypes[] }) {
23   const [dataProducts, setDataProducts] = useState<ProductTypes[]>([]);
24   const [currentPage, setCurrentPage] = useState(1);
25   const [likedProducts, setLikedProducts] = useState<string[]>([]);
26
27   useEffect(() => {
28     const loadProducts = async () => {
29       const products = await fetchProducts(currentPage);
30       setDataProducts(products);
31     };
32
33     loadProducts();
34     const storedLikes = localStorage.getItem("likedProducts");
35     if (storedLikes) {
36       setLikedProducts(JSON.parse(storedLikes));
37     }
38   }, [currentPage]);
39
40   const handlePageChange = (product: number) => {
41     setCurrentPage(product);
42   }
43
44   const handleWishlistToggle = (product: ProductTypes) => {
45     const wishlist = JSON.parse(localStorage.getItem("wishlist") || "[]");
46
47     const isInWishlist = wishlist.some(
48       (item: ProductTypes) => item.id === product.id
49     );
50
51     const updatedWishlist = isInWishlist
52       ? wishlist.filter((item: ProductTypes) => item.id !== product.id)
53       : [...wishlist, product];
54
55     localStorage.setItem("wishlist", JSON.stringify(updatedWishlist));
56   }
57
58   return (
59     <>
60       <div className="w-full h-[316px] relative flex flex-col items-center">
61         {/* Filter */}
62         <div className="w-full h-auto bg-[#FAF4F4] mt-12 flex flex-wrap items-center">
63           <div className="w-full h-auto flex flex-col items-center p-10 poppins g" ...
64
65       </div>
66     </div>
67   );
68
69   const handlePageClick = (index: number) => {
70     setCurrentPage(index + 1);
71   }
72
73   const handlePageChange = (page: number) => {
74     setCurrentPage(page);
75   }
76
77   return (
78     <div>
79       <div className="w-auto h-auto m-auto mt-10 sm:px-6">
80         <Pagination>
81           <PaginationContent className="flex justify-center sm:gap-4 md:gap-6">
82             <PaginationItem>
83               <PaginationPrevious
84                 className="bg-[#FFF9E5] rounded-[10px] sm:px-6 sm:py-6 md:px-6 md:py-6
85                 onClick={() => handlePageClick(currentPage - 1)}
86                 disabled={currentPage === 1}>
87               </PaginationPrevious>
88             </PaginationItem>
89             {[...Array(totalPages)].map((_, index) => {
90               <PaginationItem key={index}>
91                 <PaginationLink
92                   className={`${${
93                     currentPage === index + 1
94                     ? "bg-[#FBBB5] text-black"
95                     : "bg-[#FFF9E5] text-black"
96                   } rounded-[10px] sm:px-6 sm:py-6 md:px-6 md:py-6
97                   hover:bg-[#FFF9E5] rounded-[10px] sm:px-6 sm:py-6 md:px-6 md:py-6
98                   onClick={() => handlePageClick(index + 1)}>
99                   ${index + 1}
100                  </PaginationLink>
101                </PaginationItem>
102              </div>
103            );
104          }
105        </div>
106      </div>
107    </div>
108  );
109
110  const handlePageChange = (product: number) => {
111    setCurrentPage(product);
112  }
113
114  const handleWishlistToggle = (product: ProductTypes) => {
115    const wishlist = JSON.parse(localStorage.getItem("wishlist") || "[]");
116
117    const isInWishlist = wishlist.some(
118      (item: ProductTypes) => item.id === product.id
119    );
120
121    const updatedWishlist = isInWishlist
122      ? wishlist.filter((item: ProductTypes) => item.id !== product.id)
123      : [...wishlist, product];
124
125    localStorage.setItem("wishlist", JSON.stringify(updatedWishlist));
126  }
127
128  return (
129    <>
130      <div className="w-full h-auto flex flex-col items-center p-10 poppins g" ...
131
132    </div>
133  );
134}
```

## Pagination ( Code )



```
src > components > pagination.tsx > ...
1 import React from "react";
2 > import { ... }
3 from "@components/ui/pagination";
4
5 interface PaginationProps {
6   currentPage: number;
7   onPageChange: (page: number) => void;
8 }
9
10 function PaginationPage({ currentPage, onPageChange }: PaginationProps) {
11   const totalPages = 3; // Set this to the total number of pages based on you
12
13   const handlePageClick = (page: number) => {
14     onPageChange(page);
15   };
16
17   return (
18     <div>
19       <div className="w-auto h-auto m-auto mt-10 sm:px-6">
20         <Pagination>
21           <PaginationContent className="flex justify-center sm:gap-4 md:gap-6">
22             <PaginationItem>
23               <PaginationPrevious
24                 className="bg-[#FFF9E5] rounded-[10px] sm:px-6 sm:py-6 md:px-6
25                 onClick={() => handlePageClick(currentPage - 1)}
26                 disabled={currentPage === 1}>
27               </PaginationPrevious>
28             </PaginationItem>
29             {[...Array(totalPages)].map((_, index) => {
30               <PaginationItem key={index}>
31                 <PaginationLink
32                   className={`${${
33                     currentPage === index + 1
34                     ? "bg-[#FBBB5] text-black"
35                     : "bg-[#FFF9E5] text-black"
36                   } rounded-[10px] sm:px-6 sm:py-6 md:px-6 md:py-6
37                   hover:bg-[#FFF9E5] rounded-[10px] sm:px-6 sm:py-6 md:px-6 md:py-6
38                   onClick={() => handlePageClick(index + 1)}>
39                   ${index + 1}
40                  </PaginationLink>
41                </PaginationItem>
42              </div>
43            );
44          }
45        </div>
46      </div>
47    </div>
48  );
49
50  const handlePageChange = (page: number) => {
51    onPageChange(page);
52  }
53
54  const handlePageClick = (currentPage: number) => {
55    const totalPages = 3;
56
57    const handlePageClick = (index: number) => {
58      const currentPage = index + 1;
59
60      if (currentPage === totalPages) {
61        onPageChange(totalPages);
62      } else {
63        onPageChange(currentPage);
64      }
65    }
66
67    handlePageClick(currentPage);
68  }
69
70  export default PaginationPage;
```

## Dynamic Routing ( All Listing Product and Home Product)

Dynamic routing has been implemented in the Furniture E-Commerce Website using Next.js. This enables seamless navigation for individual product details, including images, descriptions, sizes, and colors. The route structure allows fetching product data dynamically for both the home page and the product listing pages, ensuring an interactive and user-friendly experience.

A screenshot of a web browser showing a product detail page for a 'Replica Table'. The URL in the address bar is 'localhost:3000/product/11'. The page includes a navigation bar with links to Home, Shop, About, and Contact, and a header with user icons. On the left, there is a vertical stack of five smaller images of the same table from different angles. The main image shows a white round table with three legs, a black alarm clock on top, and a small potted plant on the side, positioned against a rustic wooden wall. To the right of the image, the product title 'Replica Table' is displayed in bold, followed by the price 'Rs. 750.00'. Below the price is a rating of 5 stars with the text '5 Customer Review'. A descriptive text reads 'Classic wishbone chair with a dark walnut frame and cord seat.'. Size and color selection options are shown with buttons for 'L', 'XL', 'XS' (size) and blue, black, and gold (color). A quantity selector with '+' and '-' buttons is set to '1', and a large 'Add to Cart' button is at the bottom.

A screenshot of a web browser showing a product listing page for a 'Hans Wegner Style Three-Legged Shell Chair'. The URL in the address bar is 'localhost:3000/'. The page includes a navigation bar with links to Home, Shop, About, and Contact, and a header with user icons. On the left, the product title 'Hans Wegner Style Three-Legged Shell Chair' is displayed in large, bold, black font. Below the title is a 'Shop Now' button. On the right, there is a large, high-quality image of the chair, which is made of light-colored wood with a white upholstered shell and three curved legs.

## Dynamic Routing ( Product ( Code ) )

```
src > app > product > [id] > page.tsx > ...
1  "use client";
2
3  import React, { useState, useEffect } from "react";
4  import { useParams } from "next/navigation";
5  import Image from "next/image";
6  import { client } from "@sanity/lib/client";
7  import { FaFacebook, FaLinkedIn, FaStar } from "react-icons/fa";
8  import { AiFillTwitterCircle } from "react-icons/ai";
9  import AddToCardsSlider from "@components/AddtoCardSlider";
10 import RelatedProducts from "@components/relatedProducts";
11 import ProductDetailNested from "@components/ProductDetailNested";
12
13 const Product = () => {
14   const params = useParams();
15   const id = params?.id;
16
17   const [dataProducts, setDataProducts] = useState(null);
18   const [error, setError] = useState(null);
19   const [activeTab, setActiveTab] = useState<string>("");
20   const [count, setCount] = useState<number>(1);
21   const [loading, setLoading] = useState(true);
22
23   function increment() {
24     setCount(count + 1);
25   }
26
27   function decrement() {
28     if (count > 1) {
29       setCount(count - 1);
30     }
31   }
32
33   function handleClick(tab: string) {
34     setActiveTab(tab);
35   }
36
37   useEffect(() => {
38     const query = `[_type == "product" && id == ${id}[0]]`;
39
40     const fetchData = async () => {
41       setLoading(true);
42       try {
43         console.log("Fetching product with custom ID:", id);
44         const result = await client.fetch(query, { id });
45
46         if (result) {
47           console.log("Fetched Product:", result);
48           setDataProducts(result);
49         } else {
50           setError("Product not found.");
51         }
52       } catch (err) {
53         setError("Failed to load product.");
54       } finally {
55         setLoading(false);
56       }
57     };
58
59     if (id) {
60       fetchData();
61     }
62   }, [id]);
63
64   if (loading) return <div>Loading...</div>;
65   if (error) return <div>(error)</div>;
66
67   return (
68     <>
69       <div className="w-full h-auto p-4 sm:p-10">
70         <div className="flex flex-col lg:flex-row justify-evenly">
71           {/* Images Section */}
72           <div className="w-full lg:w-[553px] h-auto lg:h-[500px] mb-6 lg:mb-0">
73             <div className="flex flex-col lg:flex-row gap-4 lg:gap-0">
```

```
src > app > product > [id] > @ page.tsx > ...
13 const Product = () => {
  14   <div className="w-full lg:w-[553px] h-auto lg:h-[500px] mb-6 lg:mb-0">
  15     <div className="flex flex-col lg:flex-row gap-4 lg:gap-8">
  16       <div className="w-full lg:w-[76px] flex flex-row lg:flex-col gap-4 lg:gap-8">
  17         <div className="w-auto h-auto bg-[#FFF 9E5] rounded-lg cursor-pointer">
  18           <Image
  19             src={dataProducts.imagePath}
  20             alt="Thumbnail"
  21             width={76}
  22             height={80}
  23             className="h-[80px] rounded-lg object-cover"
  24           />
  25           <Image
  26             src={dataProducts.imagePath}
  27             alt="Thumbnail"
  28             width={76}
  29             height={80}
  30             className="h-[80px] rounded-lg object-cover"
  31           />
  32           <Image
  33             src={dataProducts.imagePath}
  34             alt="Thumbnail"
  35             width={76}
  36             height={80}
  37             className="h-[80px] rounded-lg object-cover"
  38           />
  39           <Image
  40             src={dataProducts.imagePath}
  41             alt="Thumbnail"
  42             width={76}
  43             height={80}
  44             className="h-[80px] rounded-lg object-cover"
  45           />
  46         </div>
  47       </div>
  48     </div>
  49   <div className="w-full lg:w-[481px] h-[300px] lg:h-[500px] rounded-lg">
  50     <Image
  51       src={dataProducts.imagePath}
  52       alt="Product Image"
  53       width={1000}
  54       height={1000}
  55       className="rounded-lg object-cover"
  56     />
  57   </div>
  58   <div className="text-[18px] lg:text-[24px] font-medium text-[#9F9F9F]">
  59     Rs. {dataProducts.price}.00
  60   </div>
  61   <div className="flex items-center gap-3 lg:gap-5">
  62     <div className="flex gap-1 lg:gap-2 items-center text-[#FFDA5B]">
  63       <Fastar />
  64       <Fastar />
  65       <Fastar />
  66       <Fastar />
  67       <Fastar />
  68     </div>
  69     <div className="text-[13px] text-[#9F9F9F]">
  70       5 Customer Review
  71     </div>
  72   </div>
  73   <p className="text-[13px]">>{dataProducts.description}</p>
  74   <div className="flex flex-col gap-2">
  75     <span className="text-[#9F9F9F] text-[14px]">>Size</span>
  76     <div className="flex gap-4 text-[13px]">
  77       {"L", "XL", "XXL", "map(tab)"} => (
  78     </div>
  79   </div>
  80 
```

The image shows two side-by-side code editors, likely from the VS Code IDE, displaying the same file 'page.tsx' at different stages of development. The left editor shows the initial state of the component, while the right editor shows the component after several changes, including the addition of a 'Product' prop and various styling and logic improvements.

```
src > app > product > [id] > page.tsx > ...
```

```
const Product = () => {
  const tabs = ["L", "X", "XS"].map((tab) => (
    <p key={tab}>
      className={`bg-[#FAFAF4] w-[30px] h-[30px] flex justify-center ${activeTab === tab ? "bg-[#BEBBB5]" : ""}`}
    </p>
    onClick={() => handleClick(tab)}
  ))
  {tab}
</p>
)}
```

```
const Product = () => {
  const [count, setCount] = useState(1)
  const [activeTab, setActiveTab] = useState("L")
  const handleClick = (tab) => setActiveTab(tab)
  const increment = () => setCount(count + 1)
  const decrement = () => setCount(count - 1)
  const handleAddToCart = () => {
    alert(`Added ${count} ${activeTab} to cart`)
  }
  return (
    <div>
      <h1>Product Page</h1>
      <h2>Tabs</h2>
      <div>
        {tabs}
      </div>
      <hr />
      <h2>Add To Cart</h2>
      <div>
        <input type="text" value="Sofa" />
        <button onClick={handleAddToCart}>Add To Cart</button>
      </div>
      <hr />
      <h2>Product Details</h2>
      <div>
        <h3>Product Name:</h3>
        <input type="text" value="Sofa" />
        <h3>Product Description:</h3>
        <input type="text" value="A comfortable sofa for your living room." />
        <h3>Product Price:</h3>
        <input type="text" value="1000" />
        <h3>Product Category:</h3>
        <input type="text" value="Home" />
        <h3>Product Tags:</h3>
        <input type="text" value="sofa, chair, home, shop" />
        <h3>Product Share:</h3>
        <div>
          <a href="#">Facebook</a>
          <a href="#">LinkedIn</a>
          <a href="#">Twitter</a>
        </div>
      </div>
    </div>
  )
}
```

## Product ( Wishlist's )

1. Users can add products to the Wishlist by clicking the heart icon on any product.
  2. The selected products are saved on a dedicated **Wishlist Page**.
  3. Users can view and manage their favorite items from the Wishlist.

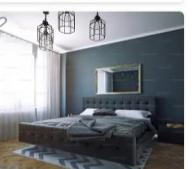
- Wishlist items can be added to the cart directly for easy purchasing.
- This feature enhances the shopping experience by allowing users to save and revisit their favorite products.

**Shop**

Home > Shop

Show 16 Sort by Default

Filter | Showing 1-16 of 32 results

			
Chair Wibe Rs. 1200	Cozy Sofa Rs. 520	White Bed Rs. 120	Blue Bed Rs. 780
			
Replica Table	Liberty Center	Diondre Chair	Luxury Flower Bed

localhost:3000/wishList

Wishlist

			
Chair Wibe Rs. 1200 <a href="#">Add to Cart</a> <a href="#">Remove</a>	White Bed Rs. 120 <a href="#">Add to Cart</a> <a href="#">Remove</a>	Blue Bed Rs. 780 <a href="#">Add to Cart</a> <a href="#">Remove</a>	Replica Table Rs. 750 <a href="#">Add to Cart</a> <a href="#">Remove</a>
			
Diondre Chair Rs. 720			

## Product ( Wishlist's ( WishList Component Code ) )

```
File Edit Selection View Go Run Terminal Help ← → Hackathon-3 Day-3 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
```

```
import React, { useState, useEffect } from "react";
import { client } from "@sanity/lib/client"; // Import Sanity client
import { ProductTypes } from "@type/productTypes";
import Image from "next/image";
import AddToCardSlider from "@components/AddToCardSlider";
import { FaRegHeart } from "react-icons/fa";

function Wishlist() {
  const [wishlist, setWishlist] = useState<ProductTypes[]>([]);
  const [likedProducts, setLikedProducts] = useState<string[]>([ ]);

  // Fetch wishlist from localStorage and update state on initial load
  useEffect(() => {
    const fetchWishlist = async () => {
      try {
        const storedWishlist = localStorage.getItem("wishlist");
        if (storedWishlist) {
          const wishlistIds = JSON.parse(storedWishlist).map(
            (item: ProductTypes) => item.id
          );

          if (wishlistIds.length > 0) {
            // Query to fetch products based on IDs
            const query = `*[_type == "product" & id in ${wishlistIds}]`;
            const products = await client.fetch(query, { wishlistIds });

            // Remove duplicates by ensuring unique IDs
            const uniqueProducts = products.filter(
              (value, index, self) =>
                index === self.findIndex((t) => t.id === value.id)
            );

            setWishlist(uniqueProducts); // Set wishlist state with unique products
          }
        }
      } catch (error) {
        console.log(error);
      }
    };
    fetchWishlist();
  }, []);

  const fetchWishlist = () => {
    const wishlistIds = wishlist.map((item) => item.id);
    if (wishlistIds.length > 0) {
      localStorage.setItem("wishlist", JSON.stringify(wishlistIds));
    }
  };

  // Save wishlist IDs to localStorage whenever it changes
  useEffect(() => {
    const wishlistIds = wishlist.map((item) => item.id);
    if (wishlistIds.length > 0) {
      localStorage.setItem("wishlist", JSON.stringify(wishlistIds));
    }
  }, [wishlist]); // Run whenever wishlist changes

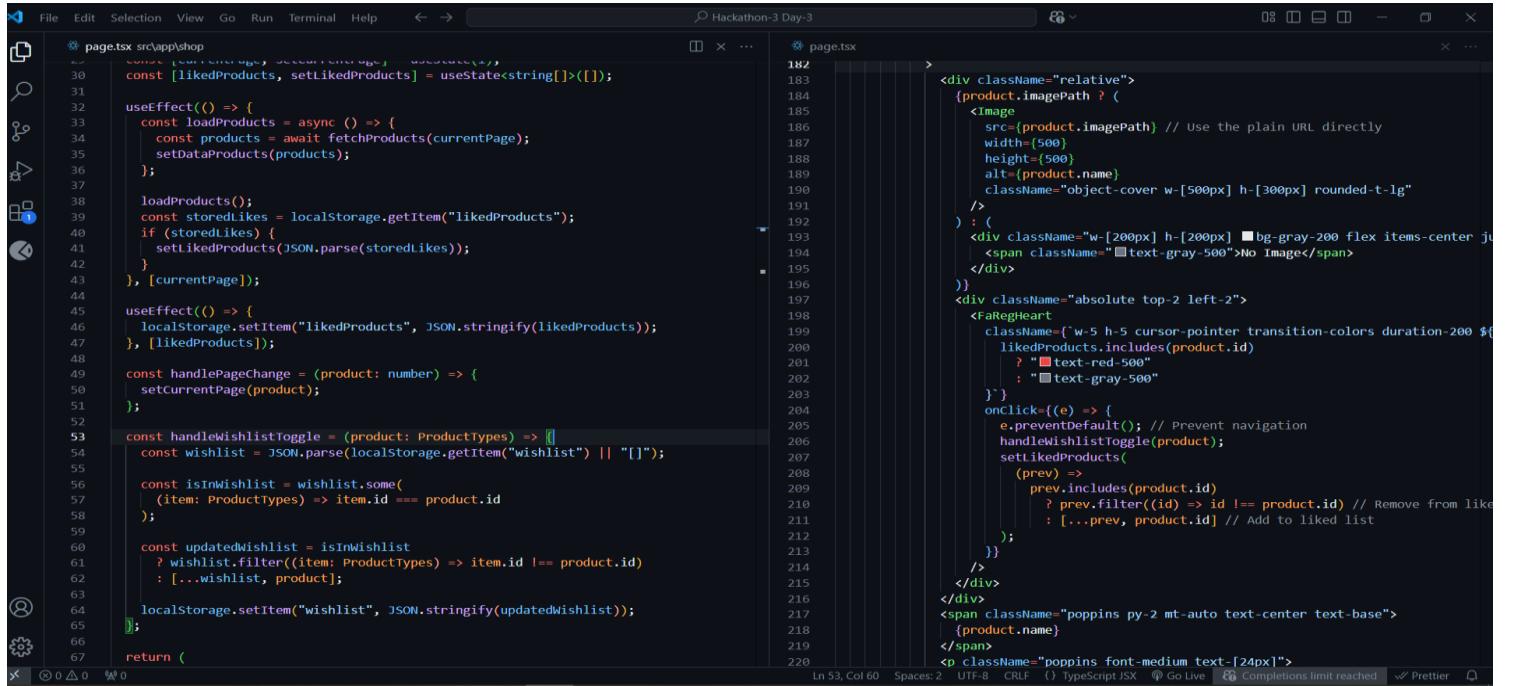
  const removeFromWishlist = (id: string) => {
    setWishlist((prev) => {
      const updatedWishlist = prev.filter((product) => product.id !== id);
      // Save the updated wishlist to localStorage
      localStorage.setItem(
        "wishlist",
        JSON.stringify(updatedWishlist.map((item) => item.id))
      );
      return updatedWishlist; // Return updated wishlist
    });
  };

  const handleWishlistToggle = (product: ProductTypes) => {
    const isInWishlist = wishlist.some((item) => item.id === product.id);

    if (isInWishlist) {
      removeFromWishlist(product.id);
    } else {
      setWishlist((prev) => [...prev, product]);
    }
  };
}

export default Wishlist;
```

## Product ( Wishlist's ( Shop Component Code ))



```
File Edit Selection View Go Run Terminal Help < > Hackathon-3 Day-3
page.tsx src/app/shop
  const [currentPage, setCurrentPage] = useState<string>('1');
  const [likedProducts, setLikedProducts] = useState<string[]>([]);

  useEffect(() => {
    const loadProducts = async () => {
      const products = await fetchProducts(currentPage);
      setDataProducts(products);
    };
    loadProducts();
    const storedLikes = localStorage.getItem("likedProducts");
    if (storedLikes) {
      setLikedProducts(JSON.parse(storedLikes));
    }
  }, [currentPage]);
  useEffect(() => {
    localStorage.setItem("likedProducts", JSON.stringify(likedProducts));
  }, [likedProducts]);
  const handlePageChange = (product: number) => {
    setCurrentPage(product);
  };
  const handleWishlistToggle = (product: ProductTypes) => {
    const wishlist = JSON.parse(localStorage.getItem("wishlist") || "[]");
    const isInWishlist = wishlist.some(
      (item: ProductTypes) => item.id === product.id
    );
    const updatedWishlist = isInWishlist
      ? wishlist.filter((item: ProductTypes) => item.id !== product.id)
      : [...wishlist, product];
    localStorage.setItem("wishlist", JSON.stringify(updatedWishlist));
  };
  return (
    <div>
      <Image src={product.imagePath} alt={product.name}>
        <span>No Image</span>
      </Image>
      <div>
        <div>
          <FaRegHeart onClick={(e) => {
            e.preventDefault();
            handleWishlistToggle(product);
            setLikedProducts(
              (prev) =>
                prev.includes(product.id)
                  ? prev.filter((id) => id !== product.id)
                  : [...prev, product.id]
            );
          }}></FaRegHeart>
        </div>
        <span>{product.name}</span>
        <p>{product.description}</p>
        <div>
          <span>${product.price}</span>
          <span>Add to Cart</span>
        </div>
      </div>
    </div>
  );
}

// Fetch products from a database or API
const fetchProducts = async (page: string) => {
  // Implementation
}
const setDataProducts = (products: ProductTypes[]) => {
  // Implementation
}
```

## Product Add To Cart and Check-Out ( Functionality ) Summary

This functionality provides users with a seamless and interactive shopping experience, covering the entire process from adding products to the cart to completing the purchase.

### Detailed Summary:

#### 1. Add to Cart Functionality:

- Users can add any product to the cart directly from the product listing or detail pages by specifying the quantity.
- The cart dynamically updates to show the total quantity of items, the total number of unique products, and the total price in real-time.

#### 2. Cart Overview and Management:

- A **Cart Preview Section** is available on all pages for quick access to cart details.
- Users can view all the added products in the **Cart Page**, where the following options are available:
  - **Increase or Decrease Quantity**: Users can adjust the product quantity directly in the cart.
  - **Remove Items**: Any unwanted item can be removed from the cart with a single click.

- **Dynamic Price Update:** The total amount adjusts automatically as users modify the quantity or remove items.

### 3. Checkout Process:

- Once satisfied with the cart items, users can proceed by clicking the **Checkout Button**.
- During checkout, users are guided through the process of entering delivery details, reviewing the order summary, and selecting payment methods.

### 4. Order Confirmation:

- After successful checkout, an **Order Confirmation Page** displays a summary of the purchased items, total amount, and an estimated delivery date.

### 5. Additional Features:

- Responsive Design ensures smooth functionality across devices, whether mobile, tablet, or desktop.
- Visual Feedback is provided with notifications (e.g., "Item added to cart" or "Quantity updated").
- Secure Checkout integrates with reliable payment methods for a trustworthy shopping experience.

### Point-Wise Summary:

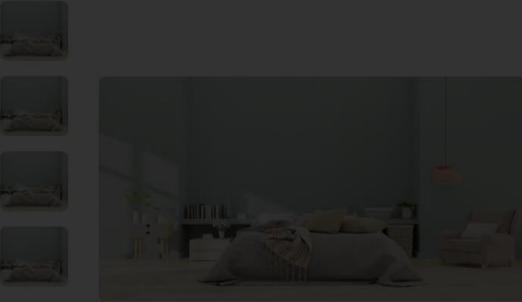
1. Users can add products to the cart and view total items, quantities, and price dynamically.
2. A cart page allows users to modify the quantity of items or remove them entirely.
3. Total price updates in real-time based on changes in quantity or product selection.
4. Users can proceed to checkout to complete their purchase with delivery and payment options.
5. The order confirmation page summarizes all details and provides tracking or delivery information.
6. Responsive and user-friendly design enhances the shopping experience across all devices.
7. Notifications and alerts improve the interactivity and reliability of the shopping process.

Home      Shop      About      Contact

 Solid Bed  
Rs. 100.00  
5 Customer Review  
Durable and lightweight plastic chair for everyday use.  
Size: L, XL, XS  
Color: Blue, Black, Gold  
+ 1 - Add to Cart

localhost:3000/product/17

Home Shop About Contact



**Solid Bed**

Rs. 100.00

5 Customer Review

Durable and lightweight plastic chair for everyday use.

Size: L XL XS

Color: Blue, Black, Green

+ 1 - Add to Cart

SKU: SS0017 Category: Bed

**Shopping Cart**

Product	Quantity	Price
Solid Bed	1 X	Rs. 100

Subtotal: Rs. 100

[View Cart](#) [Checkout](#)

localhost:3000/product/21

Home Shop About Contact



**Luxury Flower Bed**

Rs. 2500.00

5 Customer Review

A luxurious shell-shaped chair with gold brass metal legs.

Size: L XL XS

Color: Blue, Black, Green

+ 3 - Add to Cart

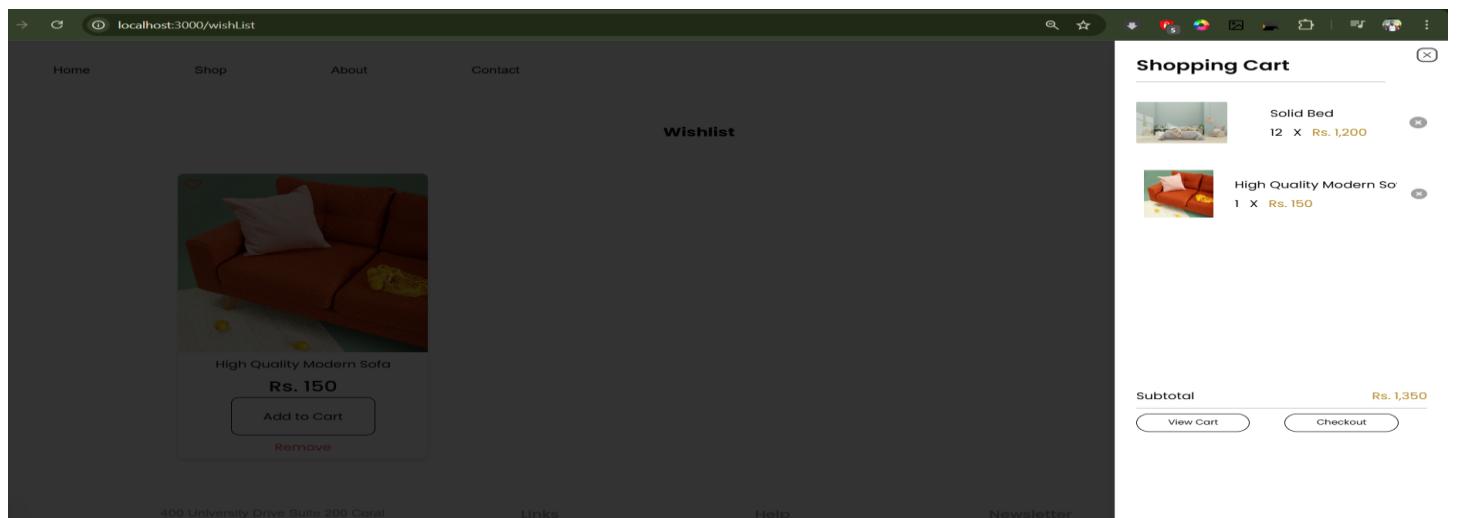
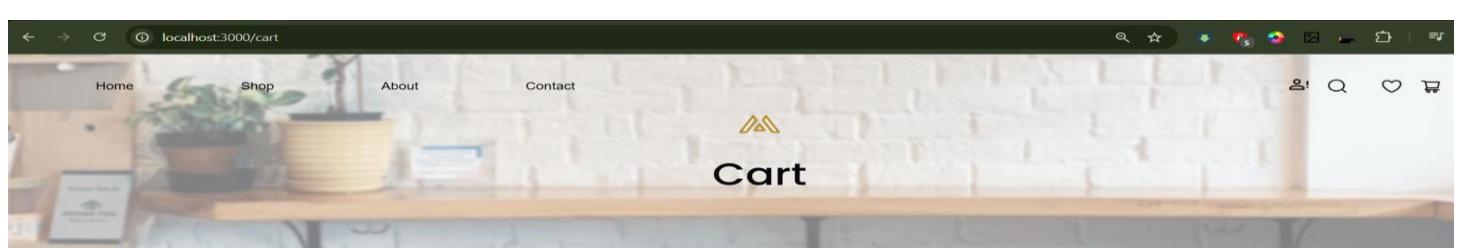
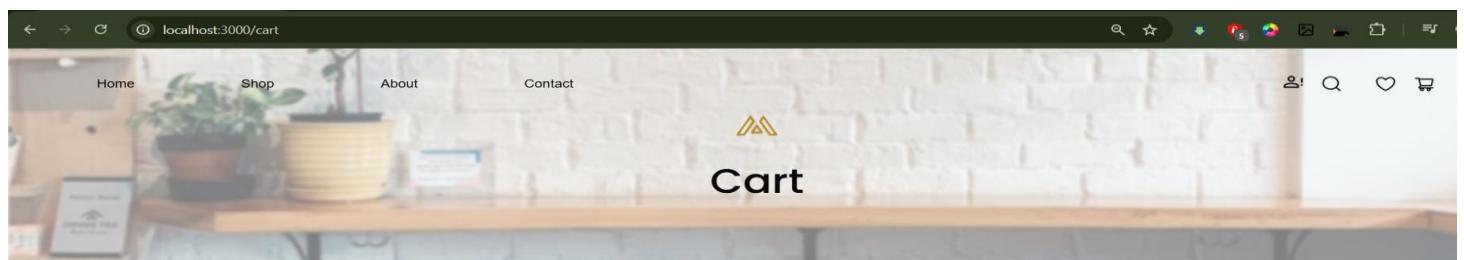
SKU: SS0021 Category: Bed

**Shopping Cart**

Product	Quantity	Price
Solid Bed	1 X	Rs. 100
Luxury Flower Bed	3 X	Rs. 7,500

Subtotal: Rs. 7,600

[View Cart](#) [Checkout](#)



**Cart**

Product	Price	Quantity	Subtotal
Solid Bed	Rs. 100	12	Rs. 1,200
High Quality Modern Sofa	Rs. 150	1	Rs. 150

**Cart Totals**

Subtotal	Rs. 1,350
Total	Rs. 1,350

**Check Out**

**Checkout**

**Billing details**

First Name \_\_\_\_\_ Last Name \_\_\_\_\_  
Company Name (Optional) \_\_\_\_\_  
Country / Region \_\_\_\_\_  
Street address \_\_\_\_\_

Product	Subtotal
Solid Bed	Rs. 1,200
High Quality Modern Sofa	Rs. 150
Total	<b>Rs. 1,350</b>

Direct Bank Transfer  
Make your payment directly into our bank account.  
 Cash On Delivery

**Place order**

**Checkout**

**Billing details**

First Name \_\_\_\_\_ Last Name \_\_\_\_\_  
Company Name (Optional) \_\_\_\_\_  
Country / Region \_\_\_\_\_  
Street address \_\_\_\_\_

Product	Subtotal
Solid Bed	Rs. 1,200
High Quality Modern Sofa	Rs. 150
Total	<b>Rs. 1,350</b>

Direct Bank Transfer  
 Cash On Delivery  
Pay with cash when your order is delivered.

**Place order**

# Product Add To Cart and Check-Out ( Functionality ( Code ) )

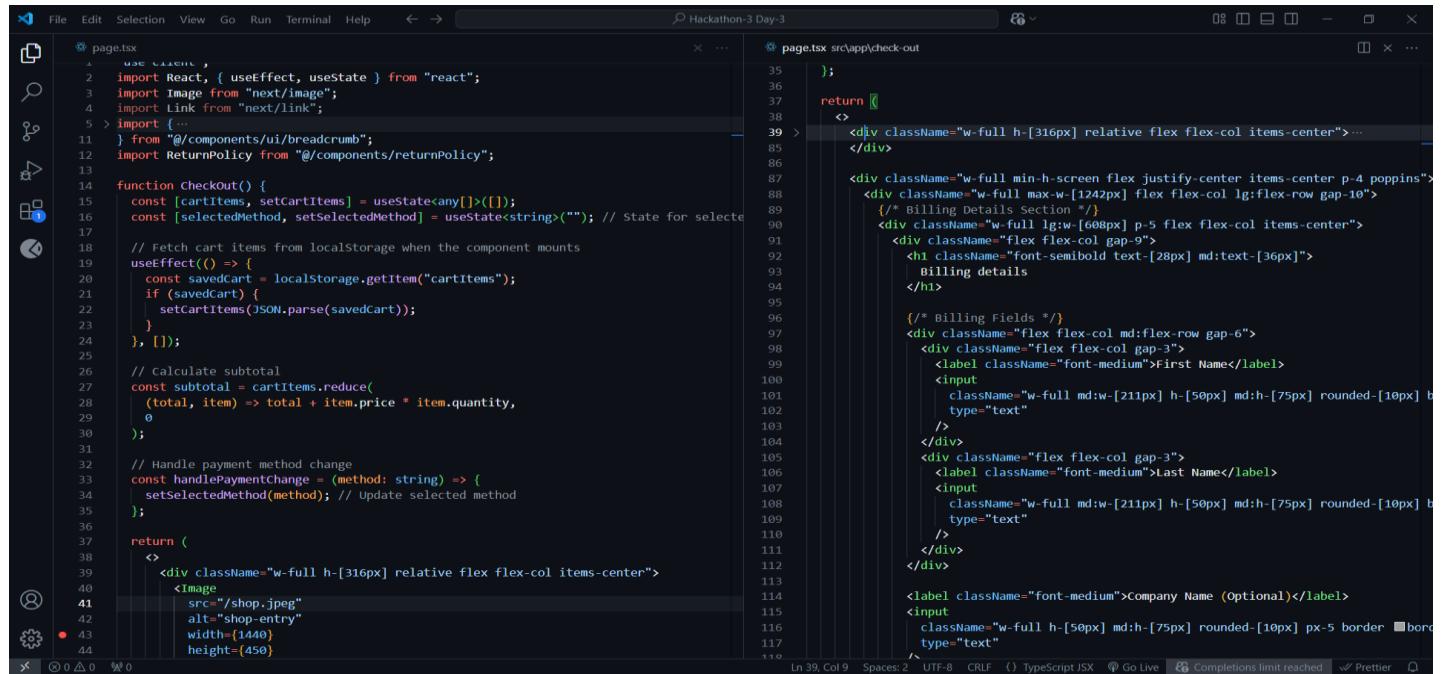
## Add to Cart Slider.

```
File Edit Selection View Go Run Terminal Help < > Hackathon-3 Day-3 AddToCardSlider.tsx AddToCardSlider.tsx
1 import React, { useState, useEffect } from "react";
2 import Image from "next/image";
3 > import { ... }
4   from "@components/ui/sheet";
5 import Link from "next/link";
6
7 interface AddToCardSliderProps {
8   name: string;
9   price: number;
10  image: string;
11  quantity: number;
12}
13
14 function AddToCardSlider({
15   name,
16   price,
17   image,
18   quantity,
19 }: AddToCardSliderProps) {
20   const [cartItems, setCartItems] = useState<AddToCardSliderProps[]>([]);
21
22   // Load cart items from localStorage when the component mounts
23   useEffect(() => {
24     const savedCart = localStorage.getItem("cartItems");
25     if (savedCart) {
26       setCartItems(JSON.parse(savedCart));
27     }
28   }, []);
29
30   // Save cart items to localStorage whenever cartItems changes
31   useEffect(() => {
32     if (cartItems.length > 0) {
33       localStorage.setItem("cartItems", JSON.stringify(cartItems));
34     }
35   }, [cartItems]);
36
37   // Handle delete product
38   function handleDeleteProduct(indexToRemove: number) {
39     setCartItems((prevItems) => {
40       const updatedCart = [...prevItems];
41       updatedCart.splice(indexToRemove, 1);
42       setCartItems(updatedCart);
43       localStorage.setItem("cartItems", JSON.stringify(updatedCart));
44     });
45   }
46
47   // Handle add to cart with quantity update
48   function handleAddToCart() {
49     const itemIndex = cartItems.findIndex(
50       (item) => item.name === name && item.image === image
51     );
52
53     if (itemIndex === -1) {
54       // If item already exists, update the quantity
55       setCartItems((prevItems) => {
56         const updatedItems = [...prevItems];
57         updatedItems[itemIndex].quantity += quantity; // Add quantity to existing product
58         localStorage.setItem("cartItems", JSON.stringify(updatedItems)); // Update localstorage
59         return updatedItems;
60       });
61     } else {
62       // Add new item to cart
63       setCartItems((prevItems) => {
64         const updatedItems = [...prevItems, { name, price, image, quantity }];
65         localStorage.setItem("cartItems", JSON.stringify(updatedItems)); // Update localstorage
66         return updatedItems;
67       });
68     }
69
70   }
71
72   // Dynamically calculate subtotal
73   const subtotal = cartItems.reduce(
74     (total, item) => total + item.price * item.quantity,
75   );
76
77   // Calculate total (you can add additional logic for taxes, shipping, etc.)
78   const total = subtotal;
79
80   return (
81     <div>
82       <Image alt="shop-entry" width={1440} height={1450} className="object-cover w-full h-[316px] relative flex flex-col items-center">
83         <div>
84           <h1>Cart</h1>
85           <table border="1" style={{ width: "100%", border-collapse: "collapse" }}>
86             <thead>
87               <tr>
88                 <th>Item</th>
89                 <th>Quantity</th>
90                 <th>Price</th>
91               </tr>
92             </thead>
93             <tbody>
94               <tr>
95                 <td>Product A</td>
96                 <td>2</td>
97                 <td>$100</td>
98               </tr>
99             </tbody>
100            <tfoot>
101              <tr>
102                <td colspan="2" style="text-align: right; padding-right: 10px;">Subtotal:</td>
103                <td style="font-weight: bold;">$200</td>
104              </tr>
105            </tfoot>
106          </table>
107        </div>
108      </Image>
109    </div>
110  );
111}
```

## Cart.

```
File Edit Selection View Go Run Terminal Help < > Hackathon-3 Day-3 page.tsx page.tsx
1 "use client";
2 import React, { useEffect, useState } from "react";
3 import Image from "next/image";
4 import Link from "next/link";
5
6 function Cart() {
7   const [cartItems, setCartItems] = useState<any[]>([]);
8
9   // Fetch cart items from localStorage when the component mounts
10  useEffect(() => {
11    const savedCart = localStorage.getItem("cartItems");
12    if (savedCart) {
13      setCartItems(JSON.parse(savedCart));
14    }
15  }, []);
16
17  // Handle deleting a product from the cart
18  const handleDeleteProduct = (indexToRemove: number) => {
19    const updatedCart = [...cartItems];
20    updatedCart.splice(indexToRemove, 1);
21    setCartItems(updatedCart);
22    localStorage.setItem("cartItems", JSON.stringify(updatedCart));
23  };
24
25  // Handle updating the quantity of an item
26  const handleQuantityChange = (index: number, newQuantity: number) => {
27    const updatedCart = [...cartItems];
28    updatedCart[index].quantity = newQuantity;
29    setCartItems(updatedCart);
30    localStorage.setItem("cartItems", JSON.stringify(updatedCart));
31  };
32
33  // Calculate subtotal
34  const subtotal = cartItems.reduce(
35    (total, item) => total + item.price * item.quantity,
36    0
37  );
38
39  // Calculate total (you can add additional logic for taxes, shipping, etc.)
40  const total = subtotal;
41
42  return (
43    <div>
44      <div>
45        <Image alt="shop-logo.png" width={1000} height={1000} className="absolute top-1/2 transform -translate-y-1/2 flex flex-col justify-center items-center">
46          <div>
47            <h1>Cart</h1>
48            <table border="1" style={{ width: "100%", border-collapse: "collapse" }}>
49              <thead>
50                <tr>
51                  <th>Item</th>
52                  <th>Quantity</th>
53                  <th>Price</th>
54                </tr>
55              </thead>
56              <tbody>
57                <tr>
58                  <td>Product A</td>
59                  <td>2</td>
60                  <td>$100</td>
61                </tr>
62              </tbody>
63              <tfoot>
64                <tr>
65                  <td colspan="2" style="text-align: right; padding-right: 10px;">Subtotal:</td>
66                  <td style="font-weight: bold;">$200</td>
67                </tr>
68              </tfoot>
69            </table>
70          </div>
71        </Image>
72      </div>
73    </div>
74  );
75}
```

## Checkout.

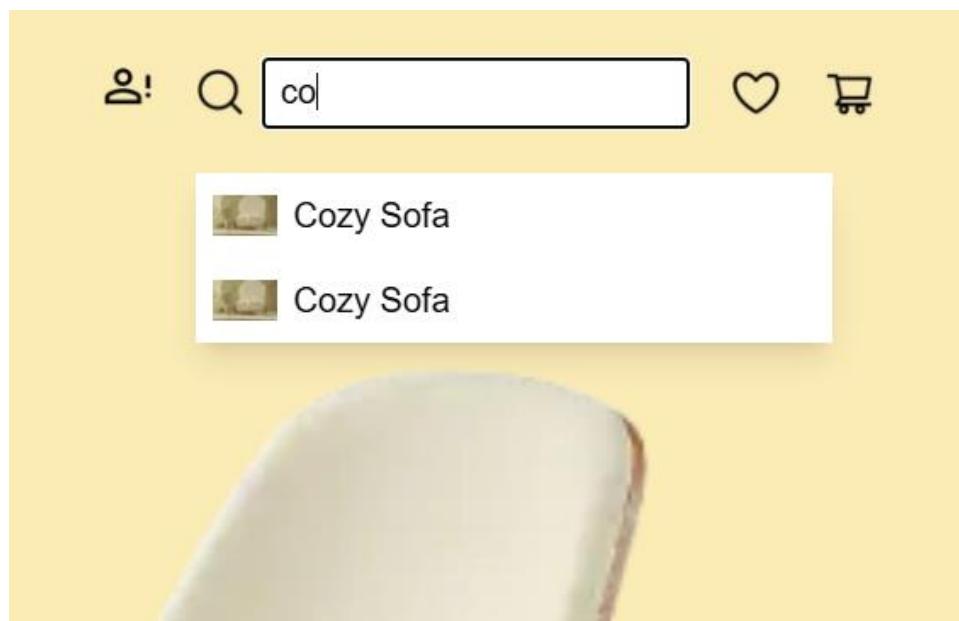


```
File Edit Selection View Go Run Terminal Help ↶ → Hackathon-3 Day-3
page.tsx
1  use client,
2  import React, { useEffect, useState } from "react";
3  import Image from "next/image";
4  import Link from "next/link";
5  > import { ... }
6  > from "@/components/ui/breadcrumb";
7  import ReturnPolicy from "@/components/returnPolicy";
8
9  function Checkout() {
10    const [cartItems, setCartItems] = useState<any>([]);
11    const [selectedMethod, setSelectedMethod] = useState<string>("");
12    // State for selected method
13
14    // Fetch cart items from localStorage when the component mounts
15    useEffect(() => {
16      const savedCart = localStorage.getItem("cartItems");
17      if (savedCart) {
18        setCartItems(JSON.parse(savedCart));
19      }
20    }, []);
21
22    // Calculate subtotal
23    const subtotal = cartItems.reduce(
24      (total, item) => total + item.price * item.quantity,
25      0
26    );
27
28    // Handle payment method change
29    const handlePaymentChange = (method: string) => {
30      setSelectedMethod(method); // Update selected method
31    };
32
33    return (
34      <>
35        <div className="w-full h-[316px] relative flex flex-col items-center">
36          <Image
37            src="shop.jpeg"
38            alt="shop-entry"
39            width={1440}
40            height={450}
41          />
42        </div>
43      </>
44    );
45  }
46
47  export default Checkout;
```

```
page.tsx src\app\check-out
35
36
37
38
39  >
40  <>
41    <div className="w-full h-[316px] relative flex flex-col items-center">
42      </div>
43
44    <div className="w-full min-h-screen flex justify-center items-center p-4 poppins">
45      <div className="w-full max-w-[1242px] flex flex-col lg:flex-row gap-10">
46        /* Billing Details Section */
47        <div className="w-full lg:w-[608px] p-5 flex flex-col items-center">
48          <h1 className="font-semibold text-[28px] md:text-[36px]">
49            Billing details
50          </h1>
51
52          /* Billing Fields */
53          <div className="flex flex-col gap-3">
54            <div className="flex flex-col gap-3">
55              <label className="font-medium">First Name</label>
56              <input
57                className="w-full md:w-[211px] h-[50px] md:h-[75px] rounded-[10px] border-[1px] border-gray-300 p-2"
58                type="text"
59              />
60            </div>
61            <div className="flex flex-col gap-3">
62              <label className="font-medium">Last Name</label>
63              <input
64                className="w-full md:w-[211px] h-[50px] md:h-[75px] rounded-[10px] border-[1px] border-gray-300 p-2"
65                type="text"
66              />
67            </div>
68            <label className="font-medium">Company Name (Optional)</label>
69            <input
70              className="w-full h-[50px] md:h-[75px] rounded-[10px] px-5 border-[1px] border-gray-300 p-2"
71              type="text"
72            />
73          </div>
74        </div>
75      </div>
76    </div>
77
78  </>
79
```

## Product Search Bar

The **Product Search Bar** enhances the user experience by allowing customers to quickly find specific products within the marketplace. This feature is designed to make product discovery faster and more intuitive.



👤 Q bed



White Bed



Blue Bed



Luxury Flower Bed



Solid Bed



White Bed



Red Bed



Matilda Velvet Bed



Blue Bed



Luxury Flower Bed



Solid Bed

Show



Matilda Velvet Bed



Red Bed

## Product Search Bar ( Code )

```
Search.tsx src/components
1  import React, { useState, useEffect, useRef } from "react";
2  import Image from "next/image";
3  import { useRouter } from "next/navigation";
4  import { client } from "@/sanity/lib/client";
5
6  function Search() {
7    const [searchTerm, setSearchTerm] = useState("");
8    const [isOpen, setIsOpen] = useState(false);
9    const [products, setProducts] = useState<
10      { id: string; name: string; imagePath: string }[]
11    >([]);
12    const [filteredProducts, setFilteredProducts] = useState<
13      { id: string; name: string; imagePath: string }[]
14    >([]);
15    const containerRef = useRef<HTMLDivElement | null>(null);
16    const inputRef = useRef<HTMLInputElement | null>(null);
17    const router = useRouter();
18
19    const handleClickOutside = (event: MouseEvent) => {
20      if (
21        containerRef.current &&
22        !containerRef.current.contains(event.target as Node)
23      ) {
24        setIsOpen(false);
25      }
26    };
27
28    const handleclick = () => {
29      setIsOpen(!prevState);
30    };
31
32    useEffect(() => [
33      document.addEventListener("click", handleClickOutside),
34      return () => {
35        document.removeEventListener("click", handleClickOutside);
36      },
37    ], []);
38  }
39
40  useEffect(() => {
41    // Fetch products from Sanity
42    const fetchProducts = async () => {
43      const data = await client.fetch(
44        `*[_type == "product"]{ name, imagePath, id }`
45      );
46      setProducts(data);
47    };
48    fetchProducts();
49  }, []);
50
51  useEffect(() => {
52    // Filter products based on search term
53    if (searchTerm) {
54      const filtered = products.filter((product) =>
55        product.name.toLowerCase().includes(searchTerm.toLowerCase())
56      );
57      setFilteredProducts(filtered);
58    } else {
59      setFilteredProducts([]);
60    }
61  }, [searchTerm, products]);
62
63  const handleProductClick = (id: string) => {
64    router.push(`/product/${id}`);
65  };
66
67  return (
68    <div className="flex items-center gap-2" ref={containerRef}>
69      <div className="w-6 h-6 cursor-pointer" onClick={handleClick}>
70        <Image src="/2.png" alt="search-icon" width={28} height={28} />
71      </div>
72
73      <input
74        ref={inputRef}
75        type="text"
76        className="transition-all duration-300 ease-in-out ${isOpen ? "w-[200px]" : "w-[100px]"} placeholder='Type here...'" />
77    </div>
78  );
79}

Ln 37, Col 7  Spaces: 2  UTF-8  CRLF  {} TypeScript JSX  ⌂ Go Live  ⌂ Prettier  ⌂
```

The Furniture E-Commerce Website combines dynamic functionality, a professional design, and scalable architecture to deliver an engaging and user-friendly marketplace platform. By leveraging **Sanity CMS, API** and **Next.js**, this project ensures efficient data handling, responsive layouts, and seamless navigation. These features collectively provide a robust foundation for a modern e-commerce experience.

The End.