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 Paginations)

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

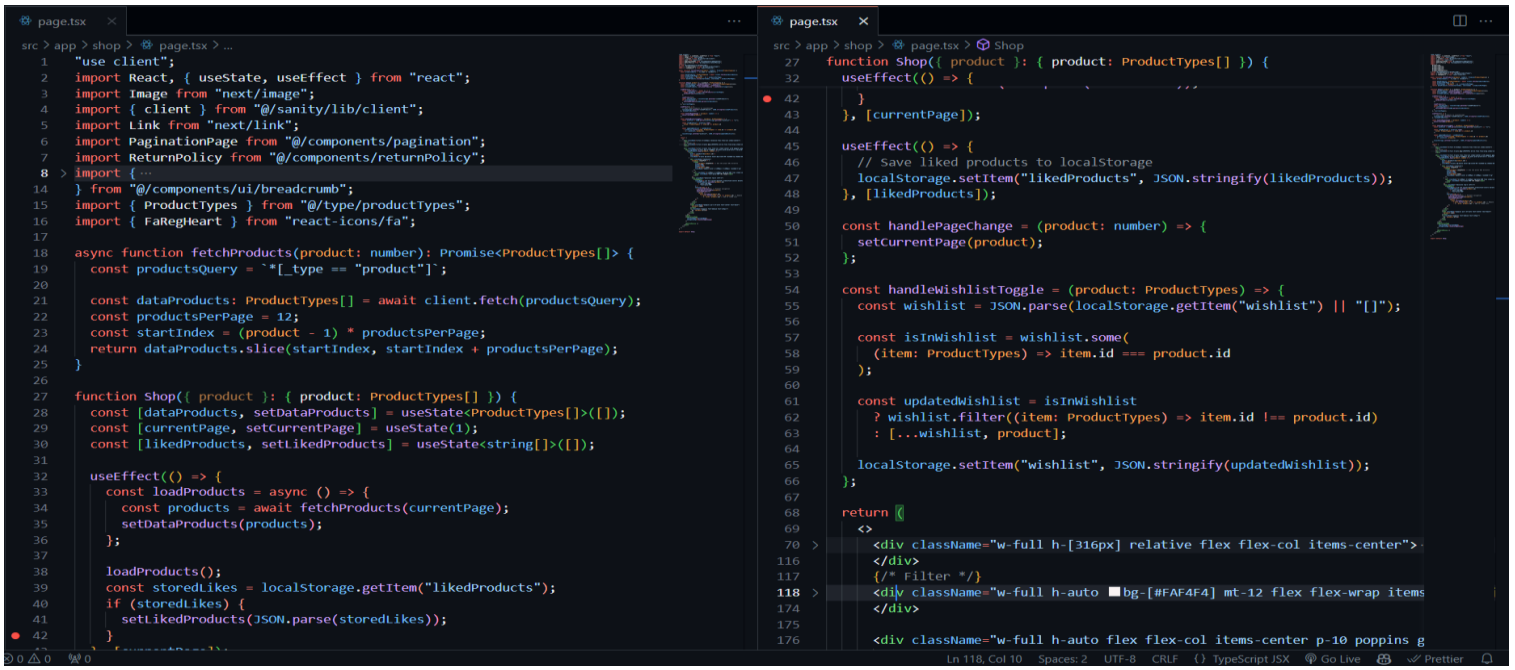
Rs. 720

Matilda Velvet Bed

Rs. 600

‹ Previous 1 2 3 Next ›

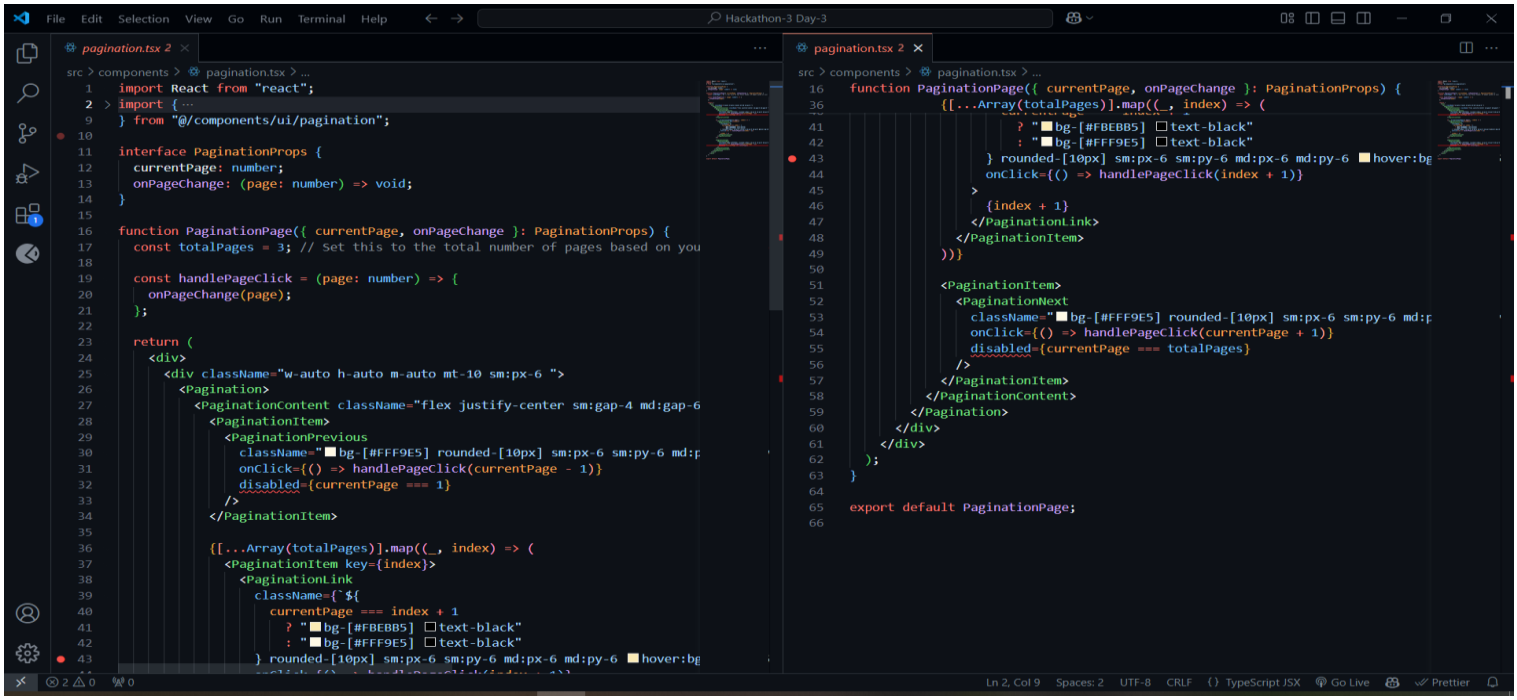
Sanity And Api Data Integration (Product Listing (Code))



```tsx
"use client";
import React, { useState, useEffect } from "react";
import Image from "next/image";
import { client } from "@/sanity/lib/client";
import Link from "next/link";
import PaginationPage from "@/components/pagination";
import ReturnPolicy from "@/components/returnPolicy";
import {…
} from "@/components/ui/breadcrumb";
import { ProductTypes } from "@/type/productTypes";
import { FaRegHeart } from "react-icons/fa";

async function fetchProducts(product: number): Promise<ProductTypes[]> {
  const productsQuery = `*[_type == "product"]`;

  const dataProducts: ProductTypes[] = await client.fetch(productsQuery);
  const productsPerPage = 12;
  const startIndex = (product - 1) * productsPerPage;
  return dataProducts.slice(startIndex, startIndex + productsPerPage);
}

function Shop({ product }: { product: ProductTypes[] }) {
  const [dataProducts, setDataProducts] = useState<ProductTypes[]>([]);
  const [currentPage, setCurrentPage] = useState(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));
    }
```

```tsx
function Shop({ product }: { product: ProductTypes[] }) {
  useEffect(() => {
    }
  }, [currentPage]);

  useEffect(() => {
    // Save liked products to localStorage
    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 className="w-full h-[316px] relative flex flex-col items-center">…
      </div>
      {/* Filter */}
      <div className="w-full h-auto bg-[#FAF4F4] mt-12 flex flex-wrap items…
      </div>

      <div className="w-full h-auto flex flex-col items-center p-10 poppins g…
```

Pagination (Code)

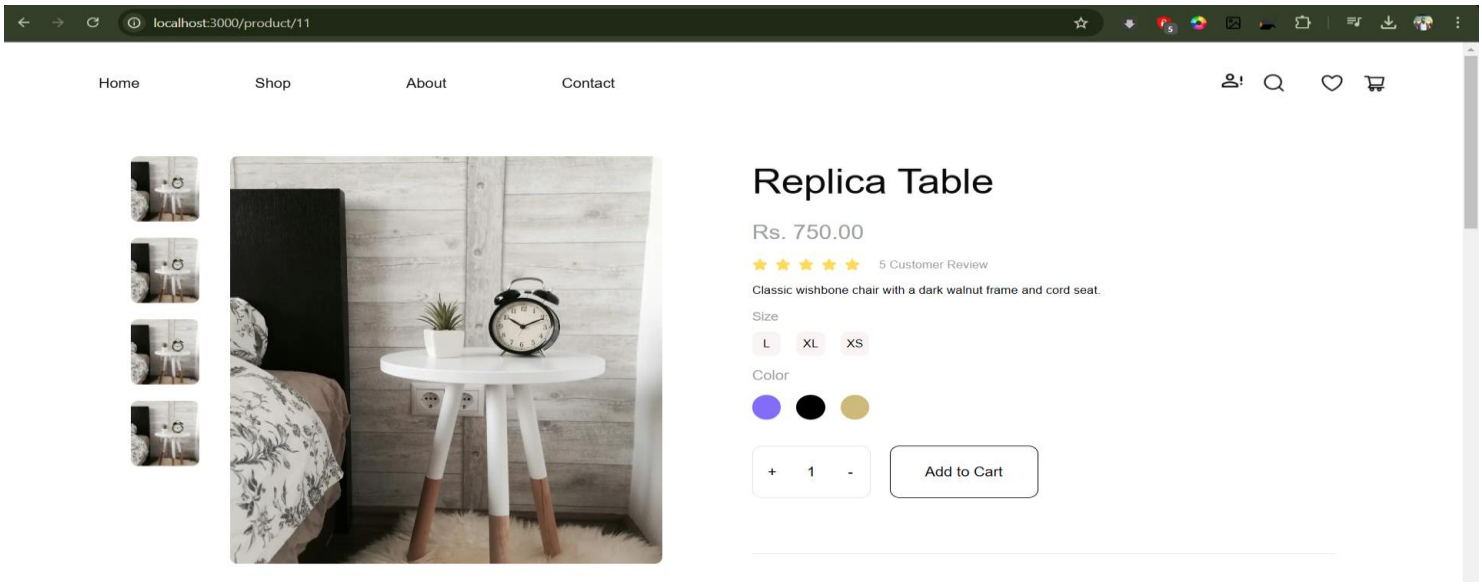


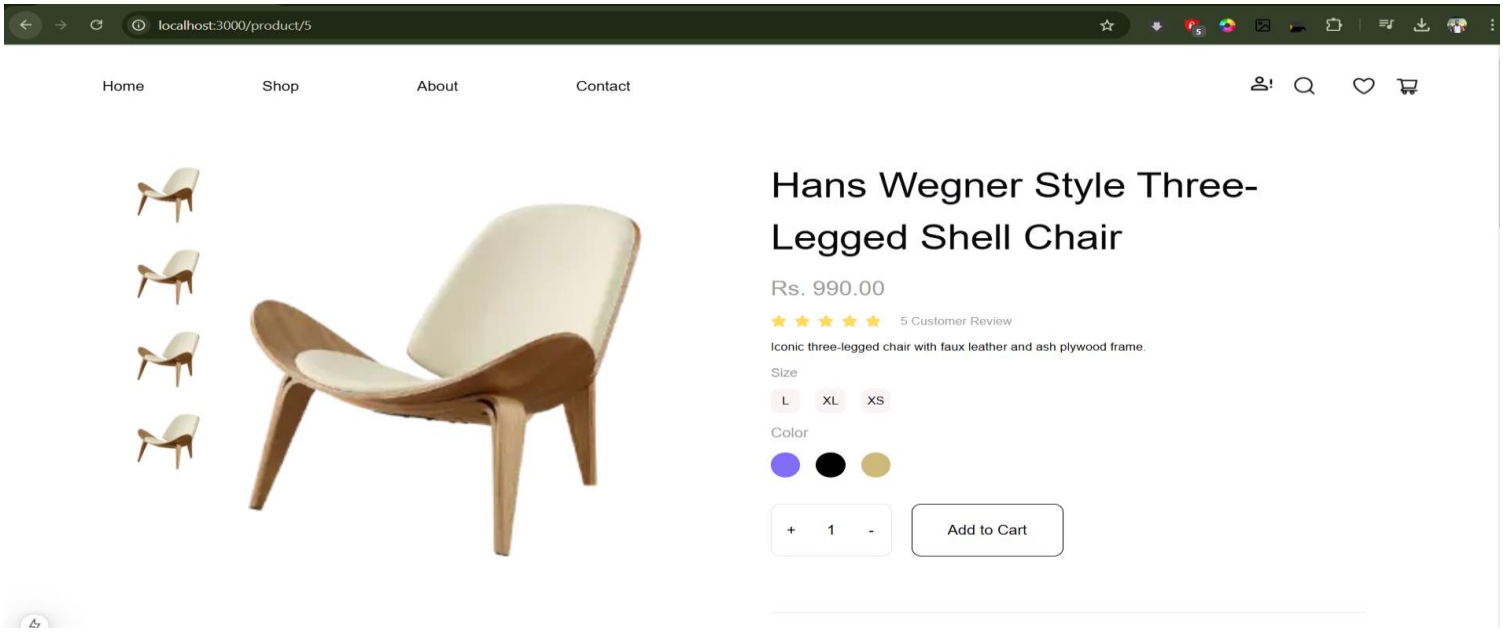
```tsx
import React from "react";
import {…
} from "@/components/ui/pagination";

interface PaginationProps {
  currentPage: number;
  onPageChange: (page: number) => void;
}

function PaginationPage({ currentPage, onPageChange }: PaginationProps) {
  const totalPages = 3; // Set this to the total number of pages based on you

  const handlePageClick = (page: number) => {
    onPageChange(page);
  };

  return (
    <div>
      <div className="w-auto h-auto m-auto mt-10 sm:px-6 ">
        <Pagination>
          <PaginationContent className="flex justify-center sm:gap-4 md:gap-6
            <PaginationItem>
              <PaginationPrevious
                className="bg-[#FFF9E5] rounded-[10px] sm:px-6 sm:py-6 md:p
                onClick={() => handlePageClick(currentPage - 1)}
                disabled={currentPage === 1}
              />
            </PaginationItem>

            {[...Array(totalPages)].map((_, index) => (
              <PaginationItem key={index}>
                <PaginationLink
                  className={`${
                    currentPage === index + 1
                      ? "bg-[#FBEBB5] text-black"
                      : "bg-[#FFF9E5] text-black"
                  } rounded-[10px] sm:px-6 sm:py-6 md:px-6 md:py-6 hover:bg
```

```tsx
function PaginationPage({ currentPage, onPageChange }: PaginationProps) {
                {[...Array(totalPages)].map((_, index) => (
                  ? "bg-[#FBEBB5] text-black"
                  : "bg-[#FFF9E5] text-black"
                } rounded-[10px] sm:px-6 sm:py-6 md:px-6 md:py-6 hover:bg
                onClick={() => handlePageClick(index + 1)}
              >
                {index + 1}
              </PaginationLink>
            </PaginationItem>
          ))}

          <PaginationItem>
            <PaginationNext
              className="bg-[#FFF9E5] rounded-[10px] sm:px-6 sm:py-6 md:p
              onClick={() => handlePageClick(currentPage + 1)}
              disabled={currentPage === totalPages}
            />
          </PaginationItem>
        </PaginationContent>
      </Pagination>
    </div>
  </div>
);
}

export default PaginationPage;
```

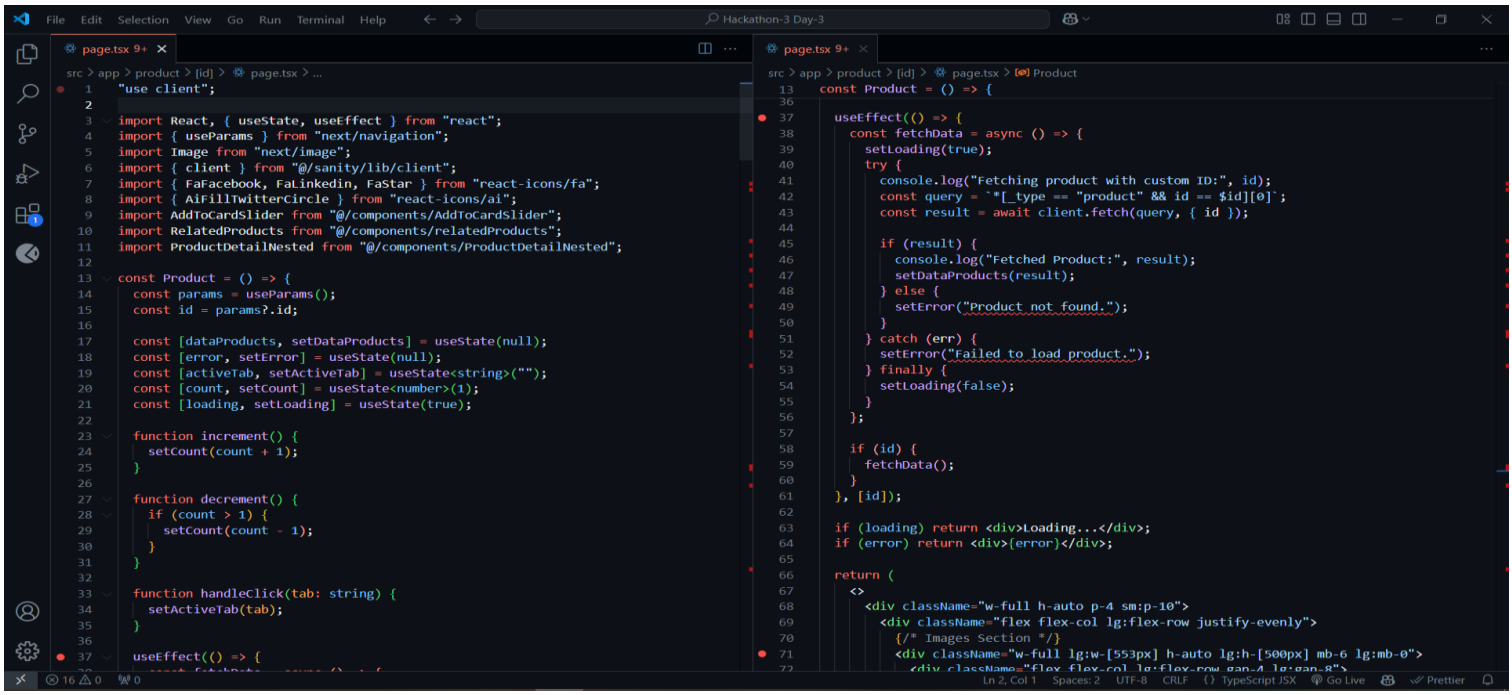
Dynamic Routing (All Listing Product and Home Product)

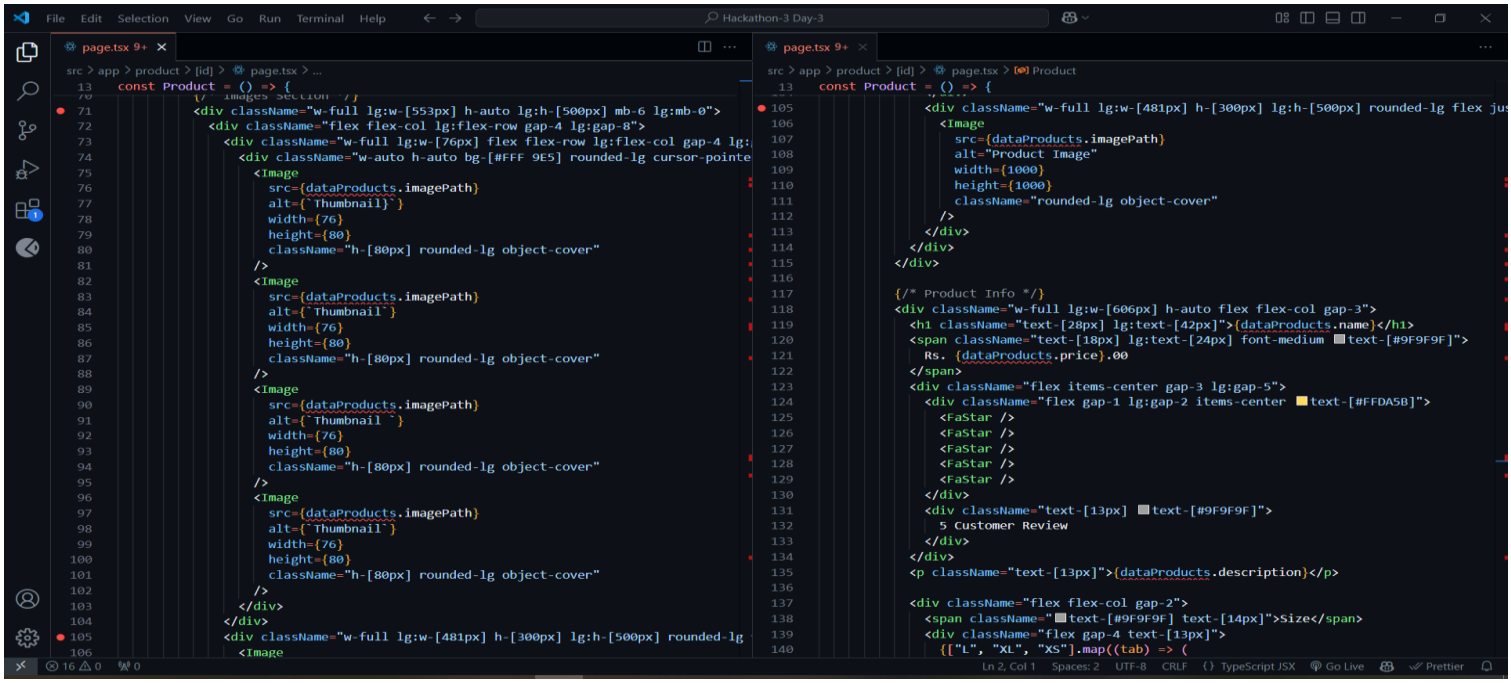
Dynamic routing has been implemented in the Furniture E-Commerce Website using Next.js. This enables seamless navigation for individual product details, including images, descriptions, sizes, and colors. The route structure allows fetching product data dynamically for both the home page and the product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

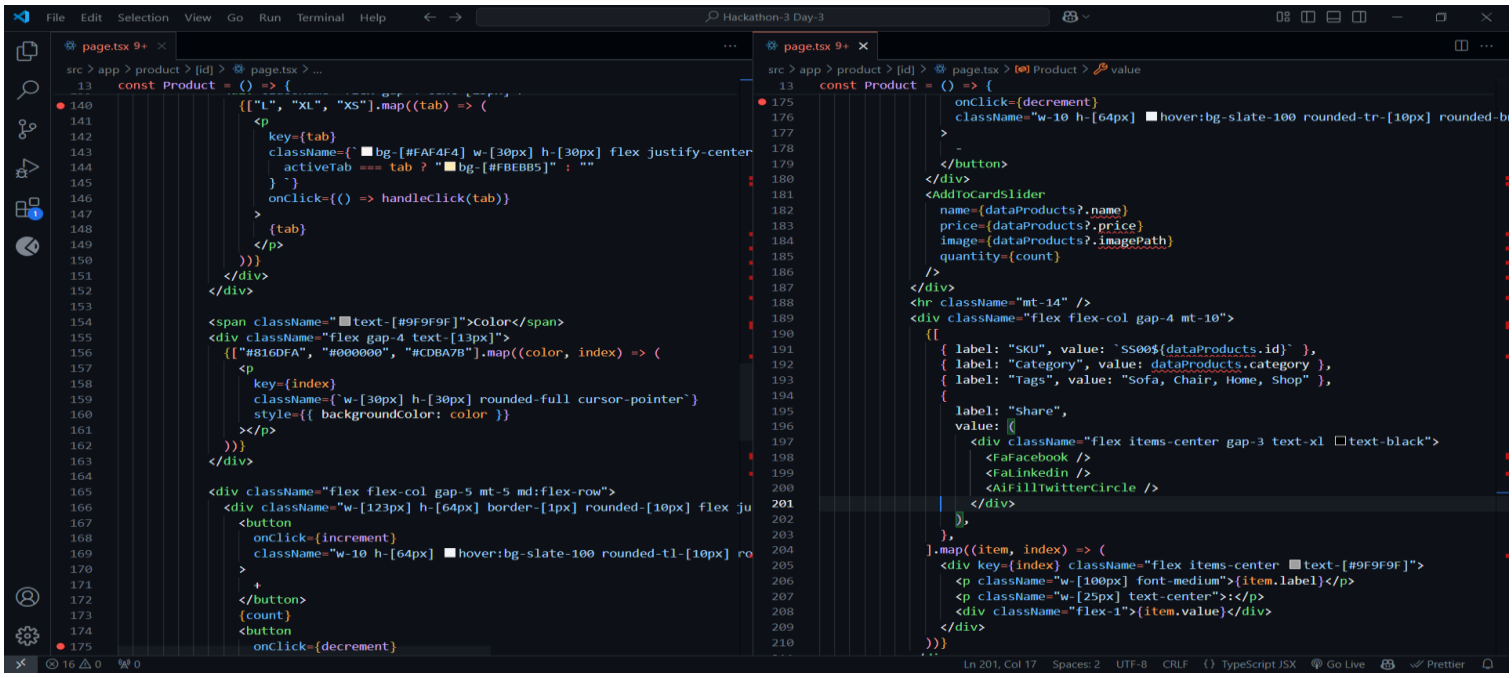




Dynamic Routing (Product (Code))



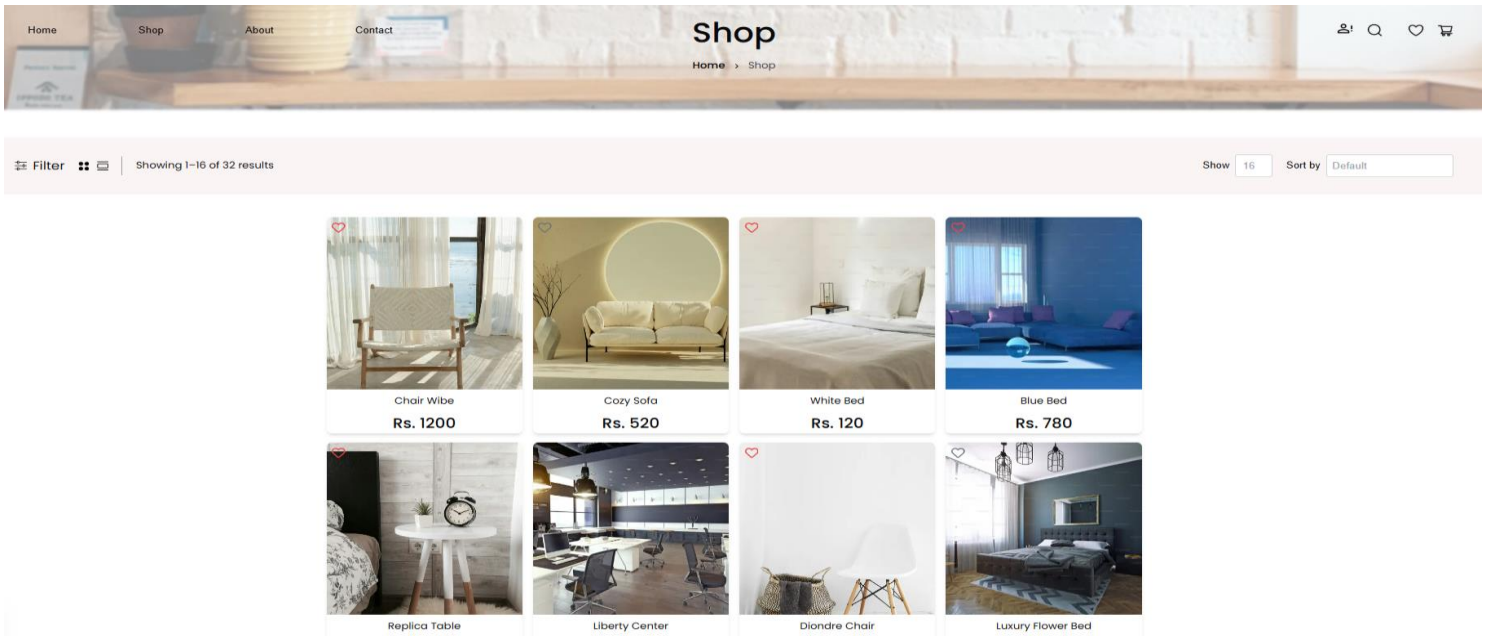


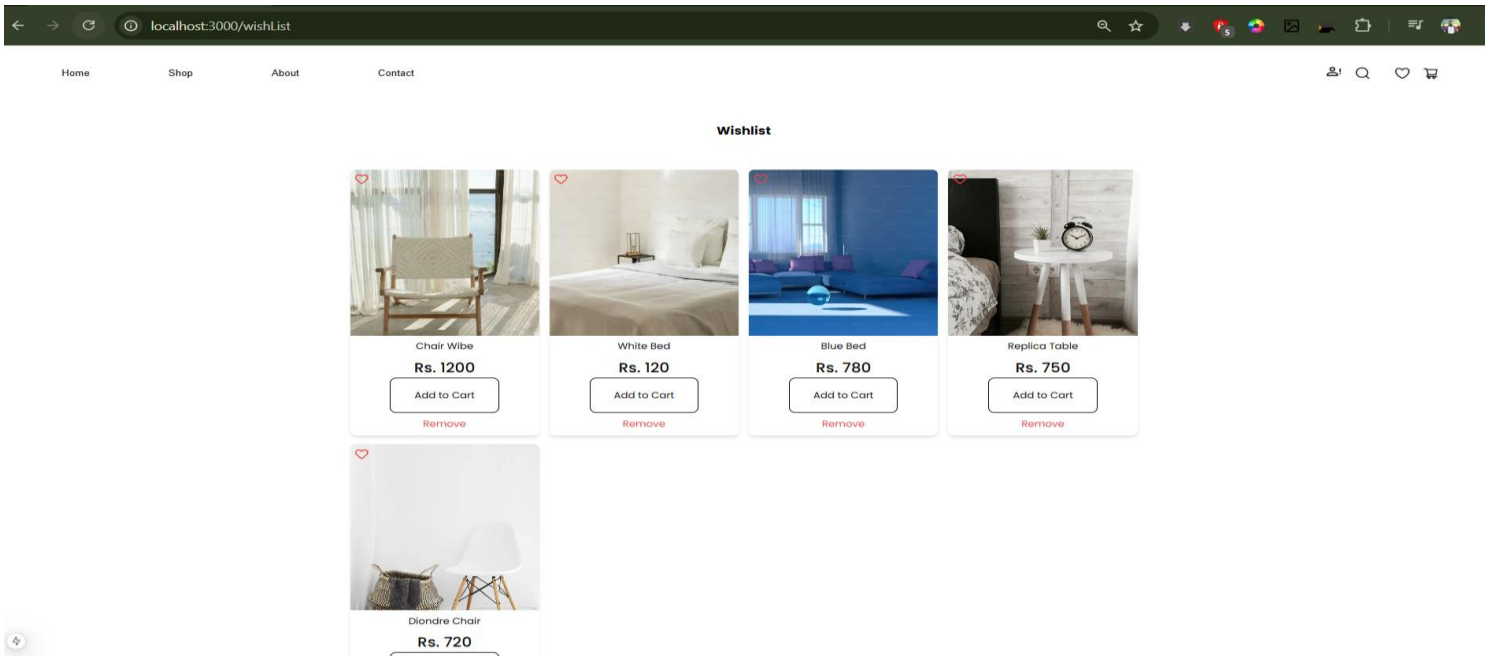


Product (Wishlist’s)

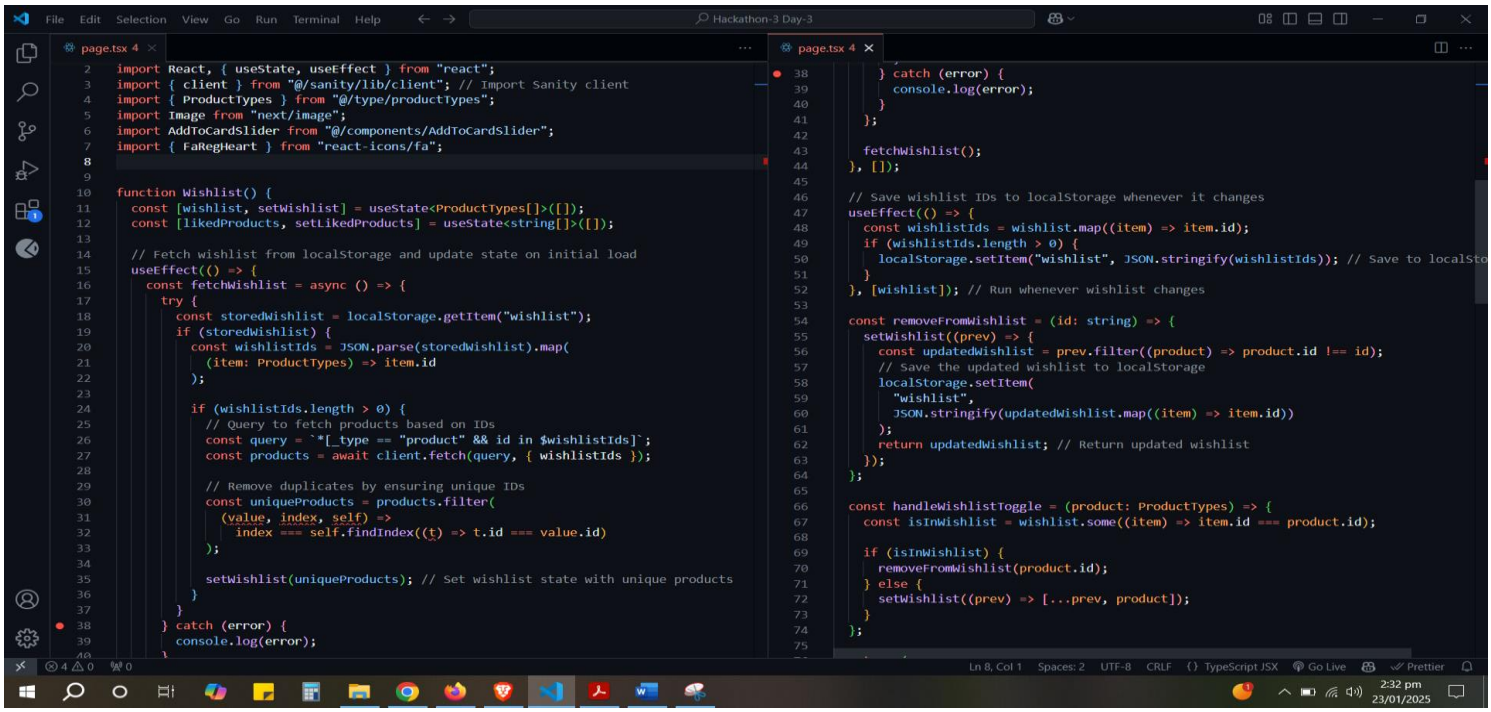
1. Users can add products to the Wishlist by clicking the heart icon on any product.
2. The selected products are saved on a dedicated **Wishlist Page**.
3. Users can view and manage their favorite items from the Wishlist.

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

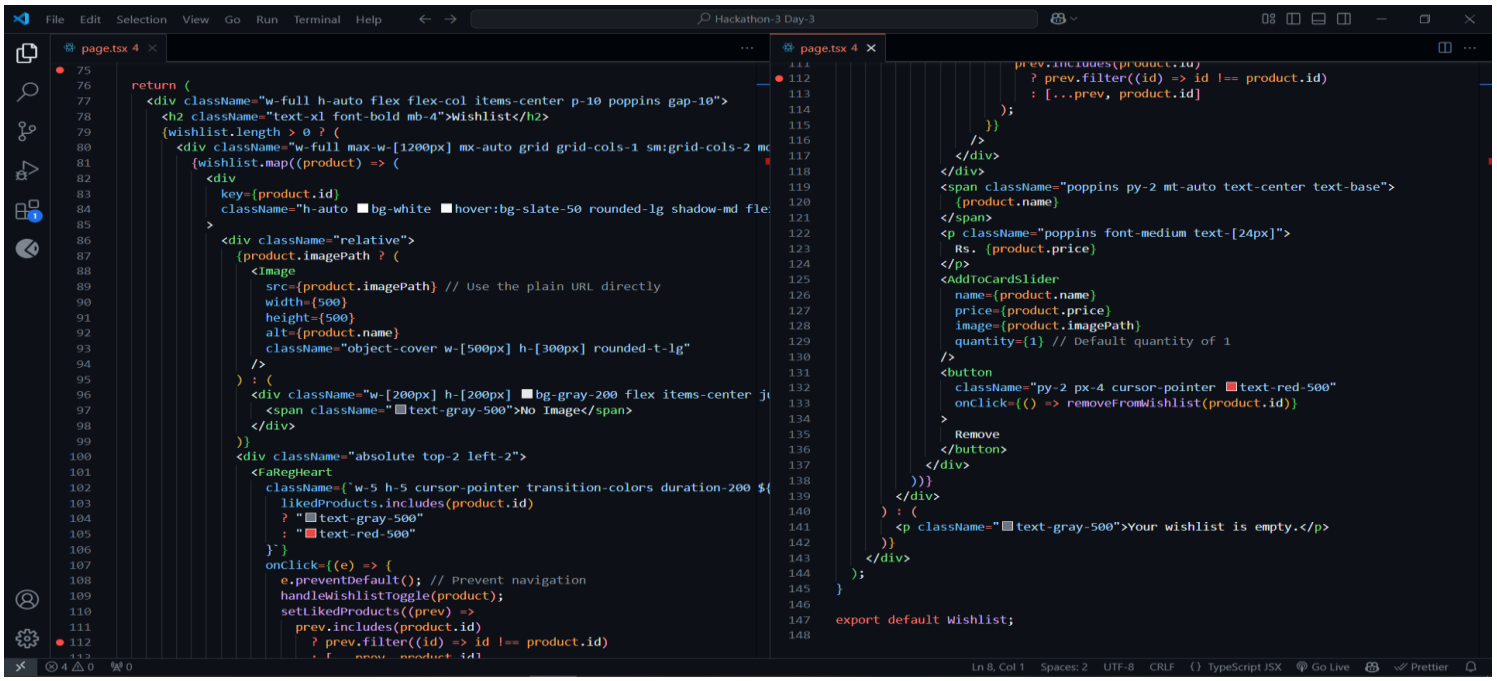




Product (Wishlist's (WishList Component Code))

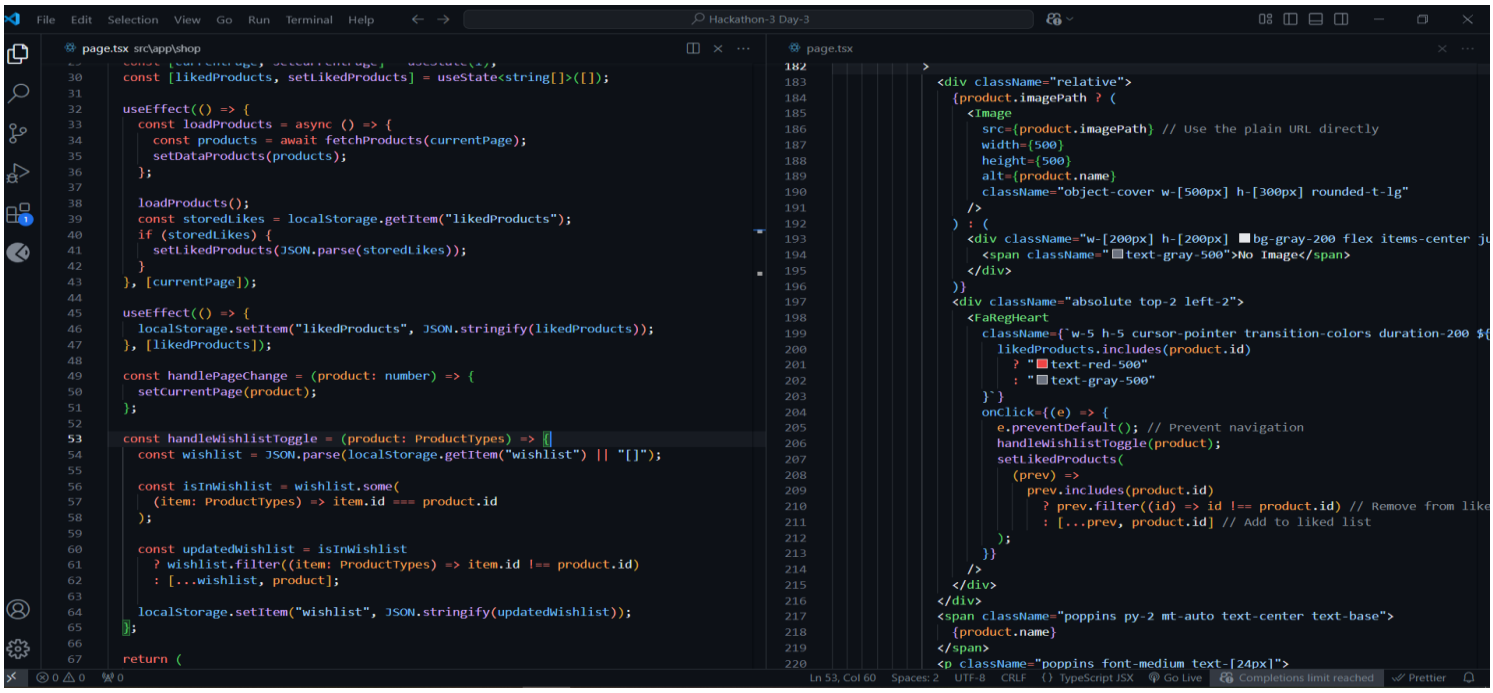


```tsx
import React, { useState, useEffect } from "react";
import { client } from "@/sanity/lib/client"; // Import Sanity client
import { ProductTypes } from "@/type/productTypes";
import Image from "next/image";
import AddToCardSlider from "@/components/AddToCardSlider";
import { FaRegHeart } from "react-icons/fa";

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();
  }, []);

  // 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)); // Save to localSto
    }
  }, [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]);
    }
  };
```



```tsx
  return (
    <div className="w-full h-auto flex flex-col items-center p-10 poppins gap-10">
      <h2 className="text-xl font-bold mb-4">Wishlist</h2>
      {wishlist.length > 0 ? (
        <div className="w-full max-w-[1200px] mx-auto grid grid-cols-1 sm:grid-cols-2 md
          {wishlist.map((product) => (
            <div
              key={product.id}
              className="h-auto bg-white hover:bg-slate-50 rounded-lg shadow-md fle
            >
              <div className="relative">
                {product.imagePath ? (
                  <Image
                    src={product.imagePath} // Use the plain URL directly
                    width={500}
                    height={500}
                    alt={product.name}
                    className="object-cover w-[500px] h-[300px] rounded-t-lg"
                  />
                ) : (
                  <div className="w-[200px] h-[200px] bg-gray-200 flex items-center ju
                    <span>No Image</span>
                  </div>
                )}
                <div>
                  <FaRegHeart
                    className={`w-5 h-5 cursor-pointer transition-colors duration-200 ${
                      likedProducts.includes(product.id)
                      ? "text-gray-500"
                      : "text-red-500"
                    }`}
                    onClick={(e) => {
                      e.preventDefault(); // Prevent navigation
                      handleWishlistToggle(product);
                      setLikedProducts((prev) =>
                        prev.includes(product.id)
                          ? prev.filter((id) => id !== product.id)
                          : [...prev, product.id]
                      );
                    }}
                  />
                </div>
              </div>
              <span>
                {product.name}
              </span>
              <p>
                Rs. {product.price}
              </p>
              <AddToCardSlider
                name={product.name}
                price={product.price}
                image={product.imagePath}
                quantity={1} // Default quantity of 1
              />
              <button
                onClick={() => removeFromWishlist(product.id)}
              >
                Remove
              </button>
            </div>
          ))}
        </div>
      ) : (
        <p>Your wishlist is empty.</p>
      )}
    </div>
  );
}

export default Wishlist;
```

Product (Wishlist's (Shop Component Code))



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:

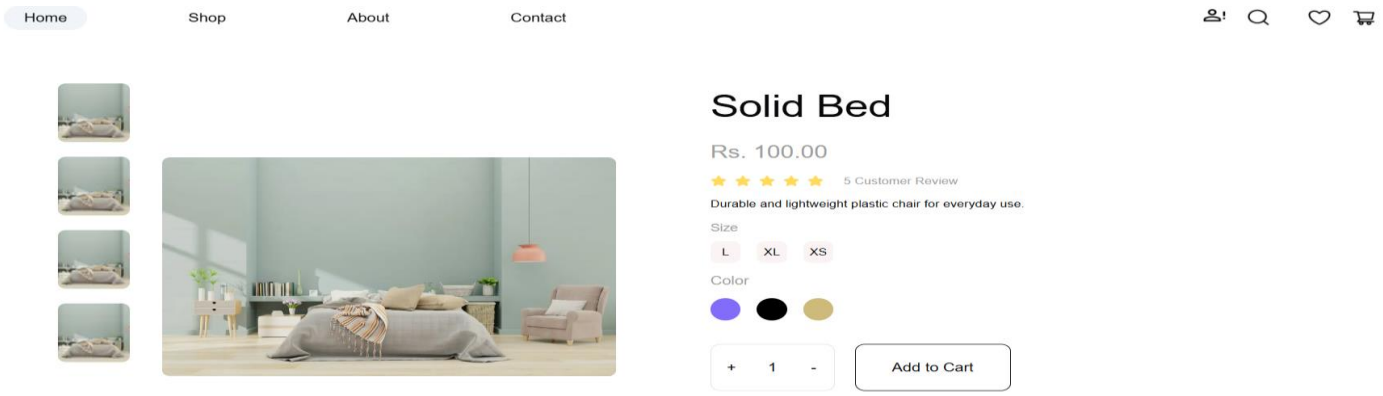
1. **Add to Cart Functionality**:
    - Users can add any product to the cart directly from the product listing or detail pages by specifying the quantity.
    - The cart dynamically updates to show the total quantity of items, the total number of unique products, and the total price in real-time.
2. **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.

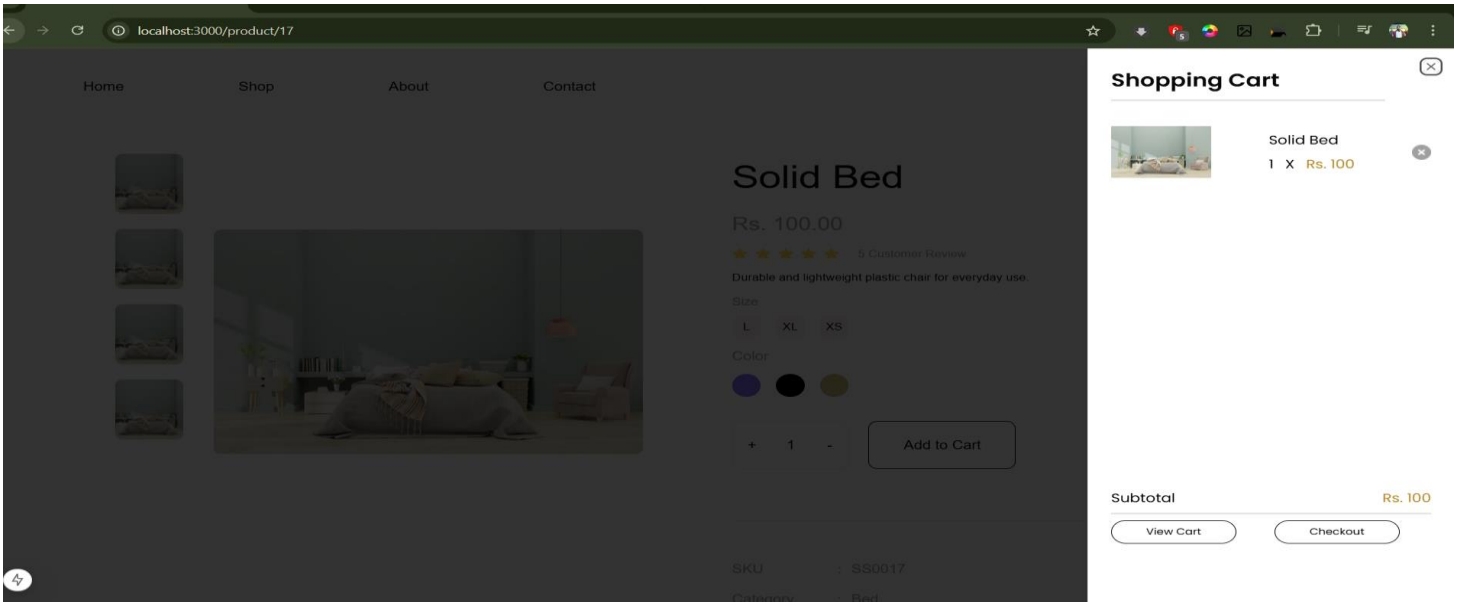
- **Dynamic Price Update**: The total amount adjusts automatically as users modify the quantity or remove items.

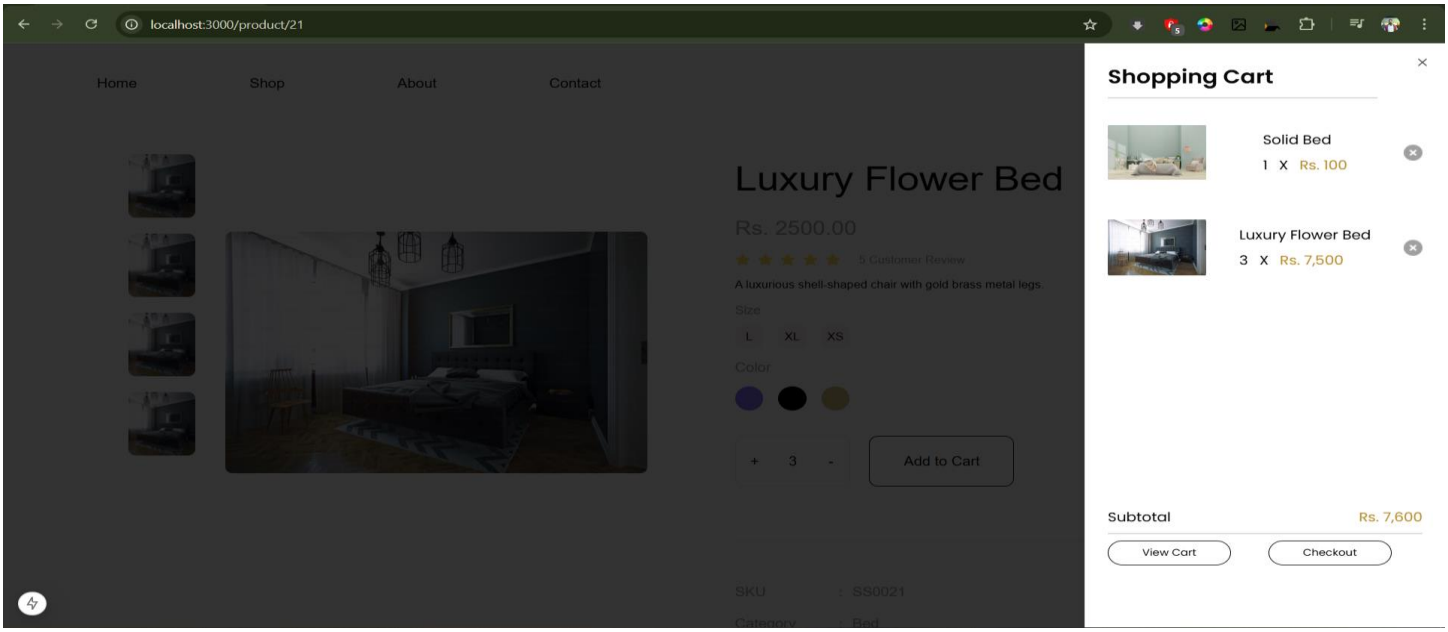
3. **Checkout Process**:
    - Once satisfied with the cart items, users can proceed by clicking the **Checkout Button**.
    - During checkout, users are guided through the process of entering delivery details, reviewing the order summary, and selecting payment methods.
4. **Order Confirmation**:
 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




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
+ 1 -
Add to Cart
SKU : SS0017
Shopping Cart
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
+ 3 -
Add to Cart
SKU : SS0021
Shopping Cart
Solid Bed
1 X Rs. 100
Luxury Flower Bed
3 X Rs. 7,500
Subtotal Rs. 7,600
View Cart
Checkout

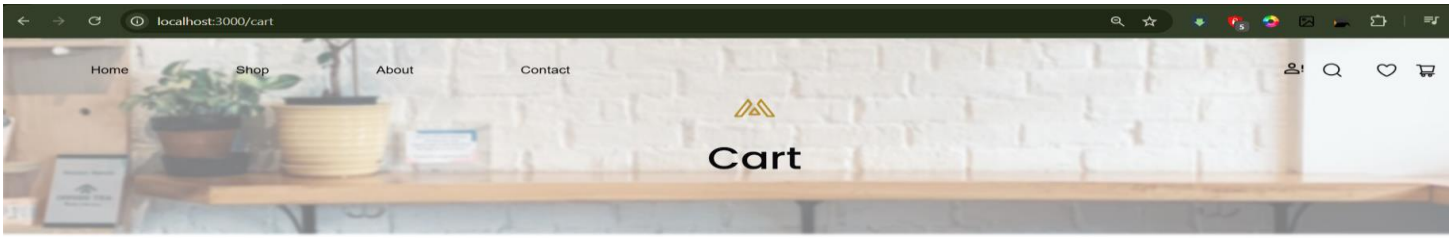
Product	Price	Quantity	Subtotal
Solid Bed	Rs. 100	12	Rs. 1,200
Luxury Flower Bed	Rs. 2,500	2	Rs. 5,000

Cart Totals

Subtotal Rs. 6,200

Total Rs. 6,200

Check Out



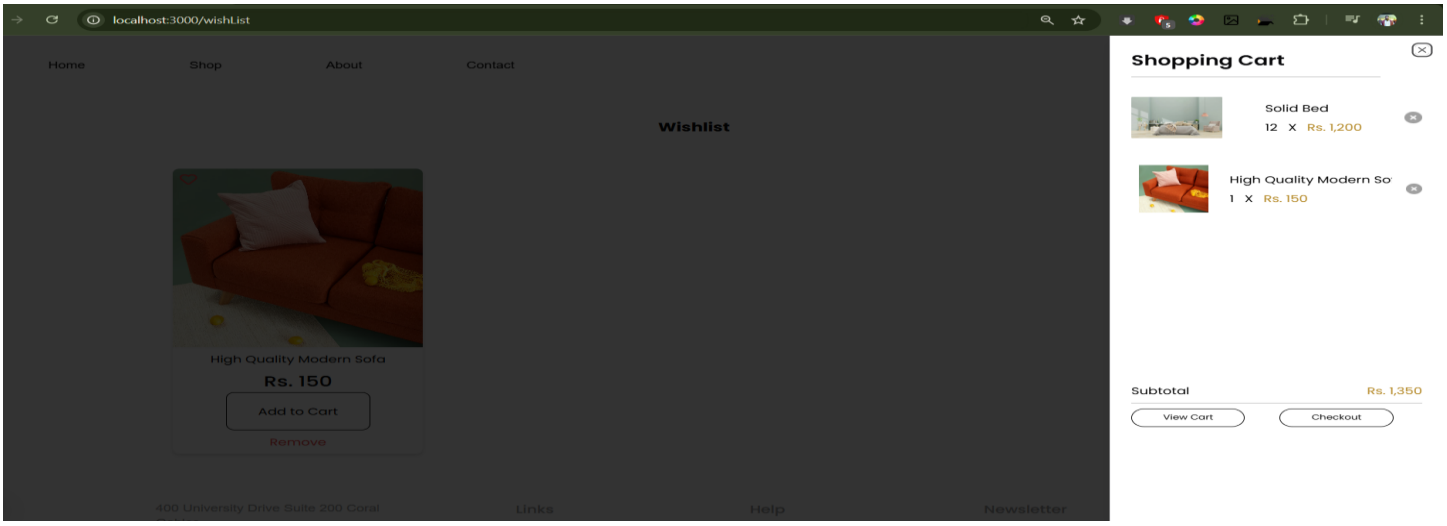
Product	Price	Quantity	Subtotal
Solid Bed	Rs. 100	12	Rs. 1,200

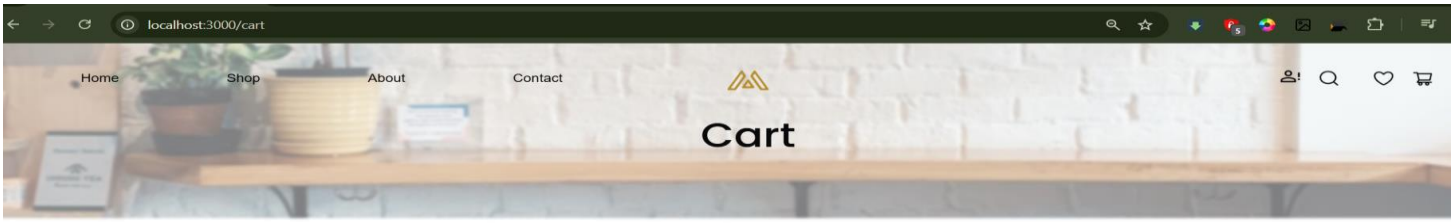
Cart Totals

Subtotal Rs. 1,200

Total Rs. 1,200

Check Out



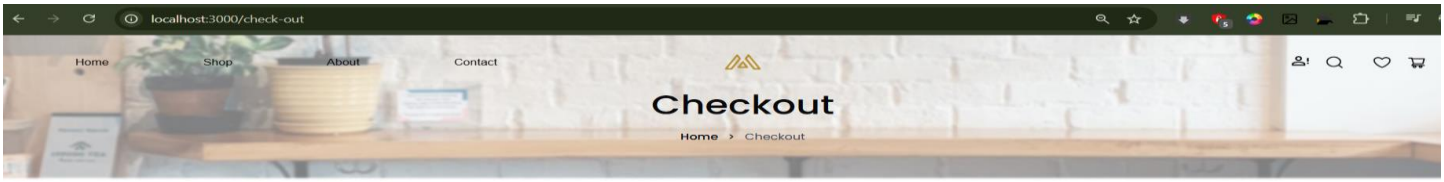


Product	Price	Quantity	Subtotal
Solid Bed	Rs. 100	12	Rs. 1,200
High Quality Modern Sofa	Rs. 150	1	Rs. 150

Cart Totals

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

Check Out



Billing details

First Name

Last Name

Company Name (Optional)

Country / Region

Sri Lanka

Street address

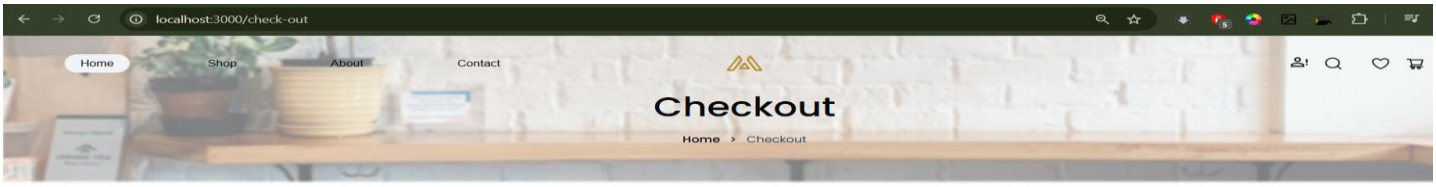
Product	Subtotal
Solid Bed	Rs. 1,200
High Quality Modern Sofa	Rs. 150
Total	Rs. 1,350

Direct Bank Transfer

Make your payment directly into our bank account.

Cash On Delivery

Place order



Billing details

First Name

Last Name

Company Name (Optional)

Country / Region

Sri Lanka

Street address

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

Direct Bank Transfer

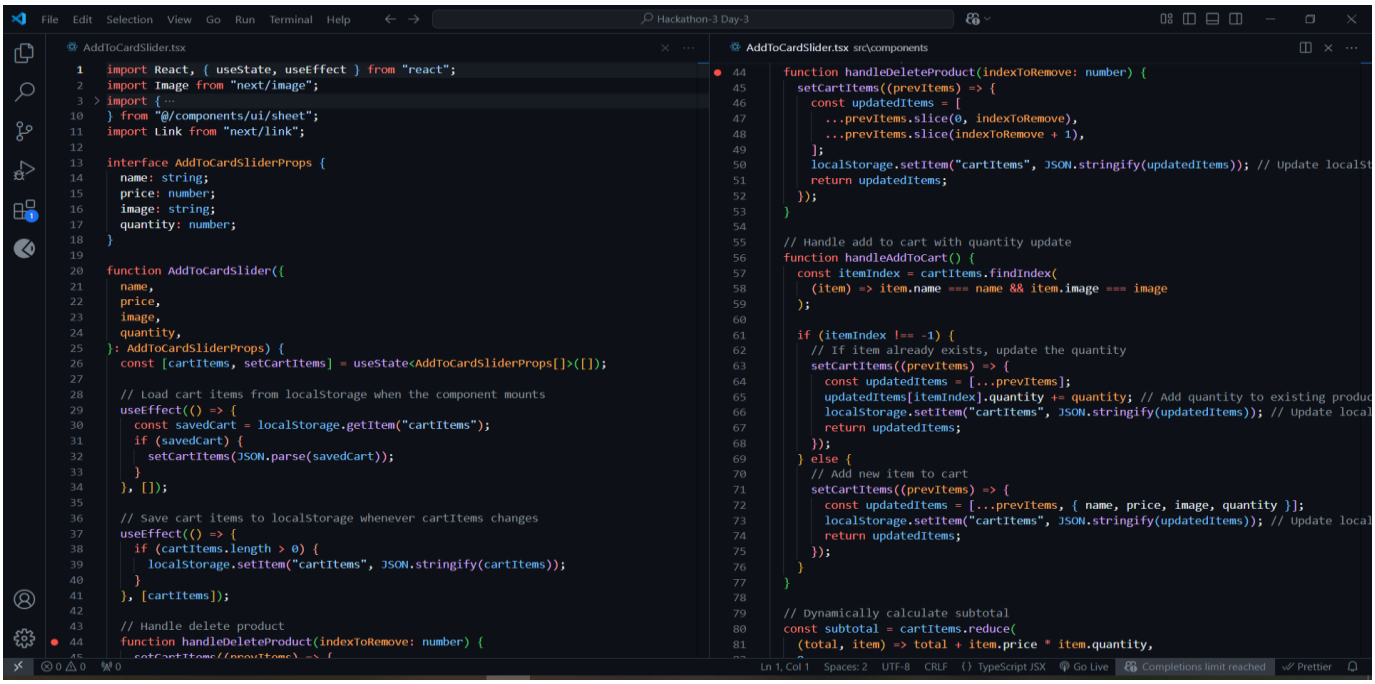
Cash On Delivery

Pay with cash when your order is delivered.

Place order

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

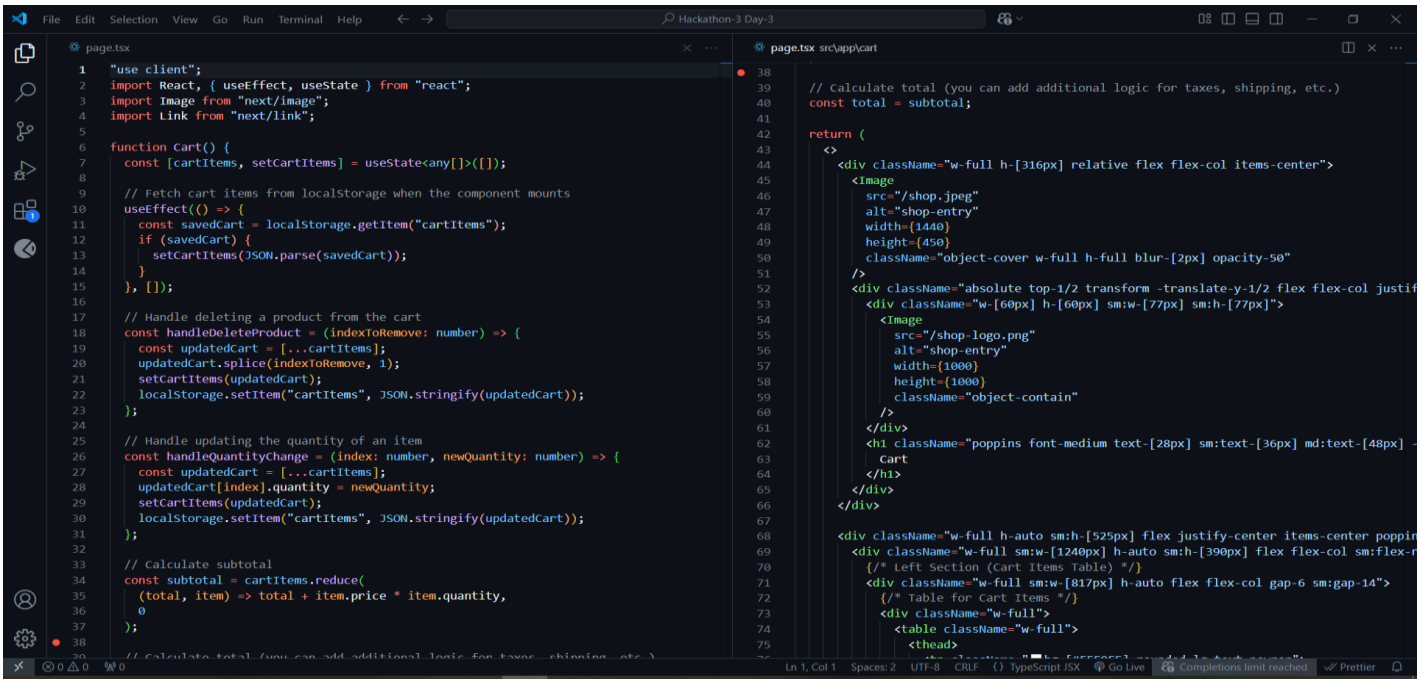
Add to Cart Slider.



```tsx
// AddToCardSlider.tsx
import React, { useState, useEffect } from "react";
import Image from "next/image";
import {
} from "@/components/ui/sheet";
import Link from "next/link";

interface AddToCardSliderProps {
  name: string;
  price: number;
  image: string;
  quantity: number;
}

function AddToCardSlider({
  name,
  price,
  image,
  quantity,
}: AddToCardSliderProps) {
  const [cartItems, setCartItems] = useState<AddToCardSliderProps[]>([]);

  // Load cart items from localStorage when the component mounts
  useEffect(() => {
    const savedCart = localStorage.getItem("cartItems");
    if (savedCart) {
      setCartItems(JSON.parse(savedCart));
    }
  }, []);

  // Save cart items to localStorage whenever cartItems changes
  useEffect(() => {
    if (cartItems.length > 0) {
      localStorage.setItem("cartItems", JSON.stringify(cartItems));
    }
  }, [cartItems]);

  // Handle delete product
  function handleDeleteProduct(indexToRemove: number) {
```

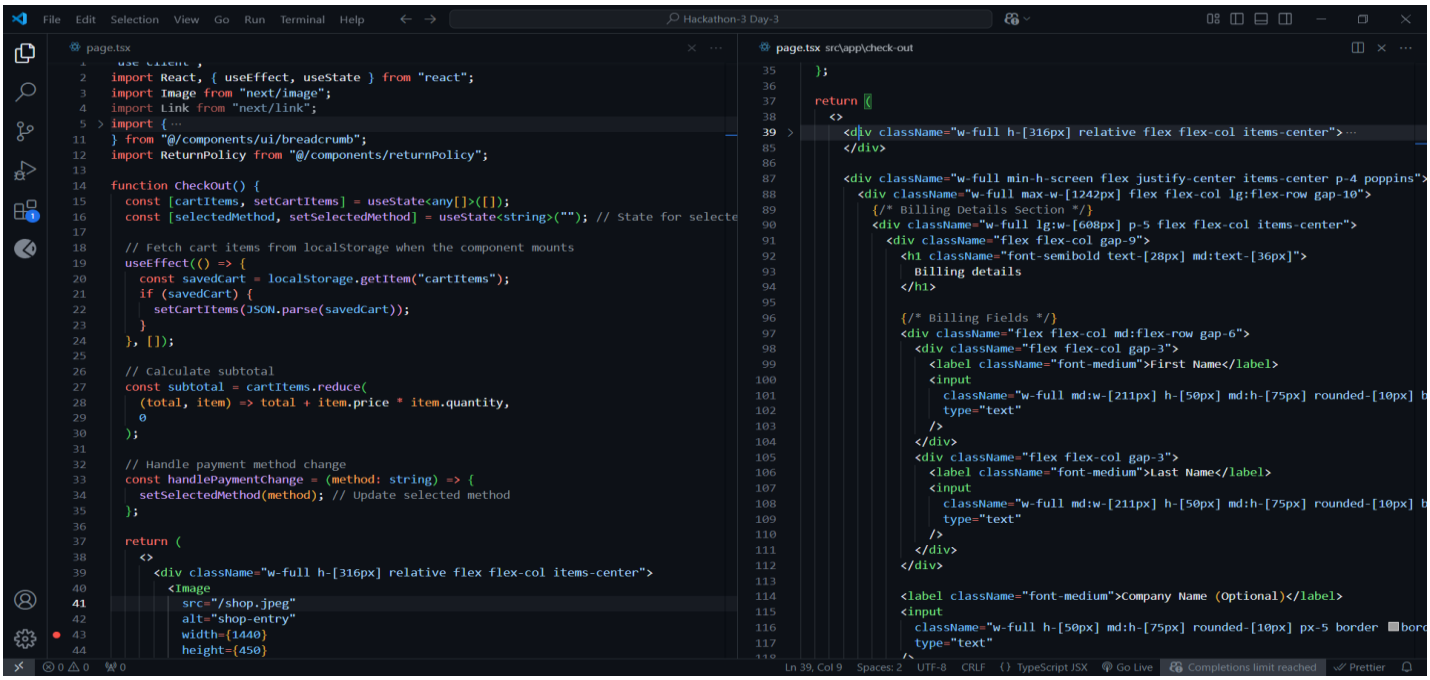
```tsx
// AddToCardSlider.tsx src\components
function handleDeleteProduct(indexToRemove: number) {
  setCartItems((prevItems) => {
    const updatedItems = [
      ...prevItems.slice(0, indexToRemove),
      ...prevItems.slice(indexToRemove + 1),
    ];
    localStorage.setItem("cartItems", JSON.stringify(updatedItems)); // Update localSt
    return updatedItems;
  });
}

// Handle add to cart with quantity update
function handleAddToCart() {
  const itemIndex = cartItems.findIndex(
    (item) => item.name === name && item.image === image
  );

  if (itemIndex !== -1) {
    // If item already exists, update the quantity
    setCartItems((prevItems) => {
      const updatedItems = [...prevItems];
      updatedItems[itemIndex].quantity += quantity; // Add quantity to existing produc
      localStorage.setItem("cartItems", JSON.stringify(updatedItems)); // Update local
      return updatedItems;
    });
  } else {
    // Add new item to cart
    setCartItems((prevItems) => {
      const updatedItems = [...prevItems, { name, price, image, quantity }];
      localStorage.setItem("cartItems", JSON.stringify(updatedItems)); // Update local
      return updatedItems;
    });
  }
}

// Dynamically calculate subtotal
const subtotal = cartItems.reduce(
  (total, item) => total + item.price * item.quantity,
```

Cart.



```tsx
// page.tsx
"use client";
import React, { useEffect, useState } from "react";
import Image from "next/image";
import Link from "next/link";

function Cart() {
  const [cartItems, setCartItems] = useState<any[]>([]);

  // Fetch cart items from localStorage when the component mounts
  useEffect(() => {
    const savedCart = localStorage.getItem("cartItems");
    if (savedCart) {
      setCartItems(JSON.parse(savedCart));
    }
  }, []);

  // Handle deleting a product from the cart
  const handleDeleteProduct = (indexToRemove: number) => {
    const updatedCart = [...cartItems];
    updatedCart.splice(indexToRemove, 1);
    setCartItems(updatedCart);
    localStorage.setItem("cartItems", JSON.stringify(updatedCart));
  };

  // Handle updating the quantity of an item
  const handleQuantityChange = (index: number, newQuantity: number) => {
    const updatedCart = [...cartItems];
    updatedCart[index].quantity = newQuantity;
    setCartItems(updatedCart);
    localStorage.setItem("cartItems", JSON.stringify(updatedCart));
  };

  // Calculate subtotal
  const subtotal = cartItems.reduce(
    (total, item) => total + item.price * item.quantity,
    0
  );

  // Calculate total (you can add additional logic for taxes, shipping, etc.)
  const total = subtotal;

  return (
    <>
      <div className="w-full h-[316px] relative flex flex-col items-center">
        <Image
          src="/shop.jpeg"
          alt="shop-entry"
          width={1440}
          height={450}
          className="object-cover w-full h-full blur-[2px] opacity-50"
        />
        <div className="absolute top-1/2 transform -translate-y-1/2 flex flex-col justif
          <div className="w-[60px] h-[60px] sm:w-[77px] sm:h-[77px]">
            <Image
              src="/shop-logo.png"
              alt="shop-entry"
              width={1000}
              height={1000}
              className="object-contain"
            />
          </div>
          <h1 className="poppins font-medium text-[28px] sm:text-[36px] md:text-[48px]
            Cart
          </h1>
        </div>
      </div>

      <div className="w-full h-auto sm:h-[525px] flex justify-center items-center poppi
        <div className="w-full sm:w-[1240px] h-auto sm:h-[390px] flex flex-col sm:flex-r
          {/* Left Section (Cart Items Table) */}
          <div className="w-full sm:w-[817px] h-auto flex flex-col gap-6 sm:gap-14">
            {/* Table for Cart Items */}
            <div className="w-full">
              <table className="w-full">
                <thead>
```

Checkout.



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.

bed


White Bed


Blue Bed


Luxury Flower Bed


Solid Bed


White Bed


Red Bed


Matilda Velvet Bed


Blue Bed


Luxury Flower Bed

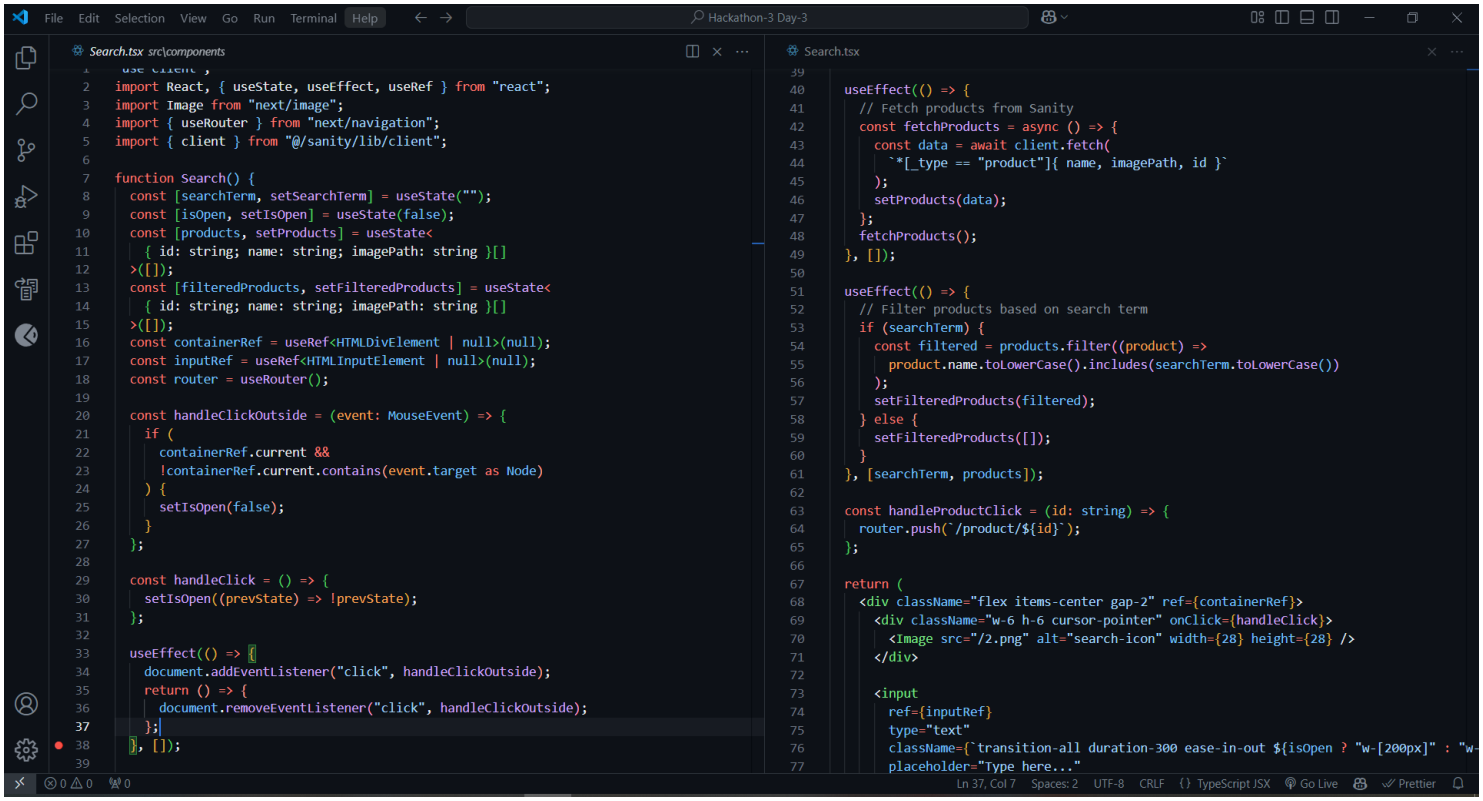

Solid Bed

Show


Matilda Velvet Bed


Red Bed

Product Search Bar (Code)



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.