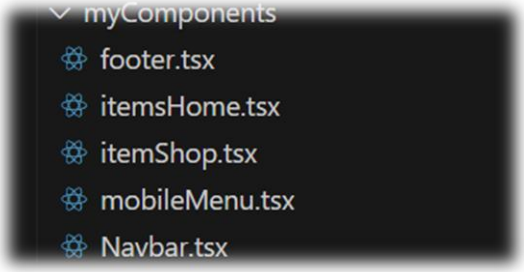
Day 4 - Building Dynamic Frontend Components for Your Marketplace

Welcome to Day 4 of the Hackathon! Today, I will be discussing the importance of modular and reusable components in building dynamic frontends. I'll also share snippets and screenshots of the dynamic and reusable components I've developed for my e-commerce furniture platform.

All of these components are designed to be responsive and user-friendly, ensuring a smooth experience for users on different devices. Let me first walk you through the key reusable components I've built for this project.

Key Components:

- **Navbar**
- **Mobile Menu**
- **Footer**
- **Items Shop**
- **Items Home**



In addition to these, I've also utilized several components from the Shadcn library, which has helped speed up development and ensure consistency across the platform.

How I've Used These Components:

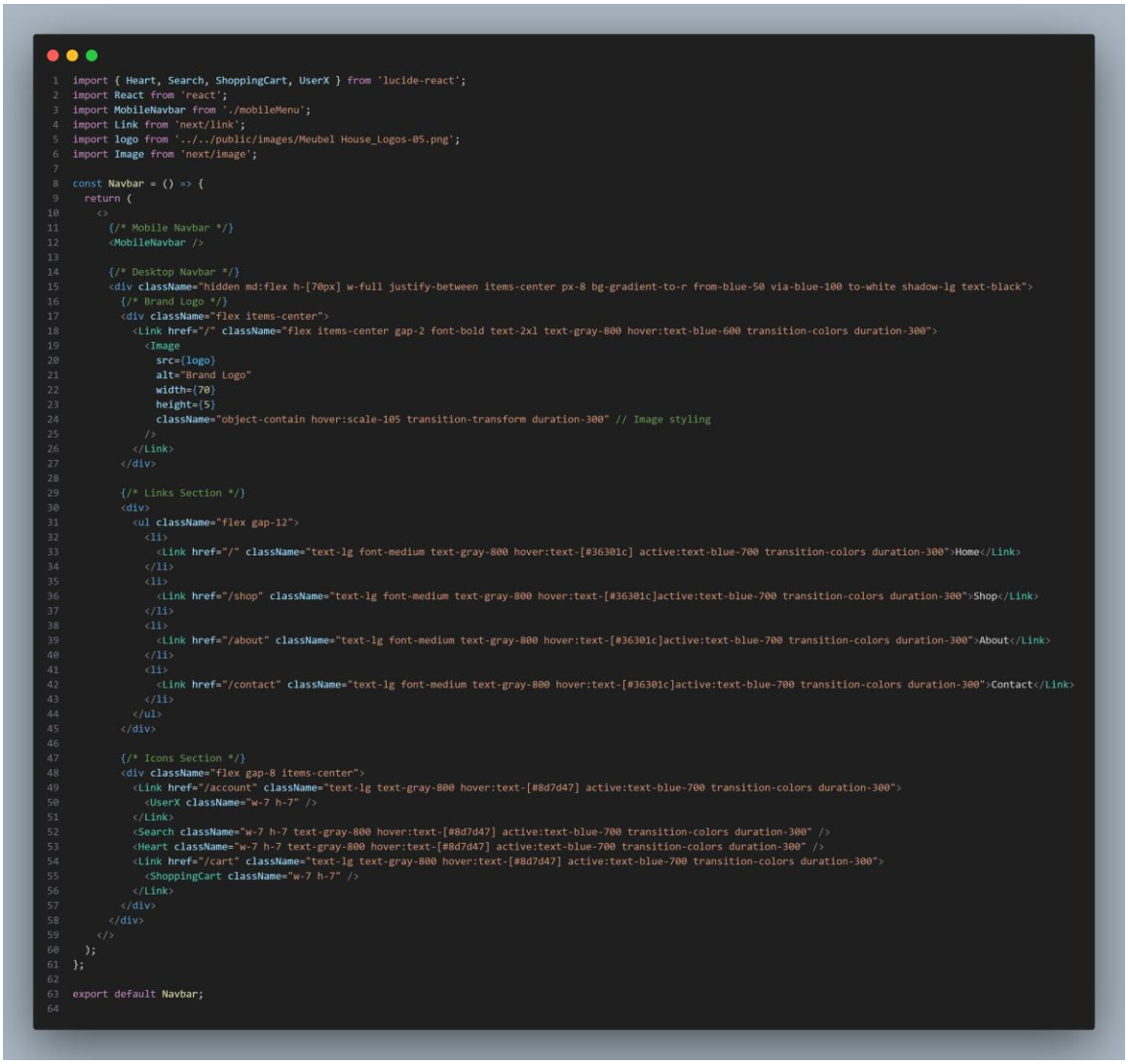
Let's start with the Navbar component.

For the **Navbar**, I've created a fully responsive design placed in the myComponents folder within the app directory of the project. The navbar includes:

- A brand logo
- Links to other pages on the platform
- Icons imported from Lucide React for additional functionality

Each element in the navbar is carefully positioned to maintain a clean layout, making it easy to navigate while ensuring the component adapts smoothly across devices.

here is the snapshot of the code from Navbar component:

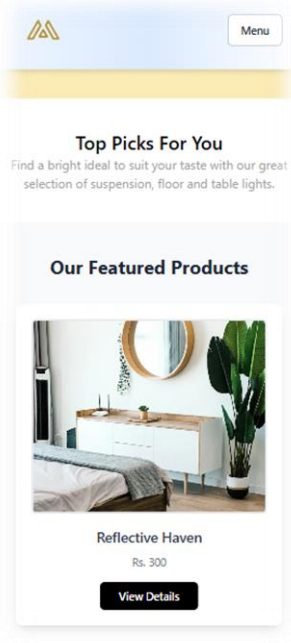


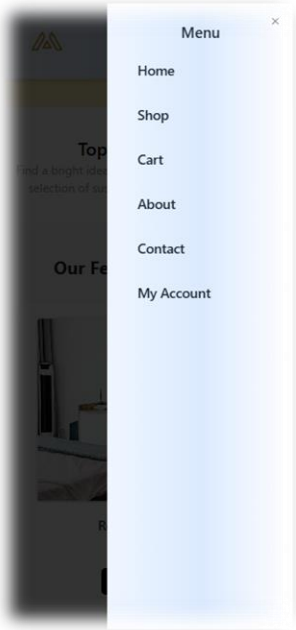
```tsx
import { Heart, Search, ShoppingCart, UserX } from 'lucide-react';
import React from 'react';
import MobileNavbar from './mobileMenu';
import Link from 'next/link';
import logo from '../../public/images/Meubel House_Logos-05.png';
import Image from 'next/image';

const Navbar = () => {
  return (
    <>
      {/* Mobile Navbar */}
      <MobileNavbar />

      {/* Desktop Navbar */}
      <div className="hidden md:flex h-[70px] w-full justify-between items-center px-8 bg-gradient-to-r from-blue-50 via-blue-100 to-white shadow-lg text-black">
        {/* Brand Logo */}
        <div className="flex items-center">
          <Link href="/" className="flex items-center gap-2 font-bold text-2xl text-gray-800 hover:text-blue-600 transition-colors duration-300">
            <Image
              src={logo}
              alt="Brand Logo"
              width={70}
              height={5}
              className="object-contain hover:scale-105 transition-transform duration-300" // Image styling
            />
          </Link>
        </div>

        {/* Links Section */}
        <div>
          <ul className="flex gap-12">
            <li>
              <Link href="/" className="text-lg font-medium text-gray-800 hover:text-[#36301c] active:text-blue-700 transition-colors duration-300">Home</Link>
            </li>
            <li>
              <Link href="/shop" className="text-lg font-medium text-gray-800 hover:text-[#36301c]active:text-blue-700 transition-colors duration-300">Shop</Link>
            </li>
            <li>
              <Link href="/about" className="text-lg font-medium text-gray-800 hover:text-[#36301c]active:text-blue-700 transition-colors duration-300">About</Link>
            </li>
            <li>
              <Link href="/contact" className="text-lg font-medium text-gray-800 hover:text-[#36301c]active:text-blue-700 transition-colors duration-300">Contact</Link>
            </li>
          </ul>
        </div>

        {/* Icons Section */}
        <div className="flex gap-8 items-center">
          <Link href="/account" className="text-lg text-gray-800 hover:text-[#8d7d47] active:text-blue-700 transition-colors duration-300">
            <UserX className="w-7 h-7" />
          </Link>
          <Search className="w-7 h-7 text-gray-800 hover:text-[#8d7d47] active:text-blue-700 transition-colors duration-300" />
          <Heart className="w-7 h-7 text-gray-800 hover:text-[#8d7d47] active:text-blue-700 transition-colors duration-300" />
          <Link href="/cart" className="text-lg text-gray-800 hover:text-[#8d7d47] active:text-blue-700 transition-colors duration-300">
            <ShoppingCart className="w-7 h-7" />
          </Link>
        </div>
      </div>
    </>
  );
};

export default Navbar;
```

Mobile Menu (Component):

The **Mobile Menu** is a specialized version of the **Navbar** component, designed specifically for mobile screens. As mentioned earlier, I imported the MobileNavbar from the mobileMenu component and rendered it above everything else to ensure it's placed correctly on mobile devices.

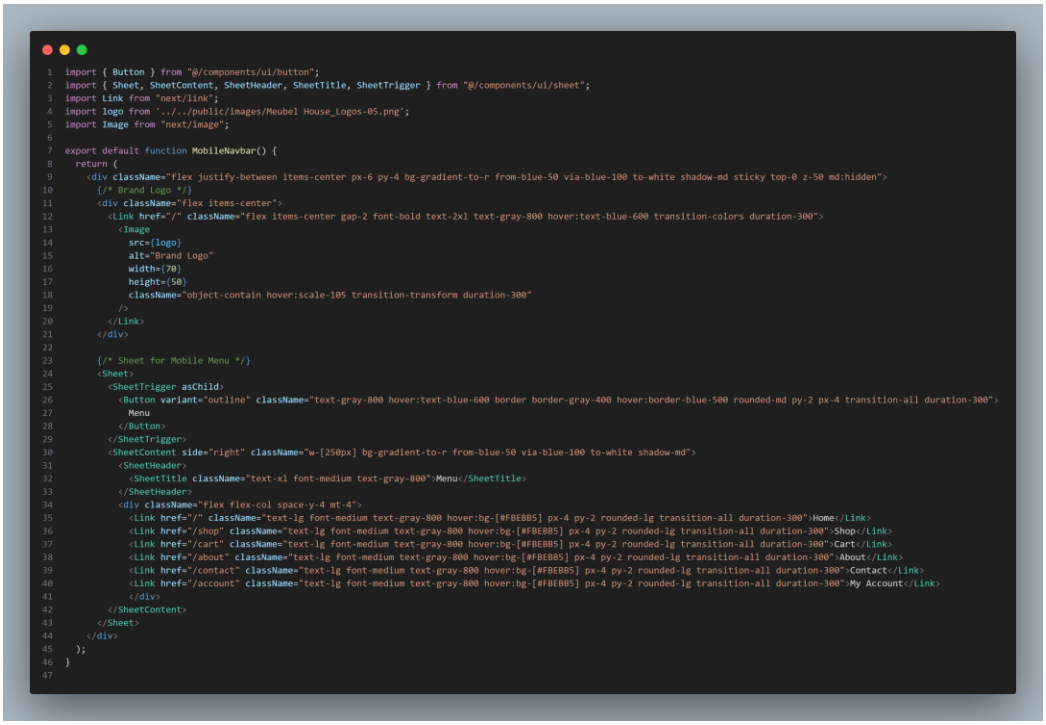
You might be wondering why I chose to create a separate mobile navbar instead of simply making the existing navbar responsive. Here's the reason: I wanted to use the **Sheet** component as the navbar on mobile screens, and it gives a sleek and modern feel. It really enhances the user experience, don’t you think? Check out the image below to see how it looks.





Nabar Close Navbar Open

Also here is the Screen shot of code:



```
import { Button } from "@/components/ui/button";
import { Sheet, SheetContent, SheetHeader, SheetTitle, SheetTrigger } from "@/components/ui/sheet";
import Link from "next/link";
import logo from '../../public/images/Meubel House_Logos-05.png';
import Image from "next/image";

export default function MobileNavbar() {
  return (
    <div className="flex justify-between items-center px-6 py-4 bg-gradient-to-r from-blue-50 via-blue-100 to-white shadow-md sticky top-0 z-50 md:hidden">
      {/* Brand Logo */}
      <div className="flex items-center">
        <Link href="/" className="flex items-center gap-2 font-bold text-2xl text-gray-800 hover:text-blue-600 transition-colors duration-300">
          <Image
            src={logo}
            alt="Brand Logo"
            width={70}
            height={50}
            className="object-contain hover:scale-105 transition-transform duration-300"
          />
        </Link>
      </div>

      {/* Sheet for Mobile Menu */}
      <Sheet>
        <SheetTrigger asChild>
          <Button variant="outline" className="text-gray-800 hover:text-blue-600 border border-gray-400 hover:border-blue-500 rounded-md py-2 px-4 transition-all duration-300">
            Menu
          </Button>
        </SheetTrigger>
        <SheetContent side="right" className="w-[250px] bg-gradient-to-r from-blue-50 via-blue-100 to-white shadow-md">
          <SheetHeader>
            <SheetTitle className="text-xl font-medium text-gray-800">Menu</SheetTitle>
          </SheetHeader>
          <div className="flex flex-col space-y-4 mt-4">
            <Link href="/" className="text-lg font-medium text-gray-800 hover:bg-[#FBEBB5] px-4 py-2 rounded-lg transition-all duration-300">Home</Link>
            <Link href="/shop" className="text-lg font-medium text-gray-800 hover:bg-[#FBEBB5] px-4 py-2 rounded-lg transition-all duration-300">Shop</Link>
            <Link href="/cart" className="text-lg font-medium text-gray-800 hover:bg-[#FBEBB5] px-4 py-2 rounded-lg transition-all duration-300">Cart</Link>
            <Link href="/about" className="text-lg font-medium text-gray-800 hover:bg-[#FBEBB5] px-4 py-2 rounded-lg transition-all duration-300">About</Link>
            <Link href="/contact" className="text-lg font-medium text-gray-800 hover:bg-[#FBEBB5] px-4 py-2 rounded-lg transition-all duration-300">Contact</Link>
            <Link href="/account" className="text-lg font-medium text-gray-800 hover:bg-[#FBEBB5] px-4 py-2 rounded-lg transition-all duration-300">My Account</Link>
          </div>
        </SheetContent>
      </Sheet>
    </div>
  );
}
```

Footer (Component):

Let’s move on to the **Footer** component. It’s a clean and straightforward design that includes:

- Links to important pages, such as **About Us**, **Contact**, and **Privacy Policy**
- Additional elements to provide users with relevant information

Here is the code for the Footer component:

```jsx
import Link from 'next/link';
import React from 'react';

const Footer = () => {
  return (
    <>
      {/* Footer Main Section */}
      <div className="w-full bg-white text-black py-16 px-6 md:px-12 lg:px-24">
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-12">
          {/* Address Section */}
          <div className="flex items-center">
            <p className="text-[#9F9F9F] text-sm">
              G223 M.T Khan Road <br />
              SultanAbad Karachi,<br />
              Pakistan

            </p>
          </div>

          {/* Links Section */}
          <div>
            <h1 className="mb-6 text-sm font-bold text-[#9F9F9F]">Links</h1>
            <ul className="space-y-4">
              <li className="text-sm">
                <Link href="/" className="text-sm hover:text-blue-400 transition-colors duration-300">Home</Link>
              </li>
              <li className="text-sm">
                <Link href="/shop" className="text-sm hover:text-blue-400 transition-colors duration-300">Shop</Link>
              </li>
              <li className="text-sm">
                <Link href="/about" className="text-sm hover:text-blue-400 transition-colors duration-300">About</Link>
              </li>
              <li className="text-sm">
                <Link href="/contact" className="text-sm hover:text-blue-400 transition-colors duration-300">Contact</Link>
              </li>
            </ul>
          </div>


          {/* Help Section */}
          <div>
            <h1 className="mb-6 text-sm font-bold text-[#9F9F9F]">Help</h1>
            <ul className="space-y-4">
              <li className="text-sm">Payment Options</li>
              <li className="text-sm">Return</li>
              <li className="text-sm">Privacy Policies</li>
            </ul>
          </div>

          {/* Newsletter Section */}
          <div>
            <h1 className="mb-6 text-sm font-bold text-[#9F9F9F]">Newsletter</h1>
            <div classname="flex">
              <input
                type="text"
                placeholder="Enter Your Email Address"
                classname="border-b-2 border-black px-2 py-1 text-sm focus:outline-none mr-2"
              />
              <button classname="border-b-2 border-black text-sm px-4 py-1">
                SUBSCRIBE
              </button>
            </div>
          </div>
        </div>
      </div>

      {/* Footer Bottom Section */}
      <div classname="w-full bg-white border-t border-gray-200">
        <div classname="py-6 items-center text-center text-sm text-gray-500">
          <p>2022 Meubel House. All rights reserved.</p>
        </div>
      </div>
    </>
  );
};

export default Footer;

```

Item Shop (Component):

Next, let's discuss the **Item Shop** component. This is one of the main components in the project, responsible for:

- Fetching data from **Sanity**
- Rendering the data dynamically on the **Shop** page

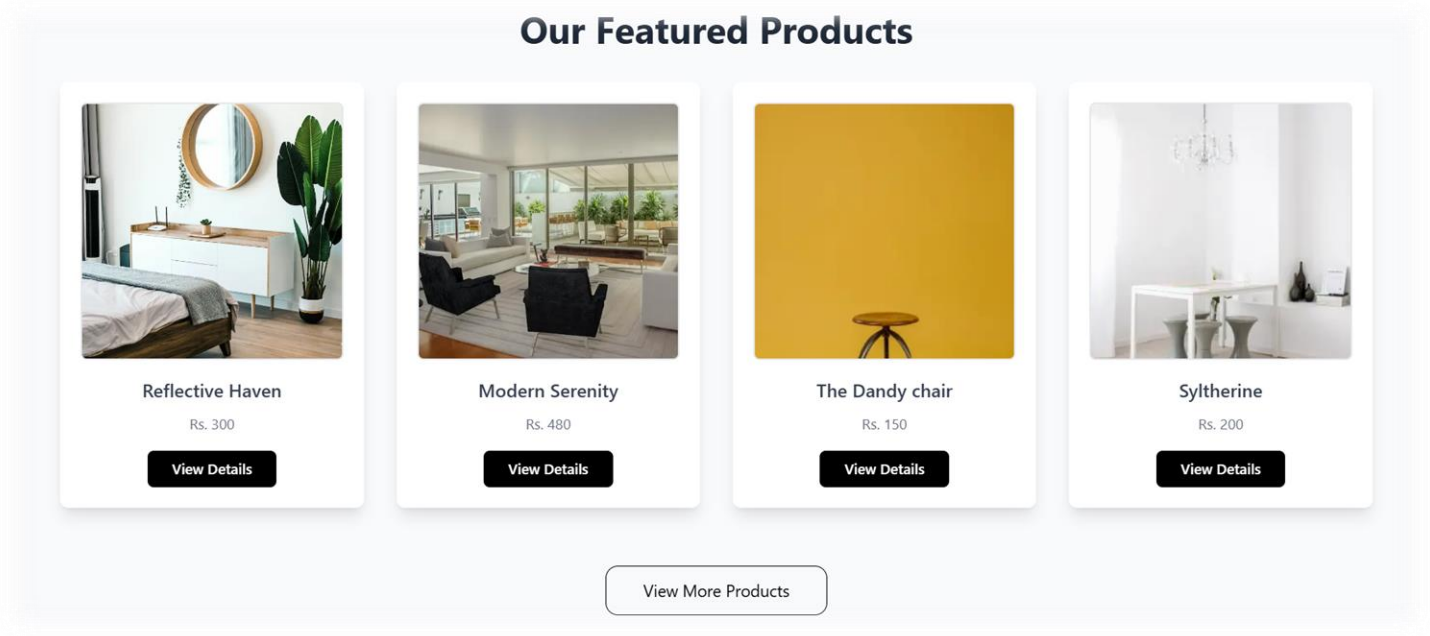
This component also includes the API integration functionality I implemented on **Day 3** of the hackathon. In that task, I posted data from an API and published it to Sanity. Here, I fetched that data from Sanity, processed it, and rendered it using a map function to display each item as a card. Below is a screenshot of the code for this component.

```tsx
import React, { useEffect, useState } from 'react';
import Link from 'next/link';
import { client } from '../../sanity/lib/client';// is cliet ko commen kr do jab api se data fetch krna ho.
import Image from 'next/image';
import imageUrlBuilder from '@sanity/image-url';

//posting content on sanity from the api
//this code will be commented after it it is used once.


// import { createClient } from '@sanity/client';

// const client = createClient({
//   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
//   dataset: 'production',
//   useCdn: true,
//   apiVersion: '2025-01-13',
//   token: process.env.SANITY_API_TOKEN,
// });



// async function uploadImageToSanity(imageUrl: string) {
//   try {
//     console.log(`Uploading image: ${imageUrl}`);

//     const response = await fetch(imageUrl);
//     if (!response.ok) {
//       throw new Error(`Failed to fetch image: ${imageUrl}`);
//     }

//     const buffer = await response.arrayBuffer();
//     const bufferImage = Buffer.from(buffer);

//     const asset = await client.assets.upload('image', bufferImage, {
//       filename: imageUrl.split('/').pop(),
//     });

//     console.log(`Image uploaded successfully: ${asset._id}`);
//     return asset._id;
//   } catch (error) {
//     console.error('Failed to upload image:', imageUrl, error);
//     return null;
//   }
// }

// interface Product {
//     title: string;
//     description: string;
//     imageUrl: string;
//     price: number;
//     tags?: string[]; // optional
//     discountPercentage?: number; // optional
//     isNew?: boolean; // optional
//   }


// async function uploadProduct(product: Product) {
//   try {
//     const imageId = await uploadImageToSanity(product.imageUrl);

//     if (imageId) {
//       const document = {
//         _type: 'product',
//         title: product.title,
//         price: product.price,
//         productImage: {
//           _type: 'image',
//           asset: {
//             _ref: imageId,
//           },
//         },
//         tags: product.tags,
//         discountPercentage: product.discountPercentage,
//         description: product.description,
//         isNew: product.isNew,
//       };

//       const createdProduct = await client.create(document);
//       console.log(`Product ${product.title} uploaded successfully: `, createdProduct);
//     } else {
//       console.log(`Product ${product.title} skipped due to image upload failure.`);
//     }
//   } catch (error) {
//     console.error('Error uploading product:', error);
//   }
// }

// async function importProducts() {
//   try {
//     const response = await fetch('https://template6-six.vercel.app/api/products');

//     if (!response.ok) {
//       throw new Error(`HTTP error! Status: ${response.status}`);
//     }

//     const products = await response.json();

//     for (const product of products) {
//       await uploadProduct(product);
//     }
//   } catch (error) {
//     console.error('Error fetching products:', error);
//   }
// }

// importProducts();



// Creating the image URL builder
const builder = imageUrlBuilder(client);

function urlFor(source: any) {
    // Check if source exists and is valid
    if (source) {
        return builder.image(source);
    }
    return null; // Return null if the source is invalid...
}

interface Product {
    _id: string;
    title: string;
    price: number;
    description: string;
    productImage: any; // Sanity image field (updated from 'image' to 'productImage')
}

const ItemsPage = () => {
    const [products, setProducts] = useState<Product[]>([]);

    useEffect(() => {
        const fetchProducts = async () => {
            // Fetch products and their productImage from sanity...
            const data = await client.fetch(
                `*[_type == "product"][_id, title, price, description, productImage}`
            );
            setProducts(data);
        };

        fetchProducts();
    }, []);

    const truncateDescription = (description: string, length: number) => {
        if (description.length > length) {
            return description.substring(0, length) + '...';
        }
        return description;
    };

    return (
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-8 p-8">
            {products.map((product) => (
                <Link key={product._id} href={`/product/${product._id}`}>
                    <div className="bg-white shadow rounded p-4">
                        <Image
                            src={urlFor(product.productImage)?.width(500).url() || '/images/default-image.jpg'} // Correct path to the default image in the public folder
                            alt={product.title}
                            width={500}
                            height={300}
                            className="w-full h-40 object-cover rounded"
                            priority
                        />
                        <h2 className="text-lg font-bold mt-2">{product.title}</h2>
                        <p className="text-gray-500">{truncateDescription(product.description, 100)}</p>
                        <p className="text-gray-700 font-semibold">${product.price}</p>
                    </div>
                </Link>
            ))}
        </div>
    );
};

export default ItemsPage;
```

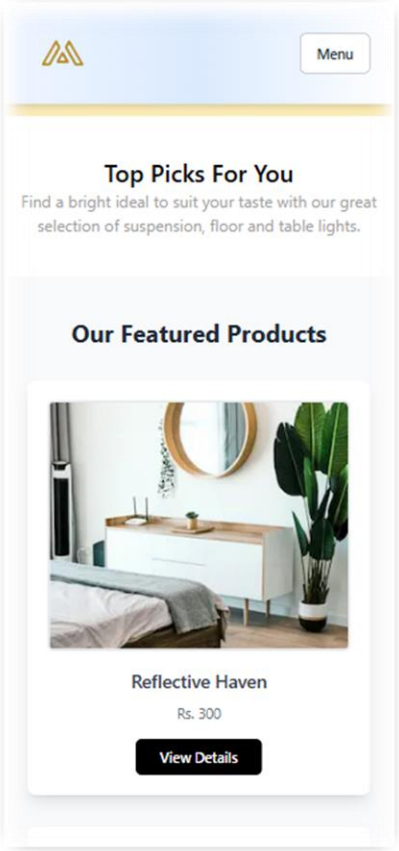
Items Home (Component):

The Items Home component is similar to the Item Shop component, with one key distinction: it fetches data from Sanity but only renders the first four products.

This component is specifically designed to showcase the latest products on the Home page, as outlined in the Figma template provided to me. The data is fetched seamlessly and rendered in a visually appealing manner, as shown below.



And on mobile it looks like this:



You can view the code below.

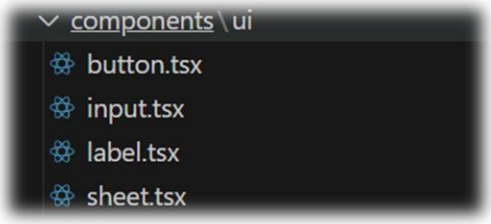
```tsx
'use client'
import React, { useEffect, useState } from 'react';
import Image from 'next/image';
import Link from 'next/link';
import { client } from '@/sanity/lib/client';

interface Product {
  _id: string;
  title: string;
  price: number;
  productImage: string;
}

export default function ItemsPage() {
  const [products, setProducts] = useState<Product[]>([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchProducts = async () => {
      const fetchedProducts: Product[] = await client.fetch(
        `*[_type == "product"][0..3]{_id, title, price, "productImage": productImage.asset->url}`
      );
      setProducts(fetchedProducts);
      setLoading(false);
    };

    fetchProducts();
  }, []);

  if (loading) {
    return <div className="text-center py-10">Loading products...</div>;
  }

  return (
    <div className="w-full px-6 py-10 md:px-12 lg:px-16 bg-gray-50">
      <h2 className="text-center text-2xl md:text-4xl font-bold text-gray-800 mb-8">
        Our Featured Products
      </h2>

      {/* Items List */}
      <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-8">
        {products.map((product) => (
          <div
            key={product._id}
            className="flex flex-col items-center bg-white shadow-lg p-5 rounded-lg hover:shadow-xl transform hover:scale-105 transition duration-300 ease-in-out"
          >
            <Image
              src={product.productImage}
              alt={product.title}
              className="h-[250px] w-full object-cover rounded-md border-2 border-gray-200"
              width={250}
              height={250}
            />
            <h4 className="text-center mt-4 font-semibold text-lg text-gray-700">
              {product.title}
            </h4>
            <p className="text-center text-gray-500 mt-2 text-sm">
              Rs. {product.price.toLocaleString()}
            </p>
            <Link
              href={`/product/${product._id}`}
              className="mt-4 bg-black text-white px-6 py-2 rounded-md text-sm font-medium hover:bg-gray-800 transition duration-300"
            >
              View Details
            </Link>
          </div>
        ))}
      </div>

      {/* View More Link */}
      <div className="mt-10 flex justify-center">
        <Link
          href="/"
          className="flex items-center justify-center w-full sm:w-[215px] h-[48px] border border-black rounded-xl mt-4
               hover:bg-black hover:text-white transition duration-300 ease-in-out transform hover:scale-105"
        >
          View More Products
        </Link>
      </div>
    </div>
  );
}

```

ShadCN Components:

That concludes the key components I built for my project. Now, let me share the components I imported from **ShadCN**.

I've utilized several pre-built components, including **Button**, **Input**, **Label**, and **Sheet**. These components have been seamlessly integrated into various parts of the e-commerce furniture platform, enhancing both functionality and design.

components folder is shown below.



Cart Context (Dynamic Component):

Another essential dynamic component in my project is the **Cart Context**.

What does this context do?

The **Cart Context** manages the functionality of adding and removing products dynamically from the cart.

- When a user clicks the "Add to Cart" button for a product, the product is instantly added to the cart.
- Similarly, users can remove items from the cart on a dedicated **Cart Page**, which is also dynamic and updates in real-time as products are removed.

Here's the code for the **Cart Context** on next page:

```tsx
'use client';

import React, { createContext, useContext, useState } from 'react';

interface CartItem {
    _id: string;
    title: string;
    price: number;
    imageUrl: string;
    quantity: number;
}

interface CartContextType {
    cart: CartItem[];
    addToCart: (item: CartItem) => void;
    removeFromCart: (id: string) => void;
}

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

export const CartProvider = ({ children }: { children: React.ReactNode }) => {
    const [cart, setCart] = useState<CartItem[]>([]);

    const addToCart = (item: CartItem) => {
        setCart((prevCart) => {
            const existingItem = prevCart.find((cartItem) => cartItem._id === item._id);
            if (existingItem) {
                return prevCart.map((cartItem) =>
                    cartItem._id === item._id
                        ? { ...cartItem, quantity: cartItem.quantity + item.quantity }
                        : cartItem
                );
            }
            return [...prevCart, item];
        });
    };

    const removeFromCart = (id: string) => {
        setCart((prevCart) => prevCart.filter((cartItem) => cartItem._id !== id));
    };

    return (
        <CartContext.Provider value={{ cart, addToCart, removeFromCart }}>
            {children}
        </CartContext.Provider>
    );
};

export const useCart = () => {
    const context = useContext(CartContext);
    if (!context) {
        throw new Error('useCart must be used within a CartProvider');
    }
    return context;
};
```

Dynamic Product Page (product/[id]/page.tsx):

Another important dynamic page in my project is the **Dynamic Product Page**, located at product/[id]/page.tsx.

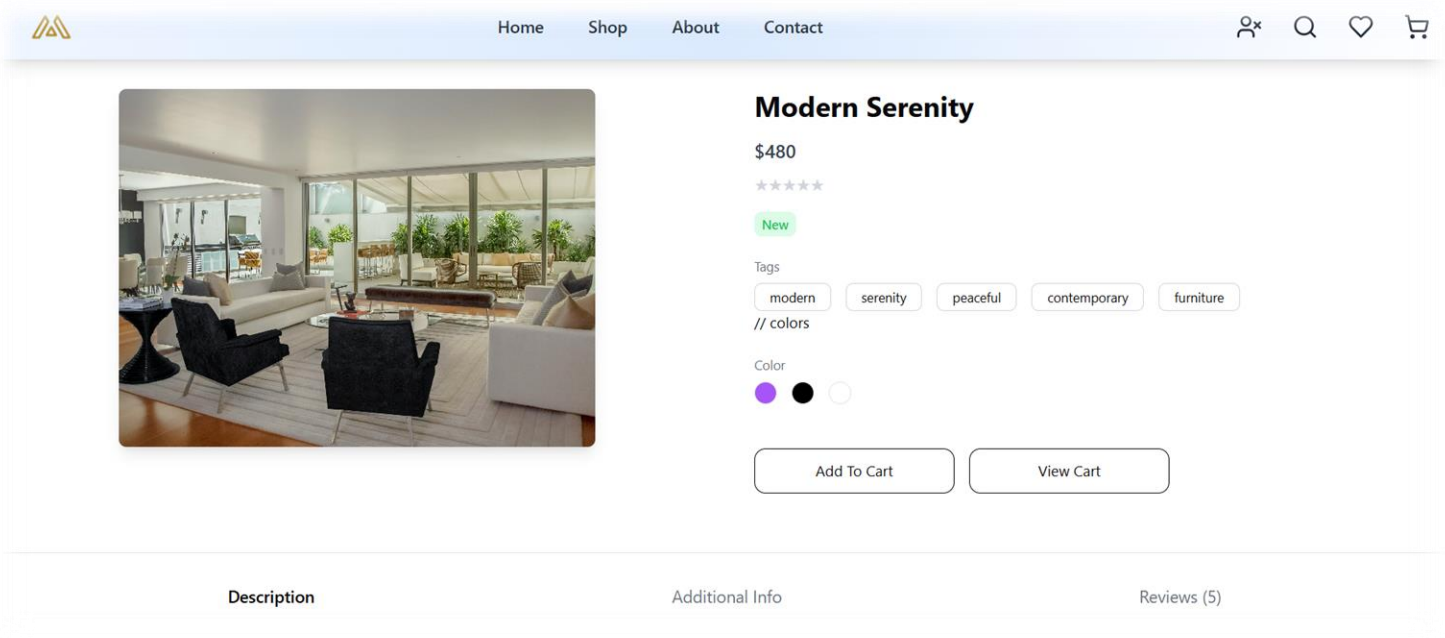
What Does This Page Do?

This page dynamically renders the details of a selected product fetched from **Sanity**. Whenever I click on a product, this page displays its information, including:

- **Name**
- **Image**
- **Price**
- **Tags**
- **Description**
- **New Product Status (is new or not)**

Additionally, users can add the product to their cart directly from this page.

View the component below:



Below is the code for this dynamic product page:

```tsx
import Navbar from '@/app/myComponents/Navbar';
import Footer from '@/app/myComponents/Footer';
import { client } from '../../../sanity/lib/client';
import { notFound } from 'next/navigation';
import CartButton from './CartButton';
import Link from 'next/link';

interface Product {
    _id: string;
    title: string;
    price: number;
    description: string;
    productImage: string;
    rating: number;
    tags: string[];
    discountPercentage?: number;
    isNew?: boolean;
    imageUrl: string;
}

async function getProductData(id: string): Promise<Product> {
    const data = await client.fetch(
        `*[_type == "product" && _id == $id][
            _id,
            title,
            price,
            description,
            "productImage": productImage.asset->url,
            rating,
            tags,
            discountPercentage,
            isNew
        ]`,
        { id }
    );

    if (data.length === 0) {
        return notFound();
    }

    return data[0];
}

const ProductPage = async ({ params }: { params: { id: string } }) => {
    const { id } = params;
    const product = await getProductData(id);

    const renderStars = (rating: number) => {
        return Array(5)
            .fill(0)
            .map((_, i) => (
                <span
                    key={i}
                    className={`inline-block text-lg ${i < rating ? 'text-yellow-500' : 'text-gray-300'}`}
                >
                    ★
                </span>
            ));
    };

    return (
        <div>
            <Navbar />
            <div className="flex flex-col lg:flex-row p-8 gap-8 lg:gap-16">
                {/* Image Section */}
                <div className="lg:w-1/2 flex justify-center">
                    <img
                        src={product.productImage}
                        alt={product.title}
                        className="w-full max-w-lg h-96 object-cover rounded-lg shadow-lg"
                    />
                </div>

                {/* Product Info Section */}
                <div className="lg:w-1/2">
                    <h1 className="text-3xl font-bold mb-4">{product.title}</h1>
                    <p classNam="text-xl font-semibold text-gray-700 mb-2">${product.price}</p>
                    <div className="flex items-center mb-4">{renderStars(product.rating)}</div>

                    {/* Discount Percentage */}
                    {product.discountPercentage && (
                        <p className="text-lg text-red-500 font-medium">
                            {product.discountPercentage}% off
                        </p>
                    )}

                    {/* New Badge */}
                    {product.isNew && (
                        <span className="text-sm font-semibold text-green-500 bg-green-100 px-2 py-1 rounded-lg">
                            New
                        </span>
                    )}

                    {/* Tags */}
                    {product.tags?.length > 0 && (
                        <div className="mt-6">
                            <h2 className="text-sm text-gray-500 mb-2">Tags</h2>
                            <div classNam="flex gap-4">
                                {product.tags.map((tag) => (
                                    <span
                                        key={tag}
                                        className="border border-gray-300 px-4 py-1 rounded-lg text-sm"
                                    >
                                        {tag}
                                    </span>
                                ))}
                            </div>
                        </div>
                    )}

                    {/* Sizes */}
                    {/* <div className="mt-6">
                        <h2 className="text-sm text-gray-500 mb-2">Size</h2>
                        <div classNam="flex gap-4">
                            {['S', 'M', 'L', 'XL'].map((size) => {
                                <label
                                    key={size}
                                    htmlFor={size}
                                    className="flex items-center gap-2 cursor-pointer"
                                >
                                    <input
                                        type="radio"
                                        id={size}
                                        name="size"
                                        value={size}
                                        className="hidden"
                                    />
                                    <span classNam="border px-4 py-1 rounded-lg text-sm">
                                        {size}
                                    </span>
                                </label>
                            })
                        </div>
                    </div> */}

                    // colors
                    <div classNam="mt-6">
                        <h2 className="text-sm text-gray-500 mb-2">Color</h2>
                        <div classNam="flex gap-4">
                            {[
                                { name: 'Purple', color: 'bg-purple-500' },
                                { name: 'Black', color: 'bg-black' },
                                { name: 'Peach', color: 'bg-peachpuff' },
                            ].map(({ name, color }) => {
                                <label
                                    key={name}
                                    htmlFor={name}
                                    classNam="flex items-center gap-2 cursor-pointer"
                                >
                                    <input
                                        type="radio"
                                        id={name}
                                        name="color"
                                        value={name}
                                        className="hidden"
                                    />
                                    <span
                                        classNam={`w-6 h-6 rounded-full border ${color}`}
                                        title={name}
                                    ></span>
                                </label>
                            })
                        </div>
                    </div>


                    {/* Buttons */}
                    <div classNam="mt-8 flex gap-4">
                        <CartButton product={product} />
                        <Link
                            href="/cart"
                            classNam="flex items-center justify-center w-full sm:w-[215px] h-[48px] border border-black rounded-xl mt-4
                            hover:bg-black hover:text-white transition duration-300 ease-in-out transform hover:scale-105"
                        >
                            View Cart
                        </Link>
                    </div>
                </div>
            </div>

            <hr classNam="my-8" />

            {/* Additional Info */}
            <div classNam="px-8 lg:px-16">
                <div classNam="flex justify-around mb-8">
                    <p classNam="text-black text-lg font-medium cursor-pointer">Description</p>
                    <p classNam="text-gray-500 text-lg cursor-pointer">Additional Info</p>
                    <p classNam="text-gray-500 text-lg cursor-pointer">Reviews (5)</p>
                </div>
                <div classNam="text-gray-600 leading-relaxed">
                    {/* Description */}
                    <p classNam="mt-6 text-gray-700">{product.description}</p>
                </div>
            </div>

            <Footer />
        </div>
    );
}

export default ProductPage;
```

Technical Report for E-Commerce Furniture Platform

1. Steps Taken to Build and Integrate Components

- **Planning and Design:**
  The first step was to understand the Figma design and break it down into key components, such as Navbar, Mobile Menu, Footer, Item Shop, and Items Home. I also identified reusable components to streamline development.
  I ensured the design was responsive and created components that aligned with the requirements.

- **Building Core Components:**

  - **Navbar & Mobile Menu:**
    I created a responsive **Navbar** component that adjusts based on screen size. A mobile version of the navbar was developed as a separate **Mobile Menu** component, using **Shadcn** for the sheet-style mobile menu.

  - **Footer:**
    A simple, responsive footer component was created with links to important pages (e.g., About Us, Contact).

  - **Item Shop & Items Home:**
    The **Item Shop** component was built to fetch data from **Sanity** and render product cards. The **Items Home** component was designed to show the latest products by fetching the first four items from **Sanity** and displaying them on the homepage.

- **Integration with Sanity:** I integrated **Sanity CMS** to manage product data. Products were fetched from Sanity and dynamically rendered on the site using React's state management. I also handled the API calls for data fetching and rendering via functions like getStaticProps or getServerSideProps to ensure dynamic updates (in which I also faced some error but fortunately I was able to solve them before the end time).

- **Cart Context:**
  I built a **Cart Context** using React's Context API to manage the cart state. This allows products to be added or removed from the cart dynamically. The cart page updates in real-time as products are added or removed.

2. Challenges Faced and Solutions Implemented

- **Challenge: Ensuring Responsiveness:**
  Making the design responsive, especially the navbar and mobile menu, was a challenge. The Figma design didn't specify different screen sizes, so I had to adjust and optimize components for various breakpoints.
  Solution:
  I used **Tailwind CSS** for its flexibility and utility-first approach to create responsive layouts without cluttering the code.

- **Challenge: Data Fetching from Sanity:**
  Initially, I faced issues with fetching the correct product data dynamically from **Sanity** and ensuring that data was rendered properly in the components.
  Solution:

I leveraged the **Sanity API** effectively and used getServerSideProps or getStaticProps to fetch product data based on product IDs. This helped me ensure that each product page displayed the correct content based on the URL.

- **Challenge: Managing Dynamic Cart State:**
  Managing the cart state dynamically across different components, while ensuring that the state remained consistent, posed a challenge.
  Solution:
  I utilized the **React Context API** for global state management of the cart, allowing easy updates and access across the app. I also implemented custom functions to handle adding/removing products from the cart.

Challenge & Solution : Issues with API Integration into Sanity

While searching for solutions on how to push data from an API into **Sanity**, I came across a method where I could create a separate folder for API integration. The process involved converting the TypeScript files into JavaScript using the command `tsc filename` and then running the JavaScript file with `node filename`. However, I encountered an error during implementation.

Upon further research, I discovered that I could create custom commands in **package.json** by defining a name and assigning the command as its value. Running `npm run <command-name>` would execute the corresponding command. Unfortunately, the separate folder approach didn't work as expected, so I had to explore alternative solutions.

I also installed **dotenv** to manage environment variables, but this approach didn't resolve the issue either. As a last resort, I decided to paste the API integration function directly into the **Item Shop** page and ran it. However, I still faced the error message indicating an invalid token.

After multiple attempts and troubleshooting, I realized that the issue stemmed from using the wrong token type. I had been using a "developer" token instead of an "editor" token. Once I switched to the correct token, everything worked as expected, and the data was successfully imported into Sanity.

3. Best Practices Followed During Development

- **Component Reusability:**
  I followed the principle of **modular and reusable components** to keep the codebase clean and maintainable. Components like Navbar, Footer, and Item Shop were designed to be reusable across multiple pages, making the code more scalable.

- **Responsive Design:**
  I adhered to **responsive design principles**, ensuring that the application looked good on different devices. I used **Tailwind CSS** for mobile-first design and leveraged its utility classes to implement breakpoints and responsiveness.

- **State Management with Context API:**
  The **React Context API** was used for state management, particularly for handling the cart. This approach made the state global and easily accessible, avoiding prop drilling and keeping the codebase manageable.

- **Optimized API Integration:**
  I followed best practices for API integration by fetching data efficiently using **getServerSideProps** and **getStaticProps** to ensure dynamic and optimized page rendering.

- **Code Quality:**
  I maintained clean, modular code and used **TypeScript** for type safety, ensuring that all components and functions had clear and defined types, which helped avoid errors during development.

Future Plans for Dynamic Components

In the future, I plan to develop several additional dynamic components to enhance the functionality and user experience of my project. My e-commerce platform still requires several key features, and I aim to integrate them progressively. Below are the planned components:

1. **Search Bar:**
   I will implement a dynamic search bar to allow users to search for products more effectively. Currently, this feature is not functional due to time constraints.

2. **Wishlist:**
   A dynamic wishlist component will be added, enabling users to save products they're interested in for future reference. This feature will enhance the shopping experience and increase user engagement.

3. **User Profile:**
   Once I integrate login and logout functionality, I will add a **User Profile** component. This will allow users to view and manage their profile details after logging in.

4. **Reviews and Ratings:**
   I plan to implement a reviews and ratings system for products, where users can provide feedback on their purchased items. This component will be dynamic, allowing real-time updates and interaction.

5. **Related Products:**
   A **Related Products** component will be introduced to recommend similar products to users based on their browsing history or the product they're currently viewing. This dynamic feature will increase cross-selling opportunities.

6. **Pagination:**
   For handling large product listings, I will add a **Pagination** component. This will allow users to navigate through pages of products without overwhelming the page load time.

7. **Multi-Language Support:**
   I plan to introduce multi-language support for the platform. I've previously implemented this feature in another project, the **Job Portal** (https://job-portal-website-project.vercel.app/), using **i18next**—a powerful internationalization library for JavaScript. This will allow users to switch between languages for a more personalized experience.

8. **Order Tracking:**
   Although I haven't implemented it yet due to time constraints, I plan to develop an **Order Tracking** feature in the future. This will enable users to track their orders in real time, enhancing customer satisfaction.

Conclusion:

And that wraps up Day 4 of the hackathon! Thank you for reading through the process. It was both enjoyable and a valuable learning experience. I've gained new insights and overcome challenges that helped improve my skills in building dynamic components. I'm excited to continue refining the project and implementing more features in the coming days.

Thank you for your time and support!

Name: Okasha Nadeem
Roll number: 99316