HACKATHON-03

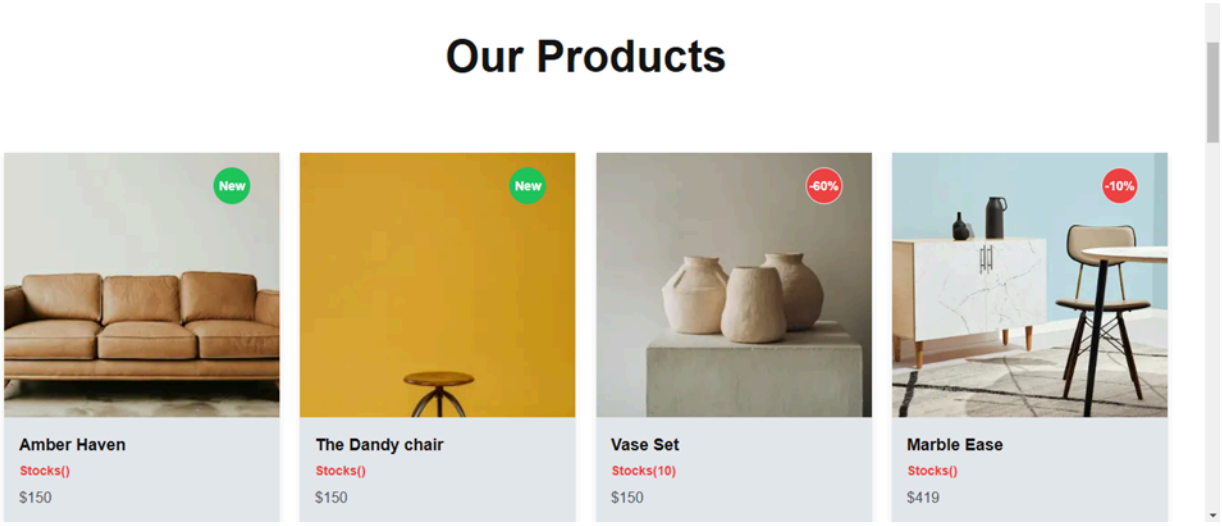
DAY-04

Dynamic Frontend Components

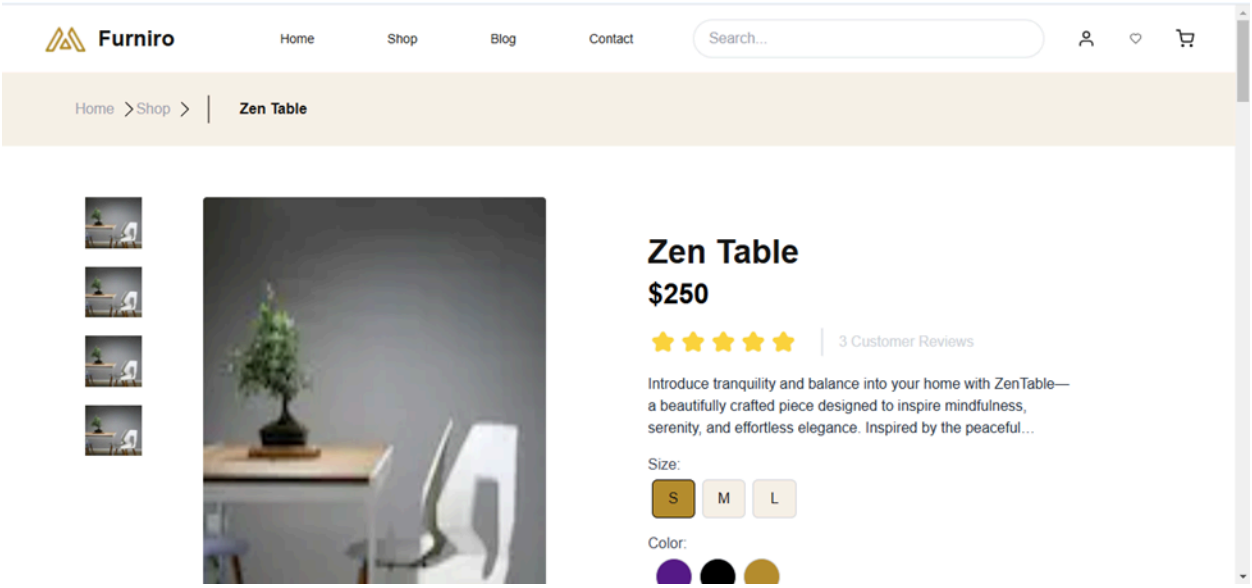
My MarketPlace Name [Honest_Bazar]

1. Functional Deliverables

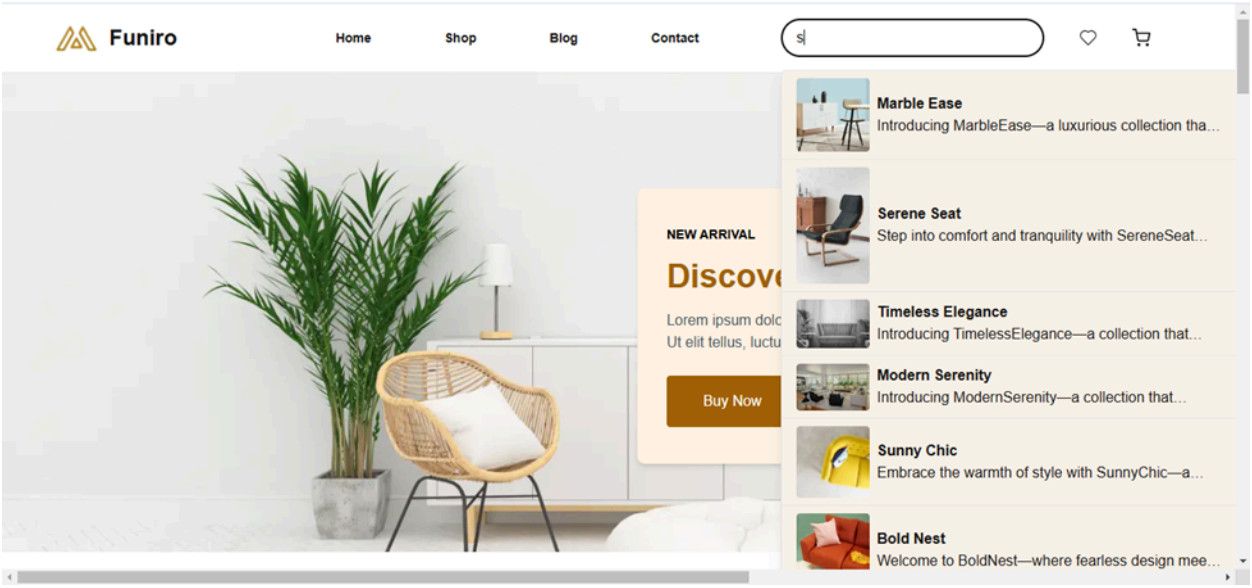
The product listing page with dynamic data.



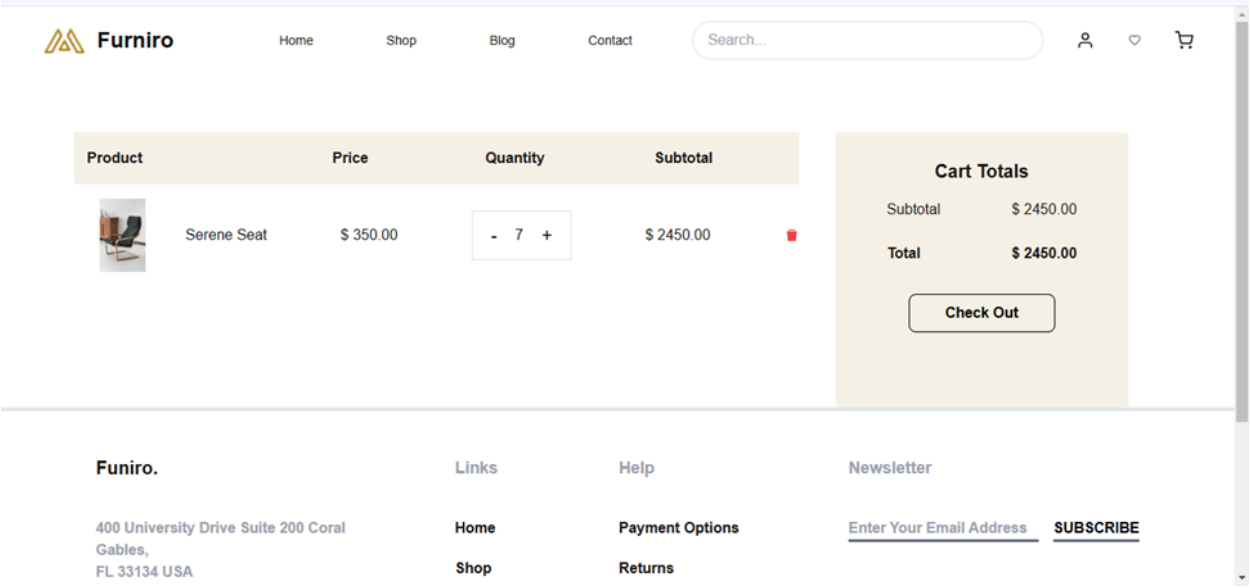
Individual product detail pages with accurate routing and data rendering.



Working category filters, search bar, and pagination.



Cart page with dynamic price and cart totals items and price.



Code Deliverables:

Product listing page

```
import Link from "next/link";

import React from "react";

import Image from "next/image";

import { CiHeart, CiShare2 } from "react-icons/ci";

import { MdCompareArrows } from "react-icons/md";

import { client } from "@/sanity/lib/client";

import CartIcon from "./cardIcon";

import { urlFor } from "@/sanity/lib/image";



type Slug = {

  current: string;
```

```
};



const Products = async () => {

  const query = `*[_type=="product"]{

  _id,

    title,

   "imageUrl" :productImage.asset -> url,

    price,

    tags,

    dicountPercentage,

    description,

    isNew,

    stocks,

    slug,

    productImage

}`;

  const res = await client.fetch(query);

  return (

    <>

      <h1 className="text-center text-3xl md:text-5xl font-bold my-10">

        Our Products

      </h1>

      <div className="p-4 px-10 my-16">

        <div className="flex  flex-wrap gap-5 justify-around  ">

          {res

            .slice(0, 8)

            .map(

              (product: {
```

```
                _id: string;

                title: string;

                imageUrl: string;

                price: number;

                tags: string[];

                dicountPercentage: number;

                description: string;

                isNew: boolean;

                stocks: number;

                slug: Slug;

              }) => (

                <div key={product._id} className="relative group">

                  <Link href={`/singlepage/${product.slug.current}`}>

                    <div className="hidden group-hover:block  sm:w-[250px] md:w-[300px]
lg:w-[300px] duration-500 z-20 w-full h-full justify-center absolute    ">

                      <div className="w-full h-full top-0  bg-gray-900  opacity-50"></div>



                      <div className=" absolute z-30 top-0 w-full h-full">

                        <button

                          id={product._id}

                          className="bg-white text-yellow-700 font-bold py-3 text-2xl  mt-28
sm:ml-10 ml-5   px-7"

                        >

                          <CartIcon id={product._id} />

                        </button>



                        <div className="flex mt-10 justify-around ">

                          <button className=" text-white font-bold gap-2 flex items-center
opacity-100">
```

```
                          <CiShare2 /> Share

                        </button>

                        <button className=" flex text-white font-bold gap-2 items-center ">

                          {" "}

                          <MdCompareArrows /> Compare

                        </button>

                        <button className=" flex text-white font-bold gap-2 items-center ">

                          {" "}

                          <CiHeart /> Like

                        </button>

                      </div>

                    </div>

                  </div>



                  <div className="bg-gray-200 w-[250px] sm:w-[250px] md:w-[300px]
lg:w-[300px]  shadow-md overflow-hidden">

                    <div className="relative">

                      <Image

                        width={330}

                        height={330}

                        src={urlFor(product.imageUrl).url()}

                        alt={product.title}

                        className="w-full h-72 object-cover"

                      />

                      {product.dicountPercentage && (

                        <div className="absolute top-0 right-8 mt-4 border rounded-full
bg-red-500 text-white h-10 w-10 flex justify-center items-center text-sm font-bold">

                          -{product.dicountPercentage}%

                        </div>
```

```
                        )}

                        {product?.isNew && (

                          <div className="absolute top-0 right-8 mt-4  rounded-full
bg-green-500 text-white h-10 w-10 text-sm font-bold flex justify-center items-center">

                            New

                          </div>

                        )}

                      </div>

                      <div className="p-4">

                        <h3 className="text-lg font-bold mb-1">

                          {product.title}

                        </h3>

                        <p className="text-sm line-clamp-2 gap-5 text-red-500 font-bold mb-2">

                          Stocks({product.stocks})

                        </p>

                        <div className="flex justify-between">

                          <p className="text-gray-600">

                            ${product.price.toLocaleString("id-ID")}

                          </p>

                        </div>

                      </div>

                    </div>

                  </Link>

                </div>

              )

            )}

          </div>
```

```
        <div className="flex justify-center mt-10">

          <Link href="/shop">

          <button className="bg-white border border-[#B88E2F] text-[#B88E2F] font-bold py-3
px-10 rounded">

              Add to Cart

            </button>

          </Link>

        </div>

      </div>

    </>

  );

};



export default Products;
```

Detailpage:

Page.tsx:

```
import React from 'react'
import { SlArrowRight } from "react-icons/sl";
import { PiLineVerticalThin } from "react-icons/pi";
import ProductDetails from './details';
import ProductDescription from './describtion';
import ProductCard from '@/components/products';
import Link from 'next/link';
import { client } from '@/sanity/lib/client';
async function page({
  params,
}: {
  params: Promise<{ slug: string }>
}) {
```

```
  const slug = (await params).slug
  const product = await client.fetch (`*[_type=="product" && slug.current == $slug][0]`, {slug})
  if(product) {
    return (
      <div>
      <div className="flex items-center w-full h-20 sm:pl-20 pl-0   bg-[#F9F1E7] ">
        <Link href="/">
    <h1  className='flex  gap-2     text-gray-400  items-center'>Home<SlArrowRight  className=  '  gap-3
text-black'/></h1>
        </Link>
        <Link href="/shop">
    <h1  className='flex  gap-2    text-gray-400  items-center'>Shop  <SlArrowRight  className='text-black
gap-3'/></h1> </Link>
    <h1  className='flex  gap-4  font-bold  items-center    '>   <PiLineVerticalThin  className='text-black
text-4xl'/>  {product.title}</h1>


      </div>
      <ProductDetails props={product}/>

      <ProductDescription/>


     <ProductCard  />


      </div>
    )
  }else{
    return <h1>Product not found</h1>
  }
}

export default page
```

details.tsx:

```
"use client";

import React, { useState } from "react";
import Image from "next/image";
import { PiLineVerticalThin } from "react-icons/pi";
import { FaFacebook } from "react-icons/fa";
import { FaLinkedin } from "react-icons/fa6";
import { IoLogoTwitter } from "react-icons/io5";
import Link from "next/link";
import { urlFor } from "@/sanity/lib/image";

// Define the Product type to ensure proper typing
interface Product {
```

```
  title: string;
  price: number;
  quantity: number;
  sizes: string[];
  productImage: string;
  thumbnails: string[];
  rating: number;
  reviews: number;
  description: string;
  sku: string;
  category: string;
  tags: string[];
}

const ProductDetails = ({ props }: { props: Product }) => {
  const initialSize = props.sizes && props.sizes.length > 0 ? props.sizes[0] : 'default-size';
  const [quantity, setQuantity] = useState(props.quantity || 1);
  const [selectedSize, setSelectedSize] = useState(initialSize);
  const [totalPrice, setTotalPrice] = useState(props.price * (props.quantity || 1));
  const sizePriceMap: { [key: string]: number } = {
    [props.sizes[0]]: 1,
    [props.sizes[1]]: 1.5,
    [props.sizes[2]]: 2,
  };

  const updatePrice = (newSize: string) => {
    const multiplier = sizePriceMap[newSize];
    setSelectedSize(newSize);
    setTotalPrice(multiplier * props.price * quantity);
  };

  const handleQuantityChange = (type: "increase" | "decrease") => {
    if (type === "increase") {
      setQuantity((prev) => {
        const newQuantity = prev + 1;
        setTotalPrice(sizePriceMap[selectedSize] * props.price * newQuantity);
        return newQuantity;
      });
    } else if (type === "decrease" && quantity > 1) {
      setQuantity((prev) => {
        const newQuantity = prev - 1;
        setTotalPrice(sizePriceMap[selectedSize] * props.price * newQuantity);
        return newQuantity;
      });
    }
  };

  return (
    <div className="container mx-auto p-4">
      <div className="flex flex-col lg:flex-row">
        <div className="flex items-center justify-between lg:items-start lg:w-1/2">
          <div className="flex mt-10 flex-col gap-5 w-[90px] sm:ml-10 ml-1 ">
            {props.thumbnails.map((item, index) => (
              <Image
                key={index}
                src={urlFor(item).url()}
```

```
                width={200}
                height={200}
                alt={props.title}
              />
            ))}
          </div>
          <Image
            src={urlFor(props.productImage).url()}
            width={100}
            height={50}
            alt={props.title}
            className="w-[400px]  sm:mr-20 mt-10 mr-10 h-[80vh] mb-4 rounded"
          />
        </div>
        <div className="lg:w-1/2 mt-20 lg:pl-8">
          <h1 className="sm:text-4xl text-2xl font-bold mb-2">{props.title}</h1>
          <div className="text-3xl font-bold text-black mb-4">
            ${totalPrice.toLocaleString()}
          </div>
          <div className="flex items-center mb-4">
            <span className="text-yellow-500 text-2xl">
              {`⭐`.repeat(Math.floor(props.rating))}
            </span>
            <span className="ml-2 text-gray-300 items-center flex">
              <PiLineVerticalThin className="text-gray-300 text-4xl" />{" "}
              {props.reviews} Customer Reviews
            </span>
          </div>
          <p className="text-gray-800 line-clamp-3 mb-4 pr-32">{props.description}</p>
          <div className="mb-4">
            <h1 className="text-gray-700">Size:</h1>
            {props.sizes.map((size, index) => (
              <button
                key={index}
                className={`mt-1 mx-1 sm:w-[8%] w-[13%] py-2 px-3 border ${
                  selectedSize === size
                    ? "border-black bg-[#B88E2F]"
                    : "border-gray-300 bg-[#F9F1E7]"
                } rounded-md`}
                onClick={() => updatePrice(size)}
              >
                {size}
              </button>
            ))}
          </div>
          <div className="mb-4 space-x-2">
            <h1 className="text-gray-700">Color:</h1>
                  <button  className="mt-1  w-10  h-10  py-2  px-3  border  border-gray-300  bg-purple-900
rounded-full"></button>
                    <button  className="mt-1  w-10  h-10  py-2  px-3  border  border-gray-300  bg-black
rounded-full"></button>
                  <button  className="mt-1  w-10  h-10  py-2  px-3  border  border-gray-300  bg-[#B88E2F]
rounded-full"></button>
          </div>
          <div className="mb-4 flex flex-wrap items-center gap-10">
```

```
                <div className="border py-3 w-32 flex justify-around items-center border-gray-300
rounded-md">
              <button
                onClick={() => handleQuantityChange("decrease")}
                className="text-2xl font-bold"
              >
                -
              </button>
              <span className="text-lg">{quantity}</span>
              <button
                onClick={() => handleQuantityChange("increase")}
                className="text-2xl font-bold"
              >
                +
              </button>
            </div>
            <div className="flex gap-6">
              <Link href="/cart">
                <button className="bg-white text-black py-3 px-6 rounded-md border border-black">
                  Add to Cart
                </button>
              </Link>
              <button className="bg-white text-black py-3 px-6 rounded-md border border-black">
                + Compare
              </button>
            </div>
          </div>
          <hr className="my-10" />
          <div className="mt-4">
            <div className="text-gray-400 items-center mt-2 ">
              SKU : <span className="pl-16"> {props.sku}</span>
            </div>
            <div className="text-gray-400 mt-2">
              Category: <span className="pl-12 "> {props.category}</span>
            </div>
            <div className="text-gray-400 mt-2">
              Tags: <span className="pl-20 "> {props.tags.join(", ")}</span>
            </div>
          </div>
          <div className="flex space-x-4">
            <h1 className="text-gray-400 mt-2">Share : </h1>
            <button className="pl-14">
              <FaFacebook />
            </button>
            <button className="">
              <FaLinkedin />
            </button>
            <button className="">
              <IoLogoTwitter />
            </button>
          </div>
        </div>
      </div>
      <hr className="my-10" />
    </div>
  );
```

```
};

export default ProductDetails;
```

REPORTS

Steps Taken:

Developed reusable components like ProductCard, ProductList, SearchBar, and Pagination to improve efficiency and consistency across the site

Integrated an API to fetch product data dynamically from the Sanity CMS, ensuring up-to-date information.

Set up dynamic routing for individual product pages, allowing users to view detailed information by accessing each product through a unique identifier (using [id]).

Best Practices:

Focused on creating reusable components to keep the code clean and easy to maintain.

Organized the project with a modular folder structure to make it easier to scale in the future.

Implemented lazy loading to improve performance and speed up the initial page load time.

Added toast notifications for the shopping cart to enhance the user experience with helpful alerts.

My Conclusion:

By following these best practices, the project ended up with a clean and efficient front-end setup. Reusable components and a well-organized folder structure

made it easier to maintain and scale the project. Performance improvements, like lazy loading and API caching, helped make the site faster and more user-friendly. Overall, these strategies not only made development smoother but also set the stage for future improvements.