BUILDING DYNAMIC FRONTEND COMPONENTS FOR YOUR MARKETPLACE

Project Name: Marketplace Ecommerce hackathon.

Author: Mohsin.

Table of Contents

1. Introduction

This document provides a comprehensive technical report on the development of a dynamic product listing system for the hackathon project. The project includes dynamic pages, filtering and search functionality, pagination, and API integration.

1. Functional Deliverables

Product Listing Page

A dynamic product listing page that fetches data from a backend (Sanity CMS).

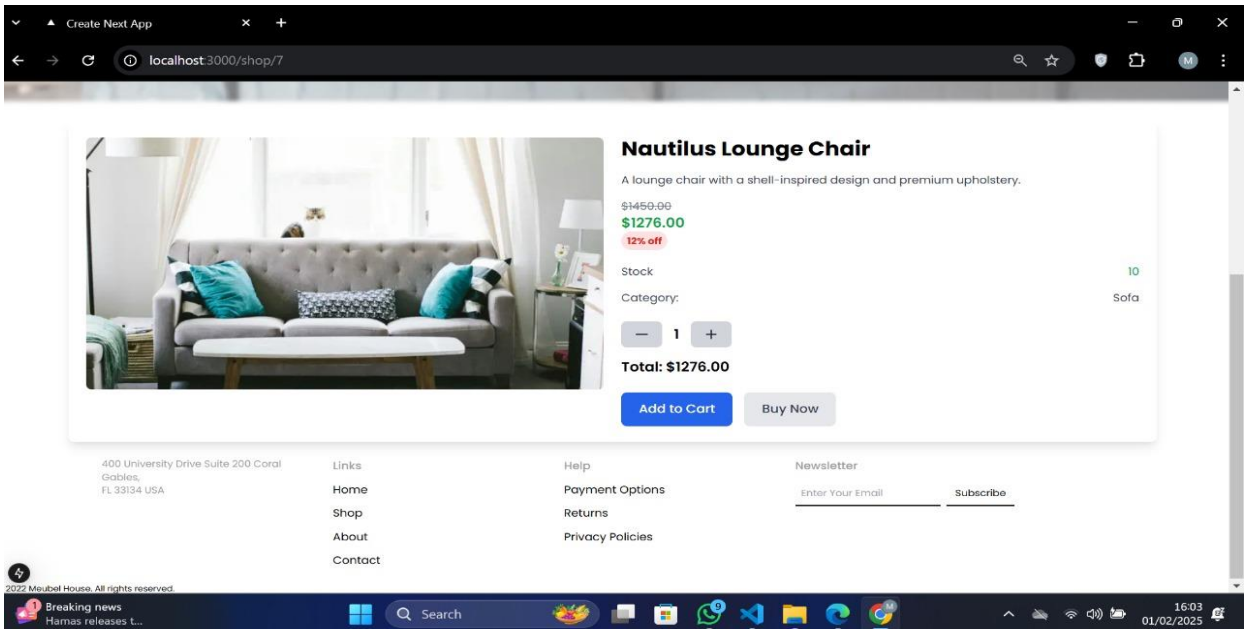
Each product is displayed using a Productcard component.

Pagination is implemented for seamless browsing.

Screenshots of Dynamic Product Page

Dynamic product pages are created using a [dynamic route] (e.g., /product/[id]).

Each page fetches product details based on the URL parameter.

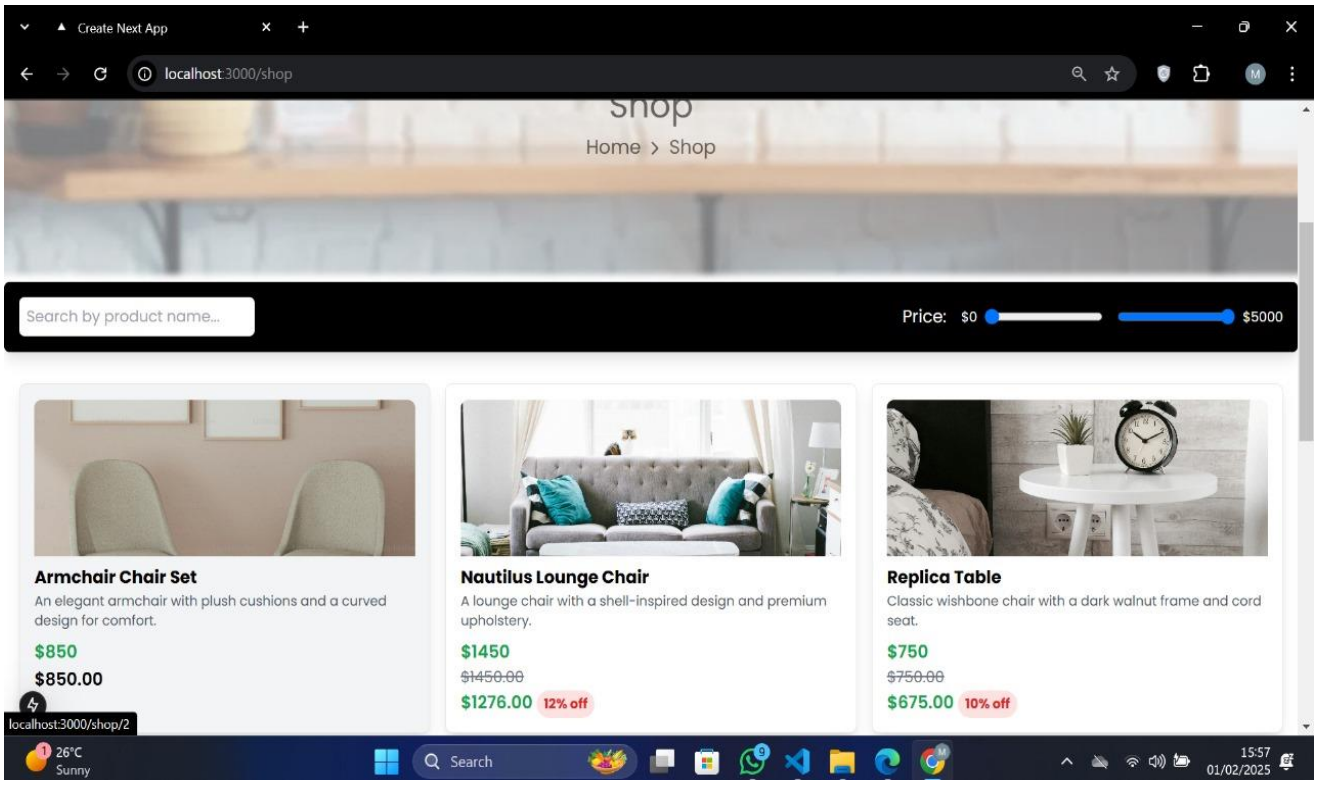


Filters & Search Bar

Users can filter products by category, price, and other attributes.

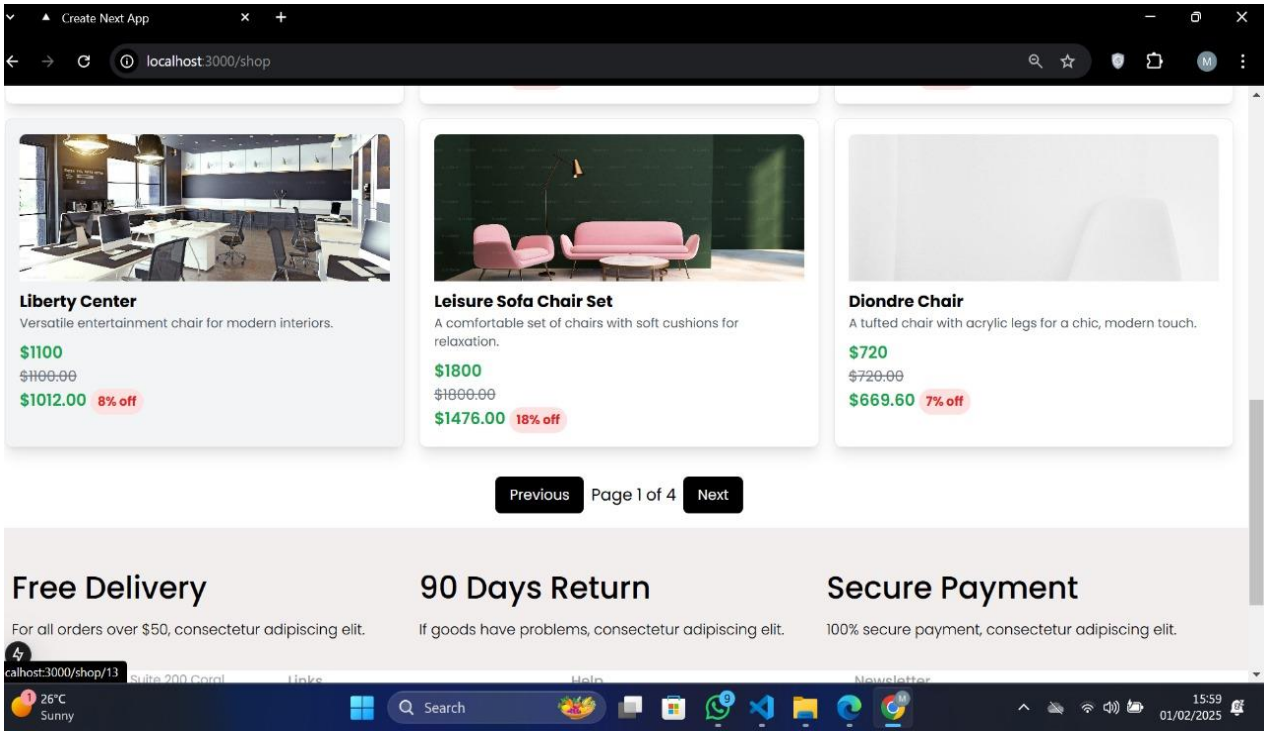
A SearchBar component enables users to search for products.

Related Products Section: Displays similar products based on category or tags.



Pagination

Pagination is implemented to enhance performance and user experience.



Code Deliverables

code snippet of product list component

```
"use client";
import { urlFor } from '@/sanity/lib/image';
import Image from 'next/image';
import React, { useState, useEffect } from 'react';
import { MdAdd,  MdOutlineHorizontalRule } from "react-icons/md";
import { useUser, useClerk } from "@clerk/nextjs";
import { addTocart } from '../action/action';
```

```
interface Product {
   id: string;
   name: string;
   description: string;
   price: number;
   discountPercentage?: number;
   category: string;
   stockLevel: number;
   imageUrl?: string;
}

export default function Productlist({ product }: { product: Product }) {
   const [quantity, setQuantity] = useState(1);
   const [totalPrice, setTotalPrice] = useState(0);

   const handleCartBtn = (e:any, product: Product) => {
      e.preventDefault()
      addTocart(product)

   }
   const { isSignedIn } = useUser();
   const { openSignUp } = useClerk(); // Function to open sign-up modal

   // ✅ Calculate discounted price per unit
   const discountedPrice = product.discountPercentage
      ? product.price * (1 - product.discountPercentage / 100)
      : product.price;

   // ✅ Update total price when quantity changes
   useEffect(() => {
      setTotalPrice(quantity * discountedPrice);
   }, [quantity, discountedPrice]);
```

```
const handleCheckOut = async () => {
    if (!isSignedIn) {
        openSignUp(); // Redirect user to sign-up page
        return;
    }

    try {
        const response = await fetch('/api/checkout', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ product, quantity })
        });
        const data = await response.json();
        window.location.href = data.url;
    } catch (error) {
        console.log(error);
    }
};

const increaseQuantity = () => {
    if (quantity < product.stockLevel) {
        setQuantity(prev => prev + 1);
    }
};

const decreaseQuantity = () => {
    if (quantity > 1) {
        setQuantity(prev => prev - 1);
    }
};
```

```
  return (
    <div className="container mx-auto p-6">
      <div className="grid md:grid-cols-2 gap-6 bg-white p-6 rounded-lg shadow-lg">
        {product.imageUrl && (
          <Image
            src={urlFor(product.imageUrl).url()}
            alt={product.name}
            width={500}
            height={384}
            className="w-full h-96 object-cover rounded-lg"
          />
        )}
        <div>
          <h1 className="text-3xl font-bold">{product.name}</h1>
          <p className="text-gray-700 mt-4">{product.description}</p>

          <div className="mt-4">
            {product.discountPercentage ? (
              <>
                <p className="text-gray-500 line-through">${product.price.toFixed(2)}</p>
                <p className="text-xl font-semibold text-green-
600">${discountedPrice.toFixed(2)}</p>
                <span className="bg-red-100 text-red-600 text-sm font-bold px-2 py-1 rounded-full">
                  {product.discountPercentage}% off
                </span>
              </>
            ) : (
              <p className="text-xl font-semibold">${product.price.toFixed(2)}</p>
            )}
          </div>

          <div className={flex justify-between mt-6 text-[16px] ${product.stockLevel > 0 ? "text-green-
600" : "text-red-600"}}>
```

```
            <span className="text-gray-700">Stock</span> {product.stockLevel > 0 ?
${product.stockLevel} : "Out of Stock"}
          </div>

          <div className="text-gray-700 mt-4 flex justify-between items-center">
            <div>Category:</div>
            <div>{product.category}</div>
          </div>

          {/* Quantity Selector */}
          <div className="flex items-center mt-6 space-x-4">
            <button
              onClick={decreaseQuantity}
              className="flex justify-center items-center bg-gray-300 hover:bg-gray-400 text-gray-800
px-4 py-2 rounded-lg text-lg font-medium transition"
              disabled={quantity === 1}
            >
              <MdOutlineHorizontalRule className='text-2xl' />
            </button>
            <span className="text-xl font-semibold">{quantity}</span>
            <button
              onClick={increaseQuantity}
              className="flex justify-center items-center bg-gray-300 hover:bg-gray-400 text-gray-800
px-4 py-2 rounded-lg text-lg font-medium transition"
              disabled={quantity >= product.stockLevel}
            >
              <MdAdd className="text-2xl" />
            </button>
          </div>

          {/* ✅ Total Price Display */}
          <div className="mt-4 text-xl font-semibold">
            Total: ${totalPrice.toFixed(2)}
          </div>
```

```
            <div className="flex space-x-4 mt-6">
              <button onClick={(e)=>handleCartBtn(e,product)} className="bg-blue-600 hover:bg-blue-
700 text-white px-6 py-3 rounded-lg text-lg font-medium transition">
                Add to Cart
              </button>
              <button onClick={handleCheckOut} className="bg-gray-200 hover:bg-gray-300 text-gray-
700 px-6 py-3 rounded-lg text-lg font-medium transition">
                {isSignedIn ? "Buy Now" : "Sign Up to Buy"}  </button>  </div>
          </div> </div>  </div>
  );
}
```

Code Snippet of dynamic page

```
import React from "react";
import { client } from "../../../sanity/lib/client";
import { urlFor } from "../../../sanity/lib/image";
import Navbar from "@/app/components/navbar";
import Footer from "@/app/components/footer";
import Mainbanner from "@/app/components/pagebanner";
import Productlist from "@/app/components/productlist";

interface Product {
 id: string;
 name: string;
 description: string;
 price: number;
 discountPercentage?: number;
 category: string;
 stockLevel: number;
 imageUrl?: string;
}
```

```
const Page = async ({ params}:{params:{id:string}}) => {
  const {id}= params;
  const query = `*[_type == "product" && id == $id]{
    "id":_id,
    name,
    description,
    price,
    "imageUrl": image.asset->url,
    category,
    discountPercentage,
    stockLevel}[0]`;
    const product:Product | null = await client.fetch(query, { id });
    if(!product){
      return(
        <div>Product Not Found</div>
      )
    }

    return(

<div>
        <Navbar />
        <Mainbanner pageName="Shop" />
        <Productlist product={product} />
        <Footer />
</div>
    )
}
export default Page;
```

4. Development Process

1. Project Setup: Initialized the project and set up the folder structure.

2. Data Fetching: Integrated Sanity CMS and fetched product data.

3. Component Creation: Developed Productcard, Productlist, and Searchbar.

4. Routing & Navigation: Implemented dynamic routing for product details.

5. Filtering & Searching: Created filtering and search functionality.

6. Pagination: Implemented pagination for better navigation.

7. Testing & Debugging: Ensured the data renders correctly and tested all features.

Challenges Faced & Solutions

Best Practices Followed

Component Reusability: Used modular and reusable components.

Optimized API Calls: Limited unnecessary API requests.

Efficient State Management: Managed state effectively using React hooks.

User Experience: Ensured smooth navigation and minimal load time.

5. Conclusion

This project successfully implemented a dynamic product listing system with filtering, searching, and pagination. The integration of Sanity CMS for data management and efficient routing enhances the user experience. Future