Day 4 - Dynamic Frontend Components - [FURNITURE STORE]

Technical Report

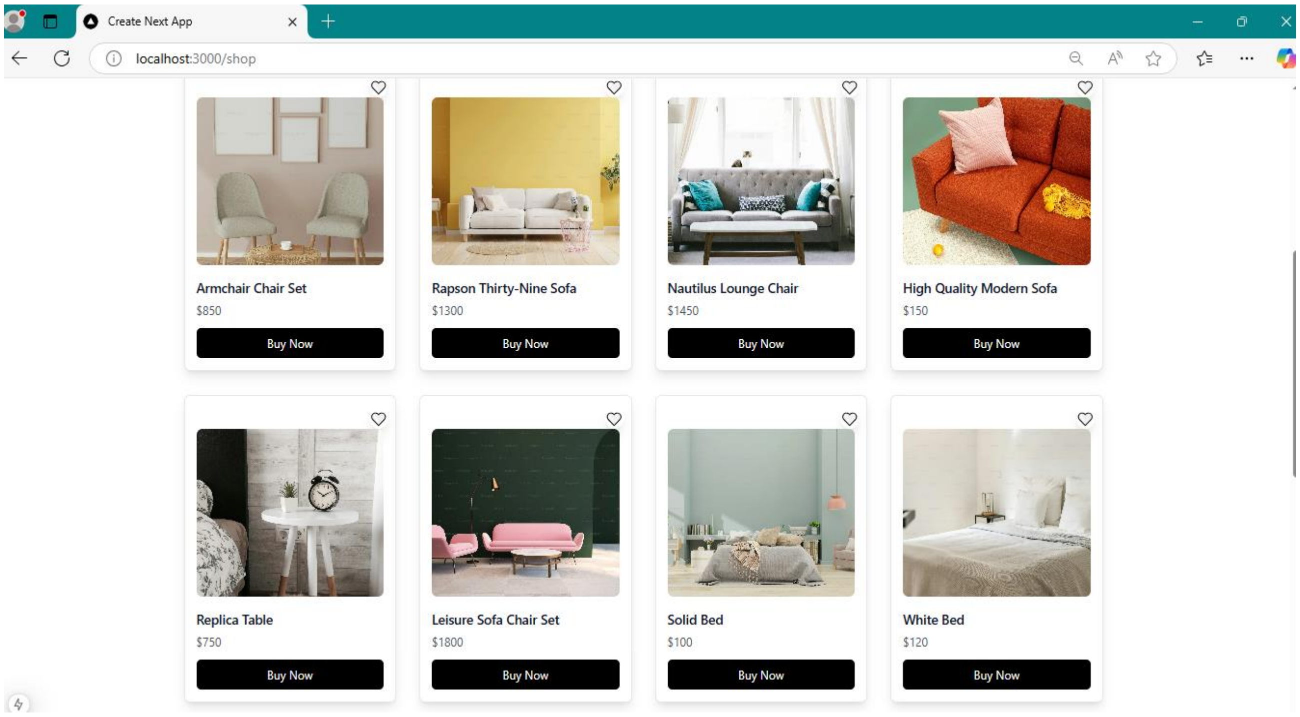
Introduction

This document provides a summary of the development process for implementing dynamic frontend components in the marketplace project. The focus of this stage was on creating a dynamic product listing page, individual product detail pages, category filters, a search bar, and pagination.

Development Process

1. Product Listing Page with Dynamic Data

The product listing page was designed to fetch and display product data dynamically. Each product is represented using a structured card format, ensuring a user-friendly interface.



Implementation Steps:

Integrated an API to fetch real-time product data.

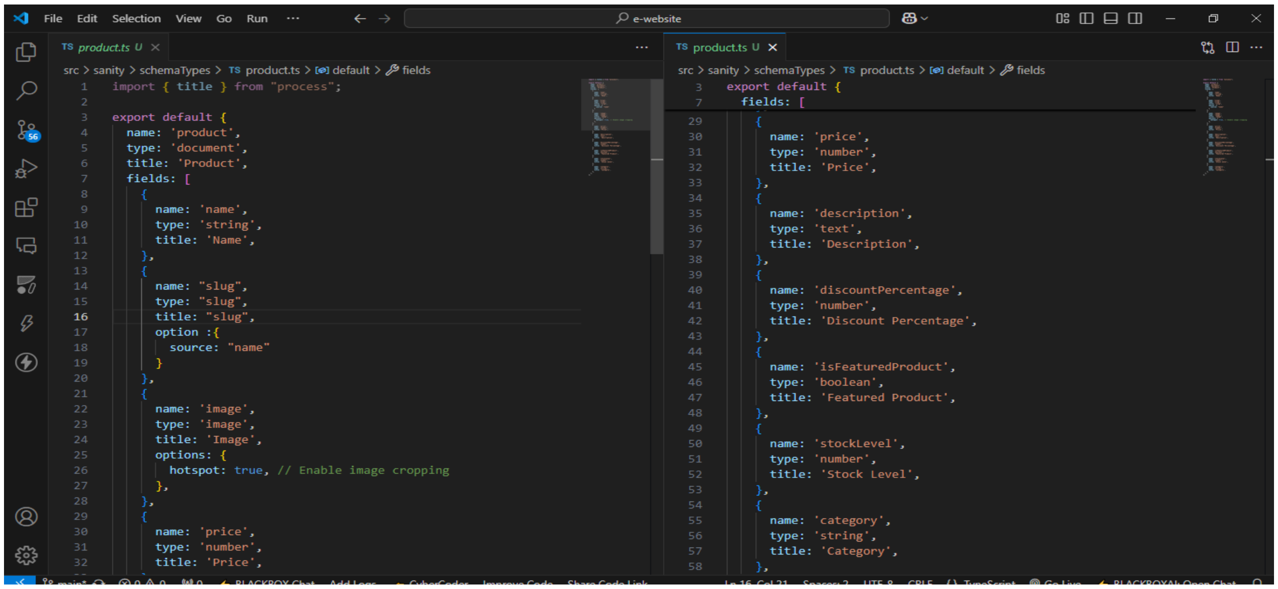
Utilized dynamic rendering to display product information.

Designed the product cards for an optimized user experience.

Implemented responsive styling to ensure consistency across devices.

2. Individual Product Detail Pages

Each product detail page was developed to display comprehensive information about a selected product, including its images, description, price, and related items.


src > sanity > schemaTypes > product.ts > default > fields

```typescript
import { title } from "process";

export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Name',
    },
    {
      name: "slug",
      type: "slug",
      title: "slug",
      option :{
        source: "name"
      }
    },
    {
      name: 'image',
      type: 'image',
      title: 'Image',
      options: {
        hotspot: true, // Enable image cropping
      },
    },
    {
      name: 'price',
      type: 'number',
      title: 'Price',
```

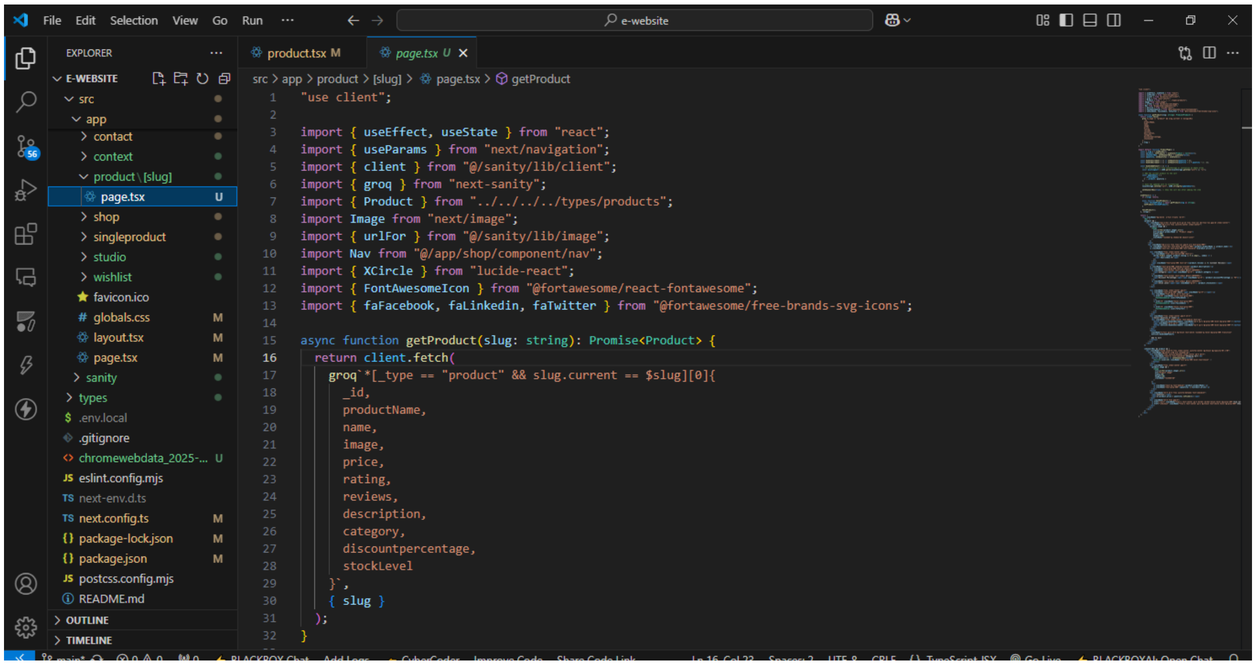
src > sanity > schemaTypes > product.ts > default > fields

```typescript
export default {
  fields: [
    {
      name: 'price',
      type: 'number',
      title: 'Price',
    },
    {
      name: 'description',
      type: 'text',
      title: 'Description',
    },
    {
      name: 'discountPercentage',
      type: 'number',
      title: 'Discount Percentage',
    },
    {
      name: 'isFeaturedProduct',
      type: 'boolean',
      title: 'Featured Product',
    },
    {
      name: 'stockLevel',
      type: 'number',
      title: 'Stock Level',
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
    },
```

Implementation Steps:

Established dynamic routing to ensure accurate navigation.

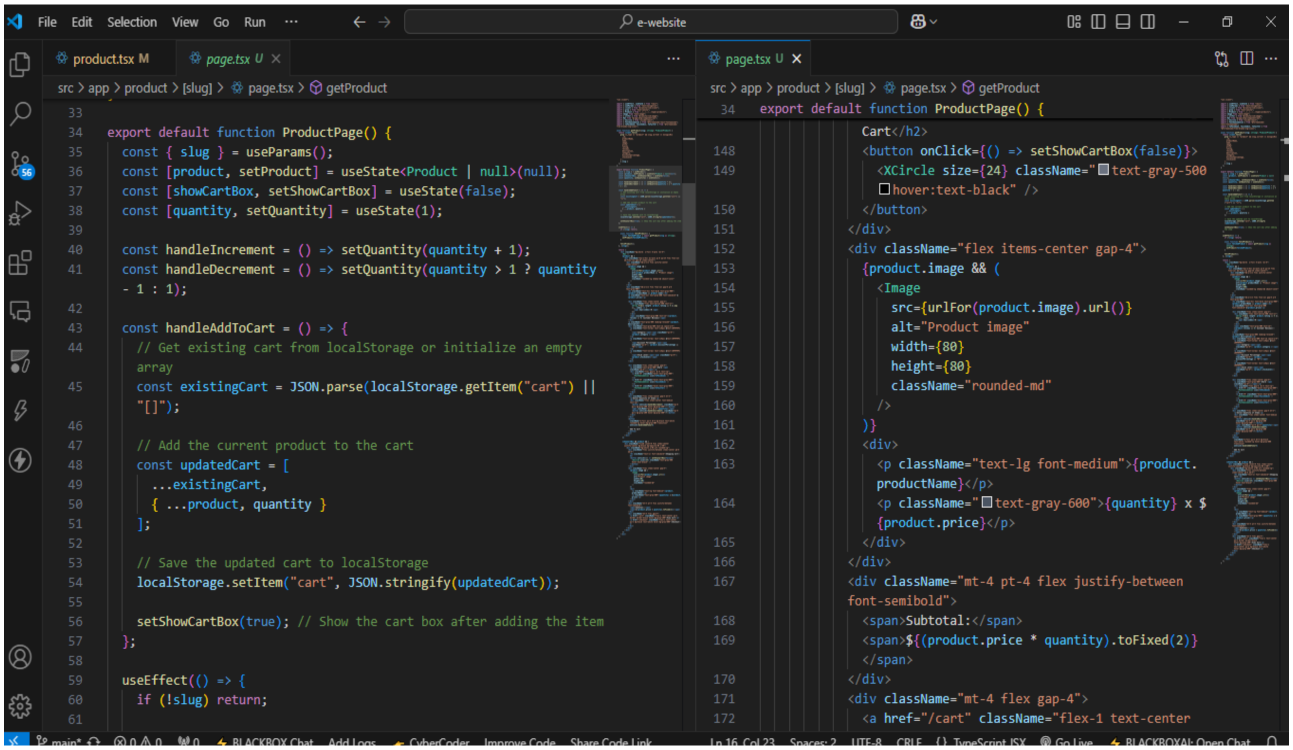
Fetched and displayed detailed product information.



```tsx
"use client";

import { useEffect, useState } from "react";
import { useParams } from "next/navigation";
import { client } from "@/sanity/lib/client";
import { groq } from "next-sanity";
import { Product } from "../../../../types/products";
import Image from "next/image";
import { urlFor } from "@/sanity/lib/image";
import Nav from "@/app/shop/component/nav";
import { XCircle } from "lucide-react";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faFacebook, faLinkedin, faTwitter } from "@fortawesome/free-brands-svg-icons";

async function getProduct(slug: string): Promise<Product> {
  return client.fetch(
    groq`*[_type == "product" && slug.current == $slug][0]{
      _id,
      productName,
      name,
      image,
      price,
      rating,
      reviews,
      description,
      category,
      discountpercentage,
      stockLevel
    }`,
    { slug }
  );
}
```

Added related products section to improve user engagement.

Ensured mobile responsiveness for better usability.

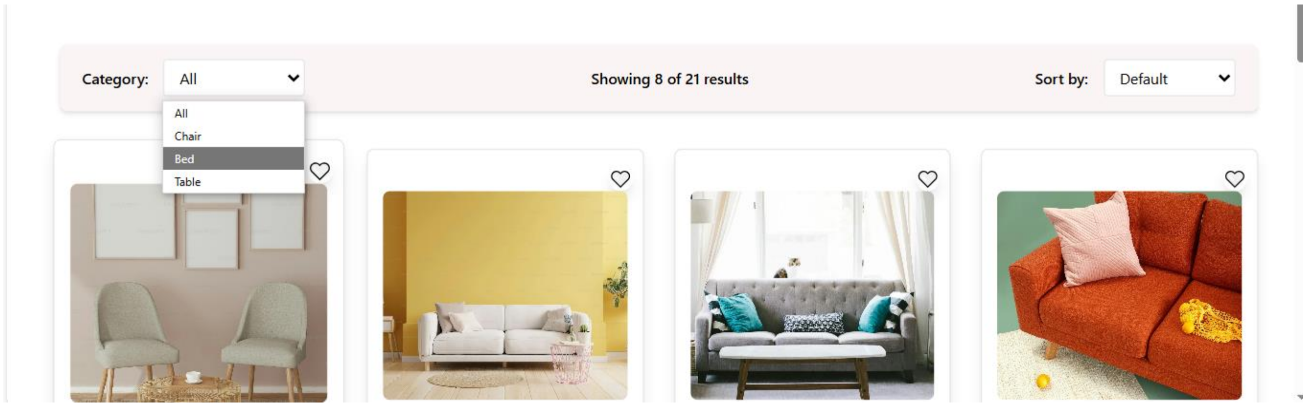


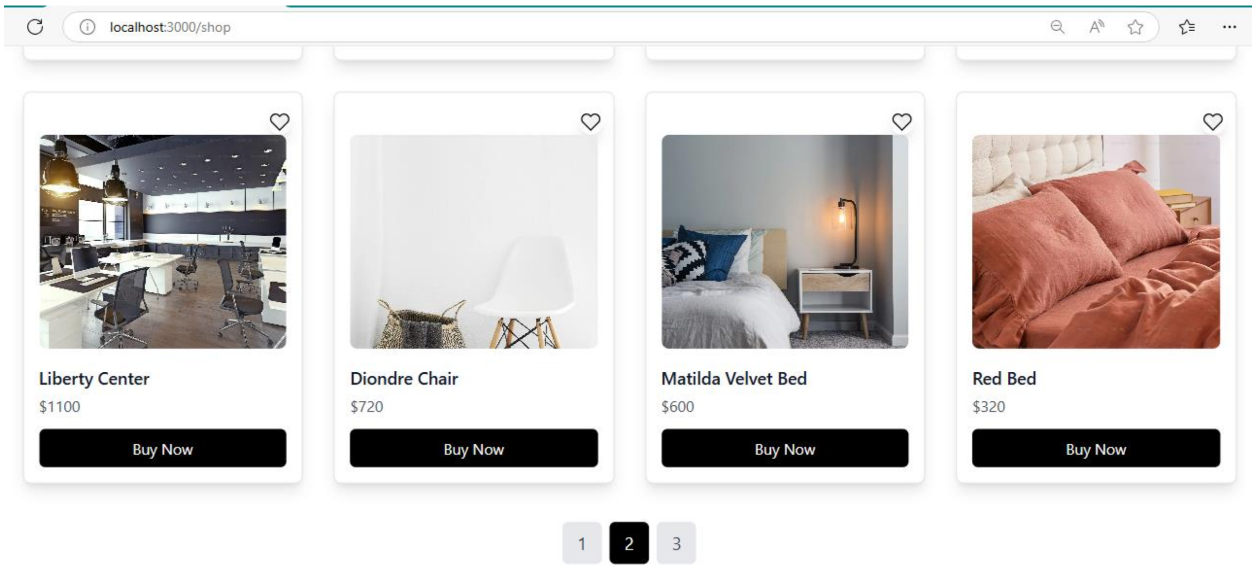
```tsx
export default function ProductPage() {
  const { slug } = useParams();
  const [product, setProduct] = useState<Product | null>(null);
  const [showCartBox, setShowCartBox] = useState(false);
  const [quantity, setQuantity] = useState(1);

  const handleIncrement = () => setQuantity(quantity + 1);
  const handleDecrement = () => setQuantity(quantity > 1 ? quantity - 1 : 1);

  const handleAddToCart = () => {
    // Get existing cart from localStorage or initialize an empty array
    const existingCart = JSON.parse(localStorage.getItem("cart") || "[]");

    // Add the current product to the cart
    const updatedCart = [
      ...existingCart,
      { ...product, quantity }
    ];

    // Save the updated cart to localStorage
    localStorage.setItem("cart", JSON.stringify(updatedCart));

    setShowCartBox(true); // Show the cart box after adding the item
  };

  useEffect(() => {
    if (!slug) return;
```

```tsx
            Cart</h2>
            <button onClick={() => setShowCartBox(false)}>
              <XCircle size={24} className="text-gray-500 hover:text-black" />
            </button>
          </div>
          <div className="flex items-center gap-4">
            {product.image && (
              <Image
                src={urlFor(product.image).url()}
                alt="Product image"
                width={80}
                height={80}
                className="rounded-md"
              />
            )}
            <div>
              <p className="text-lg font-medium">{product.productName}</p>
              <p className="text-gray-600">{quantity} x ${product.price}</p>
            </div>
          </div>
          <div className="mt-4 pt-4 flex justify-between font-semibold">
            <span>Subtotal:</span>
            <span>${(product.price * quantity).toFixed(2)}</span>
          </div>
          <div className="mt-4 flex gap-4">
            <a href="/cart" className="flex-1 text-center
```

3. Category Filters, Search Bar, and Pagination

To enhance user experience, category-based filtering, a search bar, and pagination were implemented to streamline product navigation.





Implementation Steps:

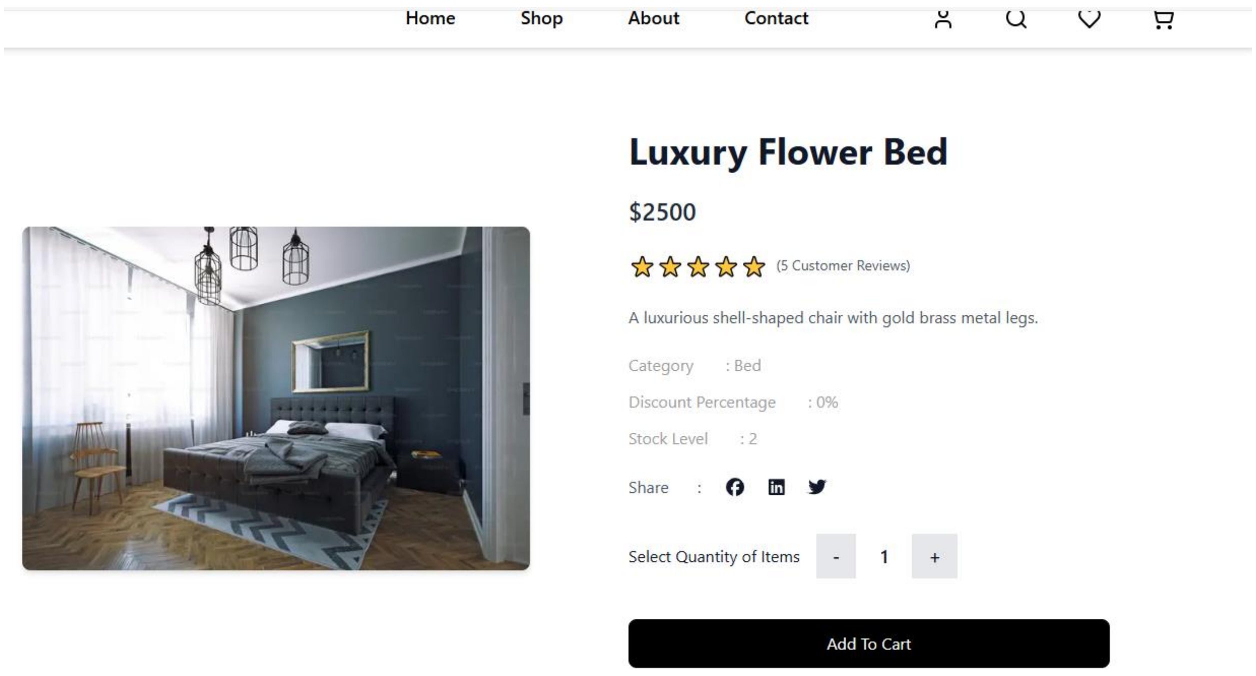
Implemented category filters to refine search results based on product types.

Designed and developed a search bar with real-time query matching.

Integrated pagination for better content organization and performance optimization.

4. Additional Features

In addition to the core functionalities, extra features were implemented to enhance the marketplace experience.



Luxury Flower Bed

$2500

(5 Customer Reviews)

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

Category : Bed

Discount Percentage : 0%

Stock Level : 2

Share :

Select Quantity of Items - 1 +

Key Enhancements:

User profile component for personalized experiences.

Related products section to encourage further browsing.

Enhanced UI/UX design for an intuitive shopping experience.

Challenges and Solutions

Challenges Faced & Solutions Implemented:

Rendering Dynamic Data Efficiently:

Challenge: Ensuring smooth and error-free data fetching while handling API responses.

Solution: Used efficient state management with React hooks and implemented proper error handling mechanisms.

Enhancing Search Functionality:

Challenge: Making search results accurate and responsive with minimal delay.

Solution: Utilized debounce techniques to reduce excessive API calls and improve search efficiency.

Ensuring Smooth Navigation Between Pages:

Challenge: Maintaining a seamless user experience while navigating through dynamic routes.

Solution: Implemented Next.js dynamic routing with optimized prefetching to enhance navigation speed.

Conclusion

This stage of development successfully implemented dynamic frontend components, ensuring a seamless and interactive marketplace experience. The integration of filters, search functionality, and pagination further enhanced usability, improving overall user engagement.

Prepared By: Neha Fahim

Hackathon Submission - Day 4