Day 4 - Dynamic Frontend Components –Hecto

Hackathon Assignment
Date: January 19, 2025


Tasks Completed Today:

1. Product Listing Page with Dynamic Data

```
1   import { client } from "@/sanity/lib/client";
2   import { NextResponse } from "next/server";
3
4   export async function GET(request: Request) {
5     const { searchParams } = new URL(request.url);
6     const page = Number(searchParams.get("page")) || 1;
7     const limit = Number(searchParams.get("limit")) || 10;
8
9     const start = (page - 1) * limit;
10    const end = start + limit;
11
12    try {
13      // Fetch paginated products from Sanity
14      const products = await client.fetch(
15        `*[_type == "product"] | order(_createdAt desc) [${start}...${end}] {
16          _id,
17          name,
18          "image": image.asset->url,
19          price
20        }`
21      );
22
23      // Fetch total count of products
24      const totalCount = await client.fetch(
25        `count(*[_type == "product"])`
26      );
27
28      const totalPages = Math.ceil(totalCount / limit);
29
30      return NextResponse.json({
31        products,
32        totalPages,
33        currentPage: page,
34      });
35    } catch (error) {
36      console.error("Error fetching products:", error);
37      return NextResponse.json(
38        { message: "Error fetching products" },
39        { status: 500 }
40      );
41    }
42  }
43
44
```

- **Objective**: Create a product listing page that dynamically fetches data.
- **Implementation**:
  - Connected to the API to fetch and display product data dynamically.
  - Designed a responsive grid layout using Tailwind CSS.
  - Included essential fields such as product name, price, and image.
- **Reference**: Aligned with the "Product Listing Component" section from the document

2. Search Bar

- **Objective**: Implement a search bar to filter products by name or tags.
- **Implementation**:
  - Added a search input field with debounce functionality to optimize filtering.
  - Dynamically filtered the product list based on user input.
  - Included a clear button to reset the search term.
- **Reference**: Built according to the "Search Bar" section in the document.
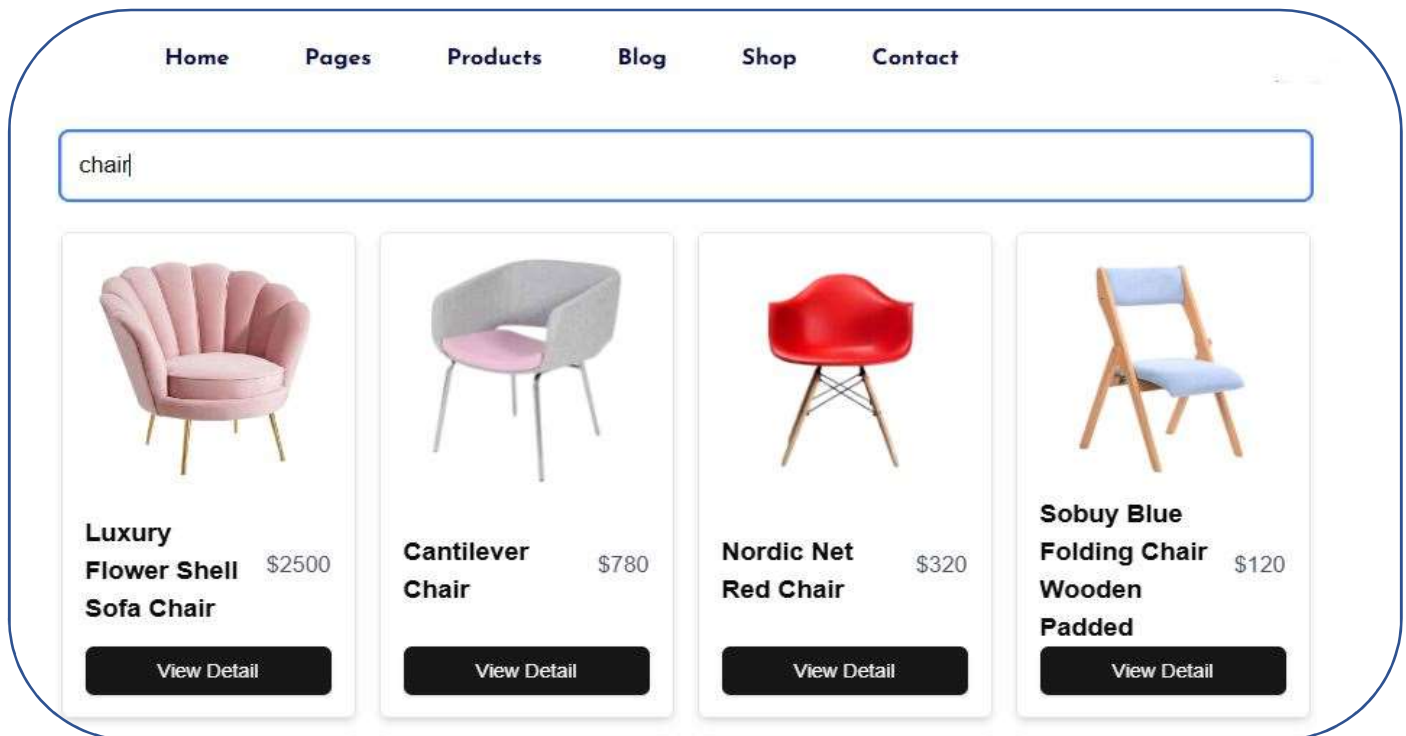
3. Pagination

- **Objective**: Divide the product list into pages to enhance performance and user experience.

- **Implementation**:
    - Set up a pagination component to navigate through pages.
    - Integrated pagination with the API to fetch products based on the current page.
    - Displayed navigation buttons for previous and next pages with appropriate styles.
- **Reference**: Followed the guidelines from the "Pagination Component" section in the document.
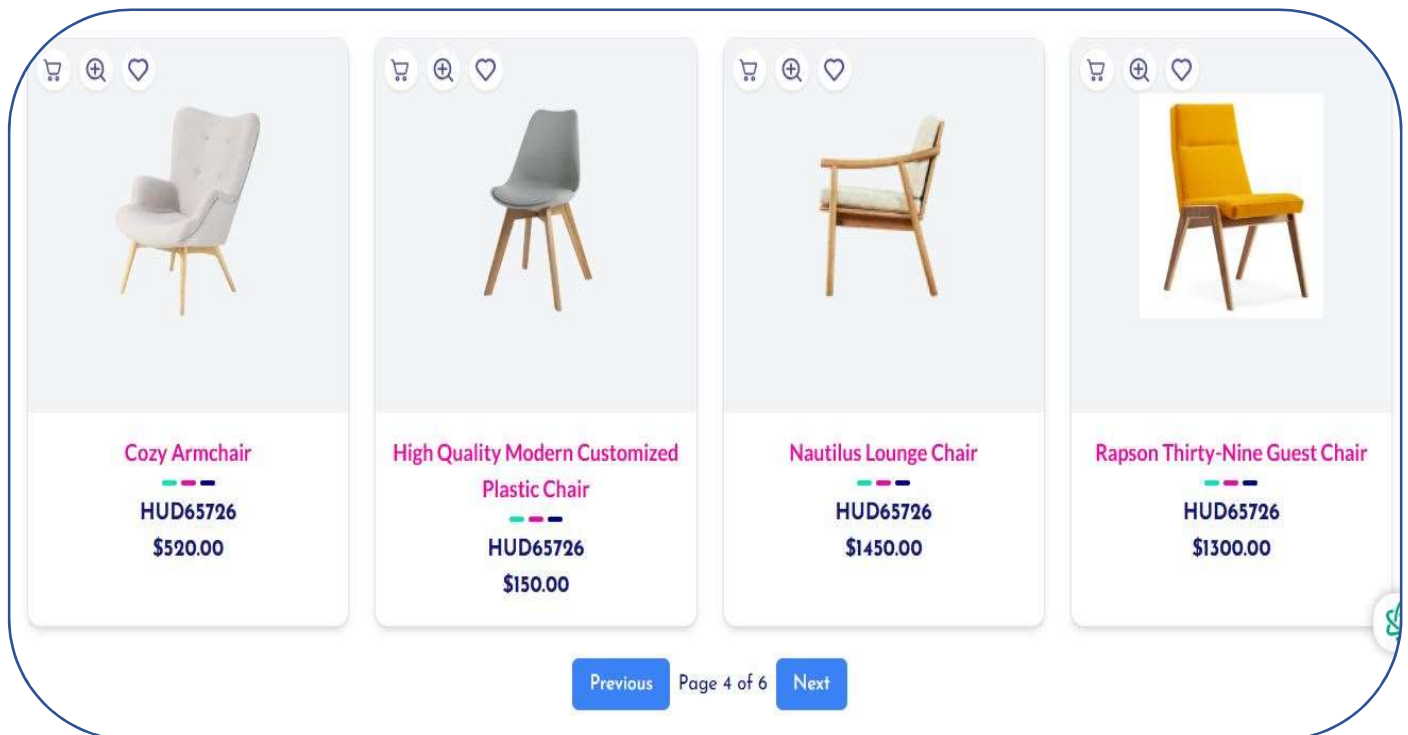
4. Product Detail Page

- **Objective**: Create a detailed product page using dynamic routing.
- **Implementation**:
    - Added dynamic routes to display individual product details.
    - Displayed additional fields such as product description, price, and stock status.
    - Ensured mobile-friendly design and smooth navigation.
- **Reference**: Developed as per the "Product Detail Component" section in the document.

```tsx
1   "use client";
2
3   import Tabs from "@/components/Tabs";
4   import RelatedProducts from "@/components/RelatedProducts";
5   import Link from "next/link";
6   import Image from "next/image";
7   import { useCart } from "@/context/CartContext";
8   import { client } from "@/sanity/lib/client";
9   import { urlFor } from "@/sanity/lib/image";
10  import { Product } from "../../../../data/products";
11  import { useState, useEffect } from "react";
12  const fetchProductsbyId = async (
13    _id: string
14  ): Promise<Product | null> => {
15    const query = `*[_type == "product" && _id == $_id][0]{
16      _id,
17      name,
18      price,
19      discountPercentage,
20      code,
21      image,
22      rating,
23      category,
24      isSale,
25      description,
26      colors,
27      stockLevel,
28      size,
29    }`;
30    const allProducts = await client.fetch(query, { _id });
```

```tsx
35  type ProductDetailsProps = {
36    params: {
37      _id: string;
38    };
39  };
40  export default function ProductDetailsPage({ params }: ProductDetailsProps) {
41    const { addToCart } = useCart();
42    const { _id } = params;
43    const [product, setProduct] = useState<Product | null>(null);
44
45    useEffect(() => {
46      const fetchProduct = async () => {
47        try {
48          const data = await fetchProductsbyId(_id);
49          setProduct(data);
50        } catch (error) {
51          console.error("Error fetching product:", error);
52        }
53      };
54      fetchProduct();
55    }, [_id]);
56
57    if (!product) {
58      return (
59        <div className="text-center py-20 text-2xl">Product loading ...</div>
60      );
61    }
```

roduct Details

Home / Pages Product Details

Luxury Flower Shell Sofa Chair

★★★★☆ (12)

$2500.00 $0.00

Color

High-quality plywood chair with ergonomic design. Corporis archi voluptate eius cupiditate vitae, soluta.

Add to Cart ♡

Tags

Share 🇫 📷 🐦

5. Wishlist Functionality

- **Objective**: Allow users to save products for future reference.
- **Implementation**:
  - Built a wishlist feature using local storage to persist data.
  - Added a "Save to Wishlist" button to product cards and detail pages.
  - Styled buttons dynamically to indicate if a product is already in the wishlist.
- **Reference**: Used the "Wishlist Component" section for guidance.

Product	Price	Quantity	Total
Luxury Flower Shell Sofa Chair	$2500.00	− 1 +	$2500.00
Sobuy Blue Folding Chair Wooden Padded	$120.00	− 1 +	$120.00
Matilda Velvet Chair – Pink	$600.00	− 1 +	$600.00
Armchair Tortuga	$850.00	− 1 +	$850.00

Update Cart

Clear Cart

6. Cart with Functionalities

- **Objective**: Enable users to add, remove, and view products in the cart.
- **Implementation**:
    - Integrated a cart context to manage the cart state globally.
    - Added "Add to Cart" buttons with dynamic styling (e.g., color change on adding to cart).
    - Included cart functionalities such as increasing/decreasing quantities and calculating the total price.
- **Reference**: Followed the "Cart Component" section in the document.

Summary of Professional Practices:

1. **Dynamic Data Handling**:
    - Fetched data dynamically from APIs for a real-world, scalable setup.
2. **Responsive Design**:
    - Used Tailwind CSS to ensure all components are mobile and desktop friendly.
3. **Reusable Components**:
    - Created modular components like `ProductCard`, `Pagination`, and `SearchBar` for easy reuse and maintenance.
4. **State Management**:
    - Utilized React Context for managing cart and wishlist functionality.
5. **User Experience Enhancements**:
    - Added visual feedback (e.g., dynamic button styling) to improve interactivity.

Next Steps:

1. Implement professional multi-language support across the website.
2. Enhance the search functionality with advanced filters and suggestions.
3. Optimize API calls for better performance.
4. Add user authentication to integrate personalized features like wishlists and carts.

Submission Requirements:

- Screenshots or screen recordings of:
    - Product listing page with dynamic data.
    - Functional search bar and pagination.
    - Product detail page with accurate routing.
    - Working wishlist and cart functionalities.
- Code snippets for key components (e.g., `ProductCard`, `Pagination`, `SearchBar`).

- Video link: Linkedin/Day-4-Dynamic-Frontend-Components

Self-Validation Checklist

Tasks	✔	✗
Frontend Component Development	✔	
Styling and Responsiveness	✔	
Code Quality	✔	
Documentation and Submission	✔	
Final Review	✔	

Prepared by: Ghulam Akber

Roll No. : 00231794
Hackathon Project: Dynamic Marketplace Hecto