Market Blace Builder Hackathon 2025

Day 4 Report

Building Dynamic Frontend Components

Project Name: "Avion(FurnitureShop): A General-Ecommerce Marketplace"

Date: 19/01/2025

1. Objective

The goal of Day 4 was to create and integrate dynamic frontend components for the website, ensuring features are reusable, responsive, and user-friendly. The primary focus was on the **Products Page** and **Cart Page**.

2. Components Built

a) Navbar

Description:

A responsive navigation bar for easy access to the Home, Products, and Cart pages.

Features:

- Links to Home (/), Products (`/products`), and Cart (`/cart`).
- Highlighted styling for hover effects.

b) Products Page

Description:

A dynamic listing of furniture products fetched from the Sanity CMS.

Features:

- **Search Bar**: Allows users to search for products by name dynamically.
- **Category Filter**: Filters products based on selected categories (e.g., Sofa, Table, Chair).

- **Pagination**: Displays 6 items per page, with "Previous" and "Next" buttons.

c) ProductCard Component

Description:

A reusable component to display product data.

Features:

- Displays name, category, price, and images.
- Includes a clickable image for navigating to detailed pages.

d) Product Detail Page

Description:

Shows detailed information about a selected furniture product.

Features:

- Displays name, category, description, price, and dimensions.
- Quantity selector and "Add to Cart" button.

e) Cart Page

Description:

A dedicated page to display the items added to the cart, along with the total price and options to remove items or clear the cart.

Features:

- Displays a list of products added to the cart.
- Shows the total price of all items in the cart.
- Buttons for removing individual items and clearing the entire cart.

3. Code Snippets of Key Components

a) Navbar Component

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

export default function Navbar() {
  return (
    <nav className="bg-gray-800 text-white p-4">
      <div className="container mx-auto flex justify-between items-center">
        <h1 className="text-2xl font-bold">Furniture Shop</h1>
        <div className="space-x-4">
          <Link href="/">Home</Link>
          <Link href="/products">Products</Link>
          <Link href="/cart">Cart</Link>
        </div>
      </div>
    </nav>
  );
}
```

b) ProductCard Component

```
import { client, imageUrlBuilder } from '@/sanity/lib/client';
import Image from 'next/image';
import Link from 'next/link';

type ProductCardProps = {
  product: {
    _id: string;
    name: string;
    slug: { current: string };
    image: { asset: { _ref: string } } | null;
    price: number;
    quantity: number;
    tags: string[];
    description: string;
    features: string[];
    dimensions: {
      height: string;
      width: string;
      depth: string;
    };
  };
};

export default function ProductCard({ product }: ProductCardProps) {
  // Check if the product has an image and generate the URL correctly
  const imageUrl = product.image && product.image.asset
    ? imageUrlBuilder(client).image(product.image.asset._ref).width(300).height(300).url()
    : '/placeholder.jpg'; // Use fallback if no image

  return (
    <div className="bg-white shadow-lg rounded-lg p-4">
      <Link href={`/products/${product.slug.current}`}>
        <Image
          src={imageUrl}  // Use the generated image URL or fallback
          alt={product.name}
          width={300}
```

```
          height={300}
          className="rounded-t-lg object-cover w-full h-64"
        />
        <h2 className="text-lg font-bold mt-2">{product.name}</h2>
        <p className="text-green-600 font-semibold">${product.price}</p>
      </Link>
    </div>
  );
}
```

c) SearchBar Component

```
export default function SearchBar({ onSearch }: { onSearch: (query: string) => void }) {
    return (
      <input
        type="text"
        placeholder="Search products..."
        className="p-2 border border-gray-300 rounded w-full"
        onChange={(e) => onSearch(e.target.value)}
      />
    );
  }
```

d) Pagination Logic

```
const startIndex = (currentPage - 1) * itemsPerPage;
const paginatedProducts = filteredProducts.slice(startIndex, startIndex + itemsPerPage);

const totalPages = Math.ceil(filteredProducts.length / itemsPerPage);
```

e) Product Page

```
import { client } from '@/sanity/lib/client';
import Navbar from '../components/Navbar';
import ProductCard from '../components/ProductCard';

export default async function ProductsPage() {
  const query = `*[_type == "product"]{ name, price, image, slug }`;
  const products = await client.fetch(query);

  return (
    <>
      <Navbar />
      <div className="container mx-auto py-8">
        <h1 className="text-3xl font-bold mb-6">Our Products</h1>
        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-8">
          {products.map((product: any) => (
            <ProductCard key={product.slug.current} product={product} />
          ))}
        </div>
      </div>
    </>
  );
}
```

f) Product Detail Page

```
'use client';

import Image from 'next/image';
import { urlFor } from '@/sanity/lib/client';
import Navbar from '@/app/components/Navbar';
import { useCart } from '@/app/components/context/CartContext';

type Product = {
  _id: string;
  name: string;
  slug: { current: string };
  image: { asset: { _ref: string } }; // Ensure this matches your schema
  price: number;
  quantity: number;
  tags: string[];
  description: string;
  features: string[];
  dimensions: {
    height: string;
    width: string;
    depth: string;
  };
};

export default function ProductDetails({ product }: { product: Product }) {
    const { addToCart } = useCart();

  const handleAddToCart = () => {
    addToCart({
      id: product._id,
      name: product.name,
      price: product.price,
      quantity: 1, // Default to 1 when adding for the first time
      image: product.image.asset._ref, // Pass the image reference
    });
  };

  return (
    <>
    <Navbar />
    <div className="grid grid-cols-1 md:grid-cols-2 gap-8 my-8">
      {/* Product Image */}
      {product.image?.asset?._ref ? (
        <Image
          src={urlFor(product.image.asset._ref)} // Use urlFor to get the correct image URL
          alt={product.name}
          width={300}
          height={300}
          className="rounded-lg object-cover"
        />
      ) : (
        <div className="bg-gray-200 w-full h-64 rounded-lg flex items-center justify-center">
          <span>No Image Available</span>
        </div>
      )}

      {/* Product Details */}
      <div>
        <h1 className="text-4xl font-bold">{product.name}</h1>
```

```
        <p className="text-lg mt-4">{product.description}</p>
        <p className="text-2xl text-green-600 font-bold mt-6">${product.price}</p>
        <button
          className="bg-blue-500 text-white px-4 py-2 rounded-lg mt-4"
          onClick={handleAddToCart}
        >
          Add to Cart
        </button>
      </div>
    </div>
    </>
  );
}
```

g) Cart Page

```
import { useCart } from "../components/context/CartContext";


export default function CartPage() {
  const { cart, removeFromCart } = useCart();

  return (
    <div className="container mx-auto py-8">
      <h1 className="text-3xl font-bold mb-6">Your Cart</h1>
      {cart.length === 0 ? (
        <p>Your cart is empty.</p>
      ) : (
        cart.map((item) => (
          <div key={item.id} className="flex justify-between items-center mb-4">
            <div>
              <p className="font-bold">{item.name}</p>
              <p>${item.price}</p>
            </div>
            <button
              className="bg-red-500 text-white px-4 py-2 rounded-lg"
              onClick={() => removeFromCart(item.id)}
            >
              Remove
            </button>
          </div>
        ))
      )}
    </div>
  );}
```

h) Context/CartContext.tsx

```
"use client"; // This makes the file a client component

import { createContext, useContext, useState, ReactNode } from "react";

type CartItem = {
  id: string;
  name: string;
  price: number;
  quantity: number;
  image: string;
};
```

```tsx
type CartContextType = {
  cart: CartItem[];
  addToCart: (item: CartItem) => void;
  removeFromCart: (id: string) => void;
  updateQuantity: (id: string, quantity: number) => void;
};

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

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

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


  const removeFromCart = (id: string) => {
    setCart((prev) => prev.filter((item) => item.id !== id));
  };

  const updateQuantity = (id: string, quantity: number) => {
    setCart((prev) =>
      prev.map((item) => (item.id === id ? { ...item, quantity } : item))
    );
  };

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

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

4. Challenges Faced

Search Bar:

Initially, the search bar was not functional. This was resolved by passing a callback function to filter and render products dynamically.

Pagination:

Adjusting the logic to ensure seamless navigation across pages.

Dynamic Routing:

Implemented and tested detail pages for individual products.

Cart Management:

Ensuring the cart updates dynamically and calculates the total price correctly.

5. Conclusion

All tasks for Day 4 were completed successfully:

- Dynamic components were created and integrated into the website.
- Features like search, filter, pagination, dynamic routing, and cart functionality were implemented and tested.
- Product data was displayed effectively with a professional and responsive UI.

Prepared by: Kinza M. Ayub