Day 4 -BUILDING DYNAMIC FRONTEND COMPONENTS FOR YOUR MARKETPLACE
Sit & Style Studio
Date: 30-1-25
Name: Syeda Hafiza Bibi Amna

Project Overview

Sit & Style Studio is a feature-rich online store that provides:

- Device-responsive design
- Authorisation and authentication of users
- Searching for and filtering products
- Wishlist and shopping cart features
- Safe checkout procedure
- Product and order management admin dashboard
- Subscription to a newsletter
- Updates about products in real time

Features

- Responsive Design: Adapts seamlessly to all screen sizes, providing a consistent experience.
- Dynamic Components: Built with React and Next.js for modular and scalable development.
- Product Display: Includes an image gallery and detailed descriptions for showcasing products.
- Call-to-Action Buttons: Encourages user interaction with intuitive and styled buttons.

Tech Stack

- Frontend: Next.js 14, TypeScript, Tailwind CSS
- Backend: Sanity.io (Headless CMS)
- Authentication: NextAuth.js
- State Management: React Context
- Styling: Tailwind CSS with custom configuration
- Animation: Framer Motion

Procedures Undertaken for Component Development and Integration:

1. Initialisation and Data Acquisition:

● Via the Sanity client, a connection was made between the frontend and Sanity CMS, guaranteeing effective and safe communication.

● Used API endpoints to verify the accessibility and structural soundness of all data models, including `Products` and `Categories`.

● Designed scalable and reusable data-fetching features for key elements including `SearchBar`, `CategoryFilter`, and `ProductList`.

2. Creation of Fundamental Elements:

Product Listing Component:

● Product info that is dynamically displayed in a responsive grid layout.

● Made use of card-based interfaces to show important details such product name, cost, and stock level.

Component of Product Details:

● Created distinct pages for each product entry using Next.js's dynamic routing feature.

● Integrated comprehensive product details, such as high-resolution photos, prices, and descriptions.

Category Search & Filter Component:

- The category filter component allowed for real-time product filtering based on user-selected categories and dynamically retrieved category data from APIs to aid in product categorisation.
- sophisticated search features that enable product filtering by names and related tags.

Pagination Component:

- Integrated user-friendly navigation features like "previous" and "next" buttons to effectively manage large product catalogues.

3. Styling and Adaptive Design:

- To create a cohesive, visually appealing, and mobile-responsive user experience, Tailwind CSS was used.
- Made sure component layouts could adjust to different screen sizes using dynamic styling techniques.

Challenges Recognised and Associated Solutions:

1. Problem: Response Delays and API Latency

Problem:

- Extended reaction times during data retrieval reduced the effectiveness of component rendering.
- Because of incorrectly configured origin settings in Sanity, encountered CORS-related problems when retrieving data.

Solutions:

- To give visual feedback while retrieving data, a loading state and skeleton user interface were added.
- Sanity CMS's CORS settings were modified to whitelist the frontend's origin, allowing continuous flow of data.

2. Challenge: Errors in Dynamic Routing

Problem:

- When rendering the product information page, invalid or missing product IDs caused errors.

Solution:

- Designed fallback pages and implemented strong error handling procedures to shamefully handle invalid or missing product data.

3. Challenge: Complex Filtering and Pagination Integration

Problem:

- Maintaining state consistency was difficult when coordinating several filters (such as category and price range) with pagination.

Solution:

- To synchronise filtering and pagination states across browser reloads, URL-based query parameters were implemented.

Adopted Best Practices:

Component Reusability:

● To encourage scalability and maintainability, modular and reusable components were developed, such as `ProductCard` and `CategoryFilter`.

Secure Configuration Management:

- Sensitive API keys were stored using `.env.local`, improving overall security and conformity to industry standards.

Error Mitigation:

- Used thorough error-handling techniques to control API malfunctions and guarantee a flawless user experience.

Improved Code Standards:

- incorporated thorough code comments and used descriptive naming conventions to aid in readability and future development.

Screenshots:

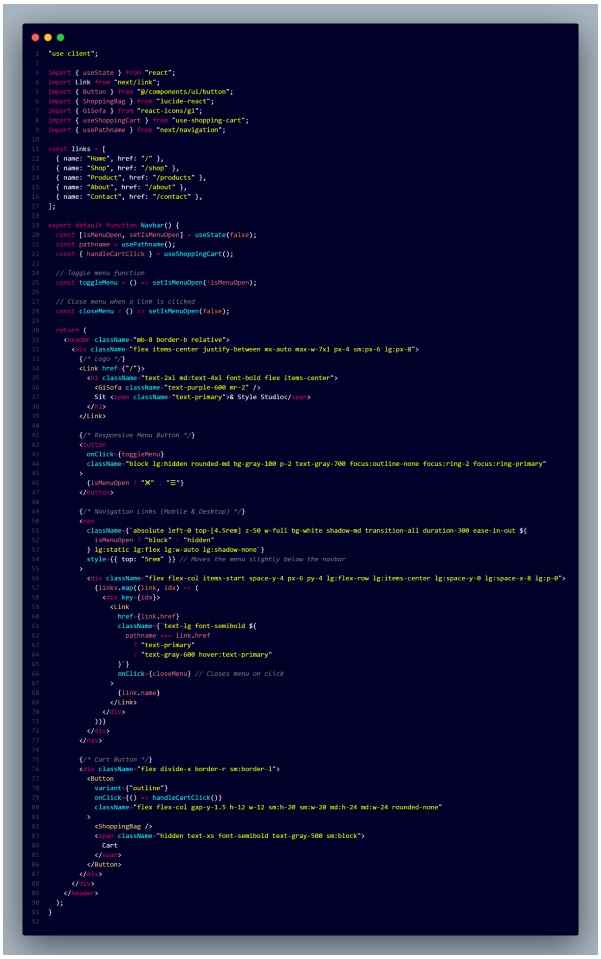
ProductDetails:

```tsx
import AddToBag from "@/app/components/AddToBag";
import CheckoutNow from "@/app/components/CheckoutNow";
import { Button } from "@/components/ui/button";
import { fullProduct } from "@/sanity/interface";
import { client } from "@/sanity/lib/client";
import { Star, Truck } from "lucide-react";

async function getData() {
  const query = `*[_type == "product"][5] {
  _id,
  title,
  price,
  "priceWithoutDiscount": priceWithoutDiscount,
  badge,
  "imageUrl": image.asset->url, // Fetch all image URLs in the array
  "categoryName": category->title, // Resolve category name
  description,
  inventory,
  tags,
}
`;

  const data = await client.fetch(query);
  console.log(data);
  return data;
}

export const dynamic = "force-dynamic";

export default async function ProductPge() {
  const data: fullProduct = await getData();

  return (
    <div className="bg-white reveal-on-scroll">
      <div className="mx-auto max-w-screen-xl px-4 md:px-8">
        <div className="grid gap-8 md:grid-cols-2 reveal-on-scroll">
          <img
                  src={data.imageUrl}
                  alt={data.title}
                  height={400}
                  width={400}
                  className="h-full w-full object-cover transition-transform duration-300 group-hover:scale-105"
                />

            <div className="md:py-8">
              <div className="mb-2 md:mb-3">
                <span className="mb-0.5 inline-block text-gray-500">
                  {data.categoryName}
                </span>
                <h2 className="text-2xl font-bold text-gray-800 lg:text-3xl">
                  {data.title}
                </h2>
              </div>

              <div className="mb-6 flex items-center gap-3 md:mb-10">
                <Button className="rounded-full gap-x-2">
                  <span className="text-sm">4.2</span>
                  <Star className="h-5 w-5" />
                </Button>

                <span className="text-sm text-gray-500 transition duration-100">
                  56 Ratings
                </span>
              </div>

              <div className="mb-4">
                <div className="flex items-end gap-2">
                  <span className="text-xl font-bold text-gray-800 md:text-2xl">
                    ${data.price}
                  </span>
                  <span className="mb-0.5 text-red-500 line-through">
                    ${data.tags}
                  </span>
                </div>

                <span className="text-sm text-gray-500">
                  Incl. Vat plus shipping
                </span>
              </div>

              <div className="mb-6 flex items-center gap-2 text-gray-500">
                <Truck classNam="w-6 h-6" />
                <span className="text-sm">2-4 Day Shipping</span>
              </div>

              <div className="flex gap-2.5">
                <AddToBag
                  currency="USD"
                  description={data.description}
                  imgeUrl={data.imageUrl}
                  name={data.title}
                  price={data.price}
                  key={`${data._id}`}
                  price_id={data.price_id}
                />
                <CheckoutNow
                  currency="USD"
                  description={data.description}
                  imgeUrl={data.imageUrl}
                  name={data.title}
                  price={data.price}
                  key={data._id}
                  price_id={data.price_id}
                />
              </div>

              <p className="mt-12 text-base text-gray-500 tracking-wide">
                {data.description}
              </p>
            </div>
          </div>
        </div>
      </div>
    );
}
```

Migrate.mjs

```
// Import environment variables from .env.local
import "dotenv/config";

// Import the Sanity client to interact with the Sanity backend
import { createClient } from "@sanity/client";

// Load required environment variables
const {
  NEXT_PUBLIC_SANITY_PROJECT_ID, // Sanity project ID
  NEXT_PUBLIC_SANITY_DATASET, // Sanity dataset (e.g., "production")
  NEXT_PUBLIC_SANITY_AUTH_TOKEN, // Sanity API token
  BASE_URL = "https://giaic-hackathon-template-08.vercel.app", // API base URL for products and categories
} = process.env;

// Check if the required environment variables are provided
if (!NEXT_PUBLIC_SANITY_PROJECT_ID || !NEXT_PUBLIC_SANITY_AUTH_TOKEN) {
  console.error("Missing required environment variables. Please check your .env.local file.");
  process.exit(1); // Stop execution if variables are missing
}

// Create a Sanity client instance to interact with the target Sanity dataset
const targetClient = createClient({
  projectId: NEXT_PUBLIC_SANITY_PROJECT_ID, // Your Sanity project ID
  dataset: NEXT_PUBLIC_SANITY_DATASET || "production", // Default to "production" if not set
  useCdn: false, // Disable CDN for real-time updates
  apiVersion: "2023-01-01", // Sanity API version
  token: NEXT_PUBLIC_SANITY_AUTH_TOKEN, // API token for authentication
});

// Function to upload an image to Sanity
async function uploadImageToSanity(imageUrl) {
  try {
    // Fetch the image from the provided URL
    const response = await fetch(imageUrl);
    if (!response.ok) throw new Error(`Failed to fetch image: ${imageUrl}`);

    // Convert the image to a buffer (binary format)
    const buffer = await response.arrayBuffer();

    // Upload the image to Sanity and get its asset ID
    const uploadedAsset = await targetClient.assets.upload("image", Buffer.from(buffer), {
      filename: imageUrl.split("/").pop(), // Use the file name from the URL
    });

    return uploadedAsset._id; // Return the asset ID
  } catch (error) {
    console.error("Error uploading image:", error.message);
    return null; // Return null if the upload fails
  }
}

// Main function to migrate data from REST API to Sanity
async function migrateData() {
  console.log("Starting data migration...");

  try {
    // Fetch categories from the REST API
    const categoriesResponse = await fetch(`${BASE_URL}/api/categories`);
    if (!categoriesResponse.ok) throw new Error("Failed to fetch categories.");
    const categoriesData = await categoriesResponse.json(); // Parse response to JSON

    // Fetch products from the REST API
    const productsResponse = await fetch(`${BASE_URL}/api/products`);
    if (!productsResponse.ok) throw new Error("Failed to fetch products.");
    const productsData = await productsResponse.json(); // Parse response to JSON

    const categoryIdMap = {}; // Map to store migrated category IDs

    // Migrate categories
    for (const category of categoriesData) {
      console.log(`Migrating category: ${category.title}`);
      const imageId = await uploadImageToSanity(category.imageUrl); // Upload category image

      // Prepare the new category object
      const newCategory = {
        _id: category._id, // Use the same ID for reference mapping
        _type: "categories",
        title: category.title,
        image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
      };

      // Save the category to Sanity
      const result = await targetClient.createOrReplace(newCategory);
      categoryIdMap[category._id] = result._id; // Store the new category ID
      console.log(`Migrated category: ${category.title} (ID: ${result._id})`);
    }

    // Migrate products
    for (const product of productsData) {
      console.log(`Migrating product: ${product.title}`);
      const imageId = await uploadImageToSanity(product.imageUrl); // Upload product image

      // Prepare the new product object
      const newProduct = {
        _type: "product",
        title: product.title,
        price: product.price,
        priceWithoutDiscount: product.priceWithoutDiscount,
        badge: product.badge,
        image: imageId ? { _type: "image", asset: { _ref: imageId } } : undefined, // Add image if uploaded
        category: {
          _type: "reference",
          _ref: categoryIdMap[product.category._id], // Use the migrated category ID
        },
        description: product.description,
        inventory: product.inventory,
        tags: product.tags,
      };

      // Save the product to Sanity
      const result = await targetClient.create(newProduct);
      console.log(`Migrated product: ${product.title} (ID: ${result._id})`);
    }

    console.log("Data migration completed successfully!");
  } catch (error) {
    console.error("Error during migration:", error.message);
    process.exit(1); // Stop execution if an error occurs
  }
}

// Start the migration process
migrateData();
```

NavBar.tsx



```tsx
"use client";

import { useState } from "react";
import Link from "next/link";
import { Button } from "@/components/ui/button";
import { ShoppingBag } from "lucide-react";
import { GiSofa } from "react-icons/gi";
import { useShoppingCart } from "use-shopping-cart";
import { usePathname } from "next/navigation";

const links = [
  { name: "Home", href: "/" },
  { name: "Shop", href: "/shop" },
  { name: "Product", href: "/products" },
  { name: "About", href: "/about" },
  { name: "Contact", href: "/contact" },
];

export default function Navbar() {
  const [isMenuOpen, setIsMenuOpen] = useState(false);
  const pathname = usePathname();
  const { handleCartClick } = useShoppingCart();

  // Toggle menu function
  const toggleMenu = () => setIsMenuOpen(!isMenuOpen);

  // Close menu when a link is clicked
  const closeMenu = () => setIsMenuOpen(false);

  return (
    <header className="mb-8 border-b relative">
      <div className="flex items-center justify-between mx-auto max-w-7xl px-4 sm:px-6 lg:px-8">
        {/* Logo */}
        <Link href="/">
          <li className="text-2xl md:text-4xl font-bold flex items-center">
            <GiSofa className="text-purple-600 mr-2" />
            Sit <span className="text-primary">& Style Studio</span>
          </li>
        </Link>

        {/* Responsive Menu Button */}
        <button
          onClick={toggleMenu}
          className="block lg:hidden rounded-md bg-gray-100 p-2 text-gray-700 focus:outline-none focus:ring-2 focus:ring-primary"
        >
          {isMenuOpen ? "X" : "="}
        </button>

        {/* Navigation Links (Mobile & Desktop) */}
        <nav
          className={`absolute left-0 top-[4.5rem] z-50 w-full bg-white shadow-md transition-all duration-300 ease-in-out ${
            isMenuOpen ? "block" : "hidden"
          } lg:static lg:flex lg:w-auto lg:shadow-none`}
          style={{ top: "5rem" }} // Moves the menu slightly below the navbar
        >
          <div className="flex flex-col items-start space-y-4 px-6 py-4 lg:flex-row lg:items-center lg:space-y-0 lg:space-x-8 lg:p-0">
            {links.map((link, idx) => (
              <div key={idx}>
                <Link
                  href={link.href}
                  className={`text-lg font-semibold ${
                    pathname === link.href
                      ? "text-primary"
                      : "text-gray-600 hover:text-primary"
                  }`}
                  onClick={closeMenu} // Closes menu on click
                >
                  {link.name}
                </Link>
              </div>
            ))}
          </li>
        </nav>

        {/* Cart Button */}
        <div className="flex divide-x border-r sm:border-l">
          <Button
            variant={"outline"}
            onClick={() => handleCartClick()}
            className="flex flex-col gap-y-1.5 h-12 w-12 sm:h-20 sm:w-20 md:h-24 md:w-24 rounded-none"
          >
            <ShoppingBag />
            <span className="hidden text-xs font-semibold text-gray-500 sm:block">
              Cart
            </span>
          </Button>
        </div>
      </li>
    </header>
  );
}
```