I am Tabinda Aijza Ahmed Khan this is my day3 DOCUMENATION of Hackathon3

# **API Integration and Data Migration**

# 1. Project Title:

Dynamic Product Display Using Sanity (banked) and Next.js(frontend)

### 2. Objective:

The goal of this project was to fetch product data from Sanity CMS and display it dynamically on a Next.js frontend with proper styling and responsiveness.

### 3. Key Features:

Sanity Integration: Successfully connected Sanity CMS to Next.js using GROQ queries.

Dynamic Data Fetching: Used Sanity's APIs to retrieve product details, including name, price, description, and image.

Responsive Frontend Design: Built a responsive layout using Tailwind CSS, ensuring compatibility across devices.

Clean Code Structure: Used modular functions for fetching data and organized React components efficiently.

## 4. Technologies Used:

Frontend: Next.js (React Framework)

Backend: Sanity CMS

Styling: Tailwind CSS

Programming Language: TypeScript/JavaScript

# 5. Step-by-Step Implementation:

Set Up Sanity CMS:

Created a new dataset in Sanity.



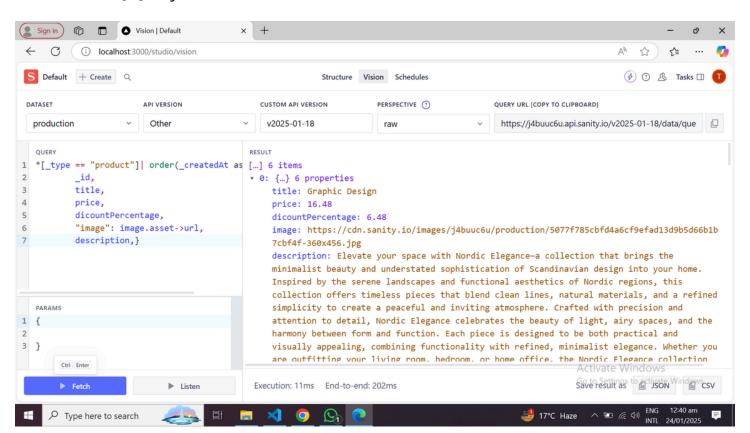
Added a schema for products with fields like name, price, description, image, and category.

#### Configured Sanity Client:

```
product.ts M ×
                                                                                                   src > sanity > schemaTypes > 🗋 product.ts > 🝘 product > 🔑 fields
       export const product = defineType({
            name: "product",
   4
            title: "Product",
   5
            type: "document",
   6
   7
            fields: [
  8
                     name: "title",
  9
                     title: "Title",
 10
                     validation: (rule) => rule.required(),
 11
                     type: "string"
 12
 13
```

Set up a reusable client instance to connect to Sanity.

#### Created GROQ Query:



Wrote a GROQ query to fetch the required product field. Fetched Data in Next. js:

Used a custom fetch Products function to call Sanity's APIs.

Managed the fetched data using React's useState and useEffect hooks.

Frontend Rendering:

```
0: □ □ □
	imes File Edit Selection View Go Run \cdots \leftarrow \rightarrow
                                                                                   ··· □ product.ts M □ page.tsx M × □ .env ↓D, M
                                                                                                                                                                            src > app > 🖰 page.tsx > 🗐 Home > 🗐 fetchProducts > 🗐 query
       ✓ OPEN EDITORS
                                              import Navbar from "@/components/Navbar";
           product.ts src\sanit... M
                                      6 import TopBar from "@/components/topnavbar";
         × 🖺 page.tsx src\app M
                                        7 import Whitepart from "@/components/white-part";
                                     import FeaturedCards from "@/components/featured";
import Footer from "@/components/footer";
import React, { useEffect, useState } from "react";
import SimpleSearchBar from "@/components/sreach_bar";
          n.env ↓D, M

✓ E-COMMERCE-WEBSITE

        ∨ 🗁 src
         ∨ 🗁 app
                                      import { client } from "@/sanity/lib/client";
import ProductsList from "@/components/productList";
import { Product } from "../../type"; // Assuming a proper Product type is defined in your type file
         ∨  products •
         ∨ 🗁 Shop
            page.tsx
          > 🗀 studio
                                      16 const Home = () => {
                                               const [products, setProducts] = useState<Product[]>([]); // State for storing products
            favicon.ico
                                              const [loading, setLoading] = useState<boolean>(true); // State for loading status
                                      18
            globals.css
            layout.tsx
                                      20
21
           page.tsx M
                                             setLoading(true); // Set loading to true before fetching const query = `*[ type == "analyse"]
         ✓ ☐ components✓ ☐ about

∨ 

    about

                                       23
                                                    const query = `*[_type === "product"] | order(_createdAt asc)[0..5]{
           > 🗀 pricing
                                       24
                                                       · id.
                                                      title,

∨  product

                                                      price,
              cart.tsx
                                                       dicountPercentage,
               companiesSection.tsx
                                                      "image": image.asset->url,
               flip.tsx M
                                                     --description,}`;
               product-sectio... M
                                       30
                                                    const data: Product[] = await client.fetch(query); // Fetch products from Sanity
               topnavbar.tsx
                                                    setProducts(data); // Update the products state
```

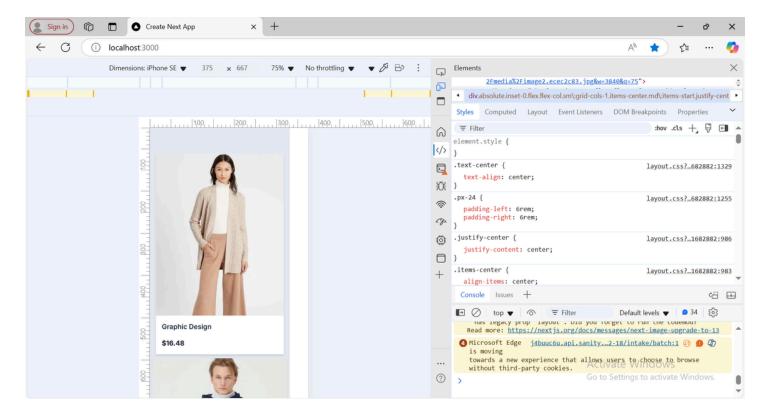
Dynamically rendered product details, including name, price, description, and images.

Used Image from Next.js for optimized image loading

Styling with Tailwind CSS:

Designed a responsive grid layout.

Styled individual product cards with hover effects for better user interaction.



## 6. Challenges and Solutions:

Dynamic Images from Sanity: Used next/image to generate URLs for Sanity assets.

Dynamic and Real-time Updates: Optimized GROQ queries for efficient data fetching and rendering.

### 7. Output:

A fully functional webpage that dynamically displays product data from Sanity CMS in a clean and responsive design.

### 8. Learning Outcome:

Gained hands-on experience with connecting a CMS to a modern frontend framework.

Writing GROQ queries for efficient data fetching.

Implementing responsive designs with Tailwind CSS.

### 9. Conclusion:

This project demonstrates the ability to integrate a CMS backend (Sanity) with a modern frontend framework (Next.js) for dynamic content rendering, providing a robust and scalable solution for real-world applications.

```
	imes File Edit Selection View Go Run \cdots \longleftrightarrow 	o
                                                                  83 ~
                                                                                                                                          Ø
      th □ ...
ф
     V OPEN EDITORS
                              src > app > page > [id] > ↑ page.tsx > ♦ CartPage > 🔊 discountedPrice
                                    default async function CartPage({ params: { id } }: { params: { id: string } }) {
         product.ts src\sanit... M
                                    !product) .{
       × 🖺 page.tsx src\app\pa... M
                               30 turn <div>Product not found!</div>;
                       ↓D, M
         n.env
                                31
     ✓ E-COMMERCE-W... [ □ □ □
      ∨ 🗁 src
                                33 t discountedPrice = product.price - (product.price * product.discountPercentage) / 100;
       ∨ 🗁 app
                                34
        > 🛅 about
                                35
                                    iv className="p-4 md:p-8 lg:p-12">
                                36
        ∨ 🔁 blog
                                37
                                    <div className="grid grid-cols-1 lg:grid-cols-2 gap-8" key={product._id}>
          page.tsx
                         M
                                      {/* Left Section */}
        > 🛅 contact
                                39
                                      <div className="flex flex-col items-center space-y-4">
        40
                                        <Image
                                41
                                          src={

→ Pricing

                                          product.image?.asset ? urlFor(product.image.asset).url() : "/images/placeholder.jpg"
                                42
                                43
          page.tsx
                                44
                                         alt={product.title}
        ∨ 🖒 products
                                45
                                         className="w-full h-auto lg:w-[450px] lg:h-[500px] rounded-lg shadow-lg transform transition duration-300
          page.tsx
                          R
                                46
                                          width={450}
        ∨ 🗁 Shop
                                47
                                          height={500}
          page.tsx
                                48
                                        priority
        > 🛅 studio
                                49
          favicon.ico
                                50
                                      </div>
                                51
          globals.css
                                52
                                      {/* Right Section */}
          layout.tsx
                         M
                                53
                                       <div className="space-y-6">
          page.tsx
                         M
                                54
                                        <h2 className="text-xl lg:text-2xl font-bold  text-gray-800">{product.title}</h2>
       > P1 components
                                        {product.description}
                                55
                                                                                                                  Activate Windows
     > OUTLINE
                                        <div className="flex items-center space-x-2">
                                56
     > TIMELINE
                                              .Array(4)].map(( , index) =>
```