Day 3 - API Integration Report - [E-Commerce Furniture Website]

---

1. Introduction

- This project is an **e-commerce furniture website**.
- Instead of using the provided API, I opted to manually add data to **Sanity CMS**.
- Sanity CMS was used to store and manage the data, which was then fetched into the project using GROQ queries and displayed on the frontend.

2. Data Integration Process

1. **Data Entry in Sanity CMS:**
    - Manually added furniture data (e.g., titles, prices, descriptions, and images).
    - Created separate documents for each furniture item.
2. **Sanity Setup in the Project:**
    - Configured the Sanity client using @sanity/client.
    - Wrote GROQ queries to fetch the data dynamically.
3. **Frontend Rendering:**
    - Used React components to display data fetched from Sanity CMS.
    - Implemented a clean UI to showcase product details like titles, images, and prices.

3. Adjustments Made to Schemas

- I customized the Sanity CMS schema to include the following fields:
    - **Title:** Name of the furniture item.
    - **Price:** Cost of the product.
    - **Description:** Details about the product.
    - **Image URL:** Path to the product image.

4. Migration Steps and Tools Used

- Since I manually added data directly into Sanity CMS, there were no external migration tools required.
- Sanity's built-in schema and data editor were used for setup and management.

5.Screenshots to Include

1. **Sanity CMS Schema Setup:** Show how the schema was structured.

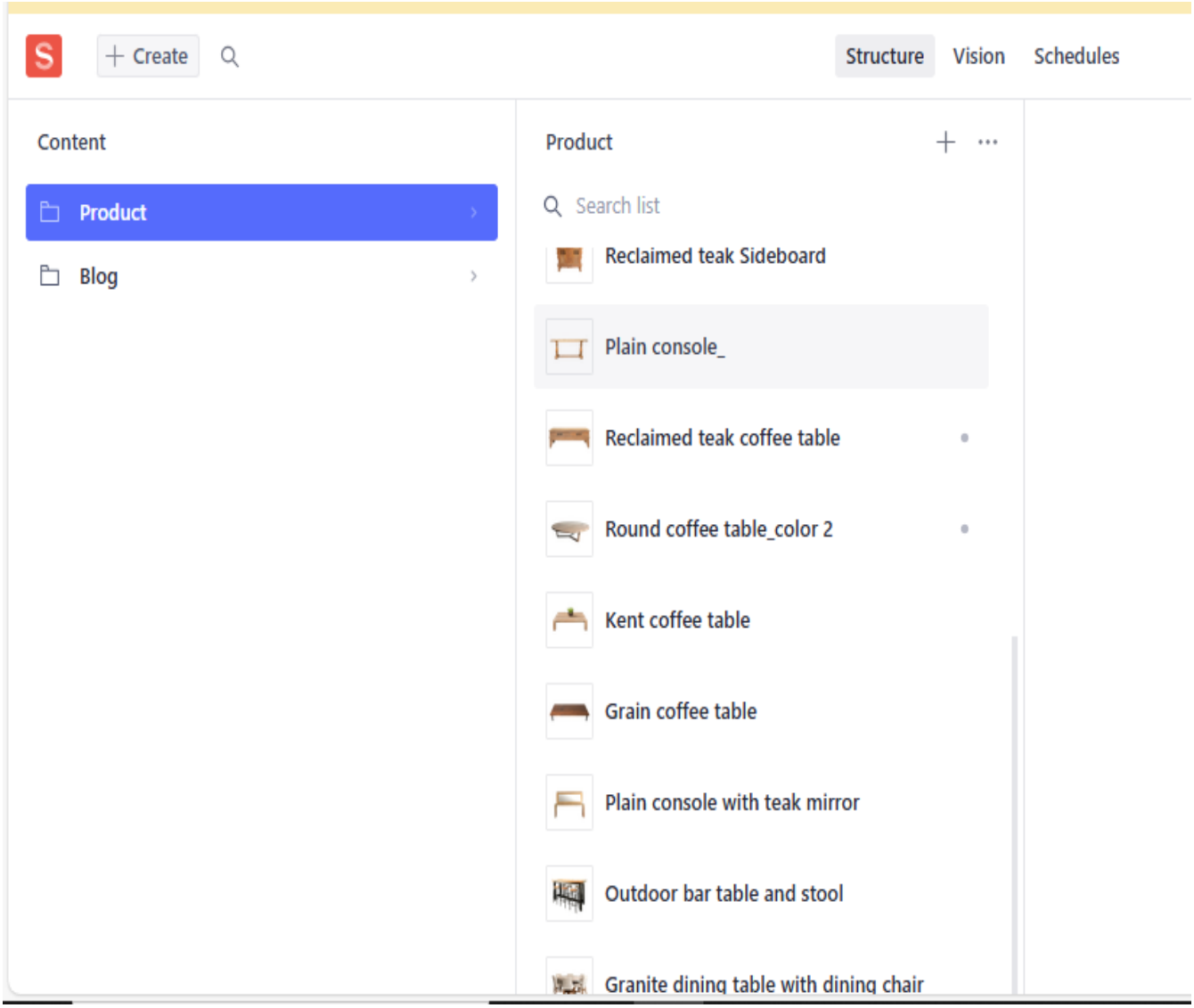
product.ts

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

```
import {defineField, defineType, defineArrayMember} from 'sanity';

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
      defineField({
        name: "name",
        title: "Product Name",
        type: "string",

      }),
      {
        name: "price",
        title: "Price",
        type: "number",

      },
      defineField({
        name: "Paragraph",
        title: "Paragraph",
        type: "text",

      }),
      defineField({
        name: "image",
        title: "Image",
        type: "image",
        options: {
          hotspot:true
        }
```

```
34      {
35          name: "thumbnailImages",
36          title: "Thumbnail Images",
37          type: "array",
38          of: [{ type: "image" }],
39      },
40      defineField({
41        name: "slug",
42        type: "slug",
43        title: "Slug",
44        options: {
45          source: "name",
46          maxLength: 96,
47          slugify: (input) =>
48            input
49              .toLowerCase()
50              .replace(/\s+/g, "-")
51              .slice(0, 96),
52        },
53      }),
54  defineField({
55      name: "block",
56      title: "Block",
57      type: "array",
58      of:[{type:'block'}]
59  }),
60      ]
61
62  })
```

2. **Populated Data in Sanity CMS:** The data entries I manually added.



3. **API Calls in the Code:** Screenshot of my GROQ queries fetching the data.

page.tsx M ×

src > app > shop > page.tsx > Shop

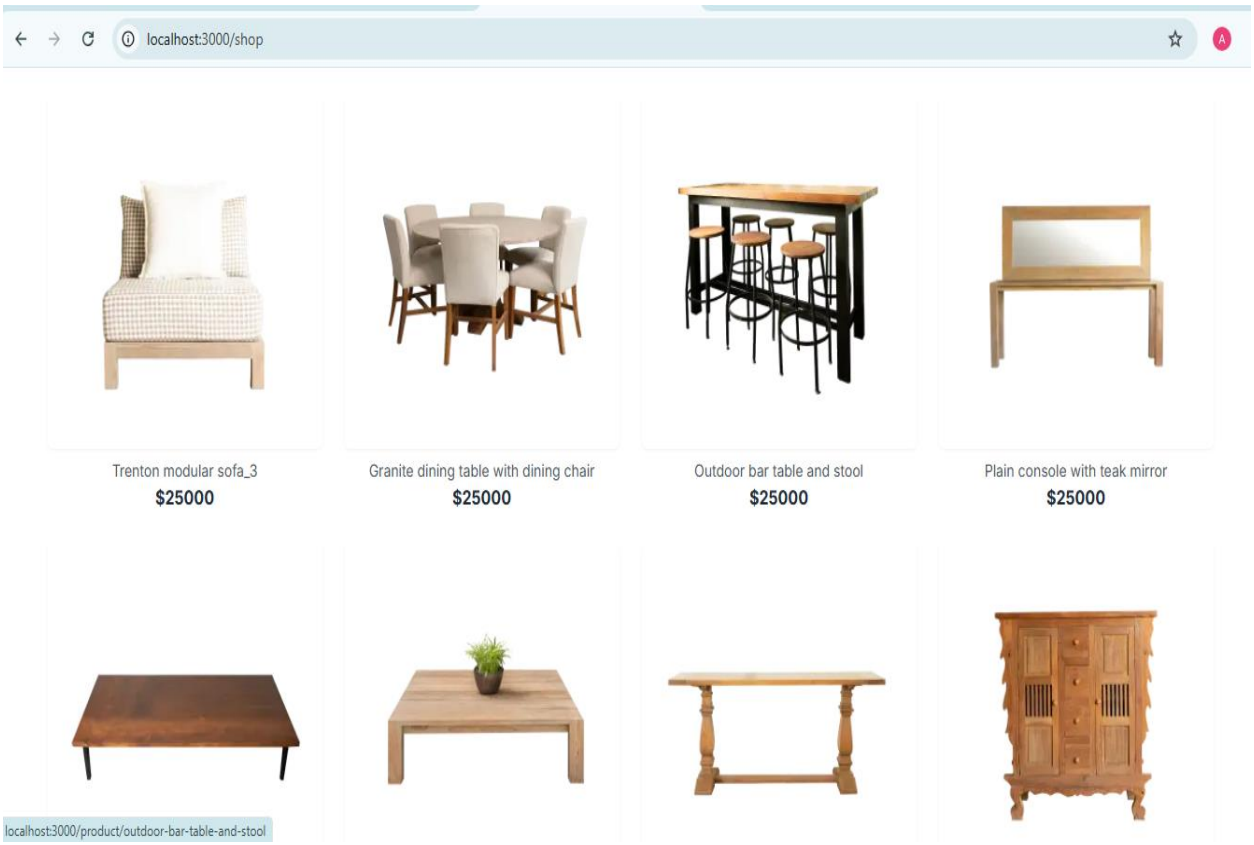
```
import { client } from "@/sanity/lib/client";
import { urlFor } from "@/sanity/lib/image";
import Image from "next/image";
import Link from "next/link";


interface Product {
  name: string;
  image: { asset: { url: string } };
  price: number;
  slug: { current: string };
}

const Shop = async () => {

  const query = `*[_type == 'product'] | order(_updatedAt asc) {
    name,
    image,
    price,
    slug
  }`;

  // Fetch data from Sanity
  const data: Product[] = await client.fetch(query);

```

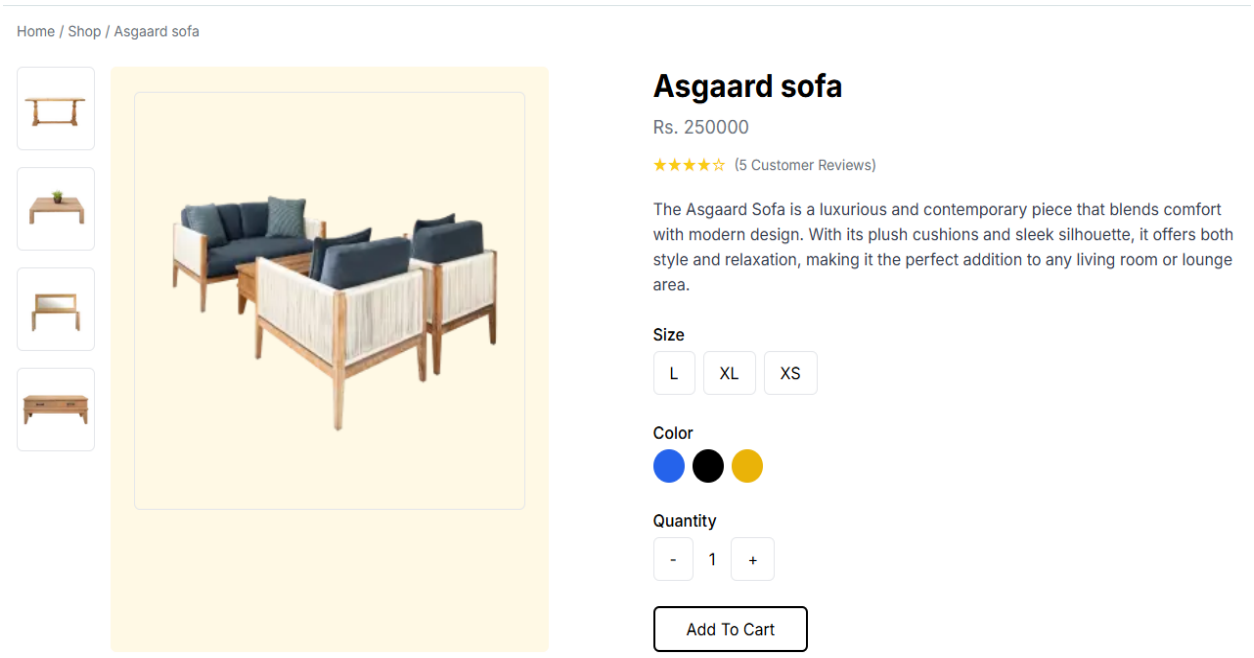
page.tsx U ×

src > app > product > [slug] > page.tsx > ...

```
1   'use client';
2   import { useState, useEffect, Key } from 'react';
3   import { client } from '@/sanity/lib/client';
4   import Link from 'next/link';
5   import Image from 'next/image';
6   import { PortableText } from 'next-sanity';
7   import { urlFor } from '@/sanity/lib/image';
8   import { useCart } from '@/context/cartContext';  // Import the useCart hook
9
10  const ProductPage = ({ params: { slug } }: { params: { slug: string } }) => {
11    const [productData, setProductData] = useState<any>(null);
12    const [quantity, setQuantity] = useState(1);
13    const { cartItems, addToCart } = useCart();  // Access cart state and addToCart from context
14
15    useEffect(() => {
16      const fetchProductData = async () => {
17        const query = `*[_type == 'product' && slug.current == '${slug}']{
18          name, price, Paragraph, image, thumbnailImages, block
19        }`;
20
21        const data = await client.fetch(query);
22        setProductData(data[0] || null);
23      };
24
25      fetchProductData();
26    }, [slug]);
27
```

4. **Frontend Display:** Here's how the furniture items are being displayed on my website (titles, images, etc.).





6. Challenges and Learnings

- **Challenges:**
  - Understanding GROQ syntax and applying filters for specific data.
  - Configuring Sanity CMS schema to suit the project requirements.
- **Learnings:**
  - Gained hands-on experience with Sanity CMS and GROQ queries.
  - Learned how to structure and fetch data efficiently for dynamic websites.

7. Conclusion

- Successfully integrated Sanity CMS with the project to fetch and display data.
- Despite not using the provided API, the workflow is seamless, and the website displays all furniture data dynamically.