Day 3: API Integration and Data Migration Documentation

Objective

The primary focus of Day 3 was to integrate APIs, migrate data into Sanity CMS, and render it dynamically on the frontend using GROQ queries and Next.js. This task replicates real-world practices in building scalable and flexible marketplaces using modern technologies.

Key Achievements

Sanity Data Integration: Successfully migrated data from external sources to Sanity CMS.

Frontend Rendering : Used GROQ queries to fetch data from Sanity CMS and rendered it dynamically on the UI.

Schema Adjustments : Customized the Sanity schema to align with data requirements.

Tech Stack: Next.js, Sanity CMS, GROQ, Tailwind CSS.

Sanity Schema

```
import { defineType } from "sanity";

export default defineType({
  name: 'products',
  title: 'Products',
  type: 'document',
  fields: [
    { name: 'name', title: 'Name', type: 'string' },
    { name: 'price', title: 'Price', type: 'number' },
    { name: 'description', title: 'Description', type: 'text' },
    { name: 'image', title: 'Image', type: 'image' },
    {
```

```
      name: "category",
      title: "Category",
      type: 'string',
      options: {
        list: [
          { title: 'T-Shirt', value: 'tshirt' },
          { title: 'Short', value: 'short' },
          { title: 'Jeans', value: 'jeans' },
          { title: 'Hoodie', value: 'hoodie' },
          { title: 'Shirt', value: 'shirt' },
        ]
      }
    },
    { name: "discountPercent", title: "Discount Percent", type: 'number' },
    { name: "new", title: "New", type: 'boolean' },
    {
      name: "colors",
      title: "Colors",
      type: 'array',
      of: [{ type: 'string' }]
    },
    {
      name: "sizes",
      title: "Sizes",
      type: 'array',
      of: [{ type: 'string' }]
    }
  ],
});
```

```

# GROQ Queries

## Query 1: Fetch T-Shirts

const query:IProduct[] = await client.fetch(`*[_type == "products" && category == "tshirt"][0...4] {

_id,

name,

price,

description,

image,

}`);

```

Query 2: Fetch New Products

```javascript

const query:IProduct[] = await client.fetch(`*[_type == "products" && new == true][0...4] {

_id,

name,

price,

description,

image,

}`);

```

API Integration Steps

Data Population in Sanity CMS

1. Migrated product data into Sanity CMS using scripts and manual uploads.

2. Adjusted field mappings to ensure seamless integration between the API and the schema.

Frontend Integration

1. Created a utility function using Axios to fetch GROQ query results from the Sanity API.

2. Rendered the fetched data dynamically in Next.js components.

Rendering Data in Next.js

```jsx
<div className="md:flex justify-center gap-5">
 {query.map((item) => (
  <div key={item._id}>
   <Image
    src={urlFor(item.image).url()}
    height={292}
    width={270}
    alt={item.name}
    className="object-cover rounded-md h-[290px] w-full"
   />
   <p className="truncate w-[250px] h-[40px] lg:text-[17px] text-[12px] font-bold py-2">
    {item.name}
   </p>
```

```
    <div className="text-[#FFC633] lg:w-[150px] lg:h-[19px] w-[127px] h-[16px] flex gap-1">
      <IoIosStar />
      <IoIosStar />
      <IoIosStar />
      <FaStarHalf />
      <p className="w-[33px] h-[19px] text-[14px] text-[#000000]">4.5/5</p>
    </div>
    <h5 className="w-[55px] h-[32px] text-[24px] text-[#000000] font-bold">
      ${item.price}
    </h5>
  </div>
 ))}
</div>
```