Comforty: Luxury Chairs and Sofas

Technical Documentation
Day 3: API Integration and Fetching Data from Sanity

Table of Contents

---

1. Introduction

On Day 3, we focused on integrating the Sanity CMS API into the Comforty platform to dynamically fetch product data. This involved reviewing the Sanity CMS documentation, adjusting the schema, and ensuring seamless data flow between the backend and frontend.

---

2. Reviewed API Documentation

- Carefully reviewed the Sanity CMS documentation to understand the data schema for products.
- Manually added product data into Sanity CMS using the Sanity Studio.
- Ensured that the schema fields in Sanity matched the API structure, including field names and data types.

3. Dynamic API Fetching

Steps for Dynamic API Fetching

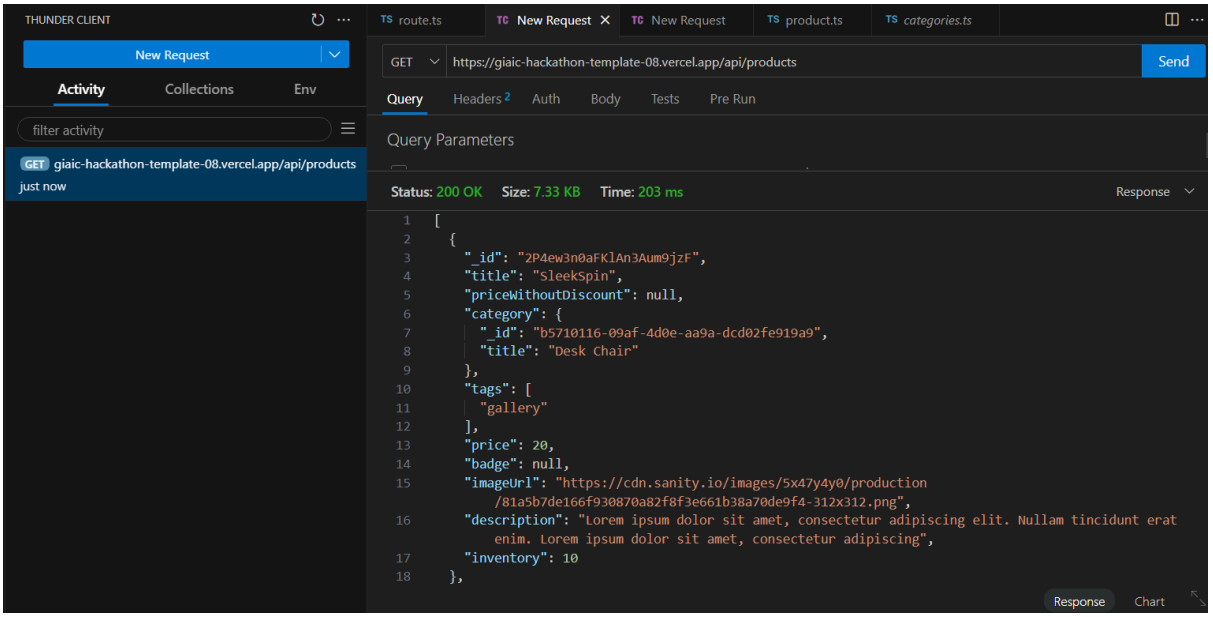
Install Sanity Client:

npm install @sanity/client

Create API Utility File:
I created a utility file (sanityClient.ts) for Sanity configuration:

```
import { createClient } from 'next-sanity'

import { apiVersion, dataset, projectId } from '../env'

export const client = createClient({
  projectId,
  dataset,
  apiVersion,
  useCdn: true, // Set to false if statically generating pages, using ISR or tag-based revalidation
})

```

API Test:


THUNDER CLIENT
New Request
Activity
Collections
Env
filter activity
GET giaic-hackathon-template-08.vercel.app/api/products
just now
route.ts
New Request
New Request
product.ts
categories.ts
GET
https://giaic-hackathon-template-08.vercel.app/api/products
Send
Query
Headers 2
Auth
Body
Tests
Pre Run
Query Parameters
Status: 200 OK
Size: 7.33 KB
Time: 203 ms
Response
[
{
"_id": "2P4ew3n0aFKlAn3Aum9jzF",
"title": "SleekSpin",
"priceWithoutDiscount": null,
"category": {
"_id": "b5710116-09af-4d0e-aa9a-dcd02fe919a9",
"title": "Desk Chair"
},
"tags": [
"gallery"
],
"price": 20,
"badge": null,
"imageUrl": "https://cdn.sanity.io/images/5x47y4y0/production/81a5b7de166f930870a82f8f3e661b38a70de9f4-312x312.png",
"description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam tincidunt erat enim. Lorem ipsum dolor sit amet, consectetur adipiscing",
"inventory": 10
},
Response
Chart

Fetch Data From Sanity:

```
import { client } from "./client";

export async function fetchProducts() {
  const query = `*[_type == "products"] {
    _id,
    title,
    price,
    priceWithoutDiscount,
    badge,
    image {
      asset -> {
        _id,
        url
      }
    },
    category -> {
      _id,
      title
    },
    description,
    inventory,
    tags
  }`;

  const products = await client.fetch(query);
  return products;
}

```

Fetch Product Data in Component:

```
useEffect(() => {
   const fetchData = async () => {
     const data = await fetchProducts();
     setProducts(data.slice(7, 16)); // Display 8 products
   };
   fetchData();
 }, []);
```

---

4. Schema Adjustment in Sanity CMS

Product Schema:

```
{
     name: "category",
     title: "Category",
     type: "reference",
     to: [{ type: "categories" }],
   },
```

```
{

     name: "tags",

     title: "Tags",

     type: "array",
```

```
      of: [{ type: "string" }],
      options: {
        list: [
          { title: "Featured", value: "featured" },
          {
            title: "Follow products and discounts on Instagram",
            value: "instagram",
          },
          { title: "Gallery", value: "gallery" },
        ],
      },
    },
```

Migration Script:

```
// 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}`);
```

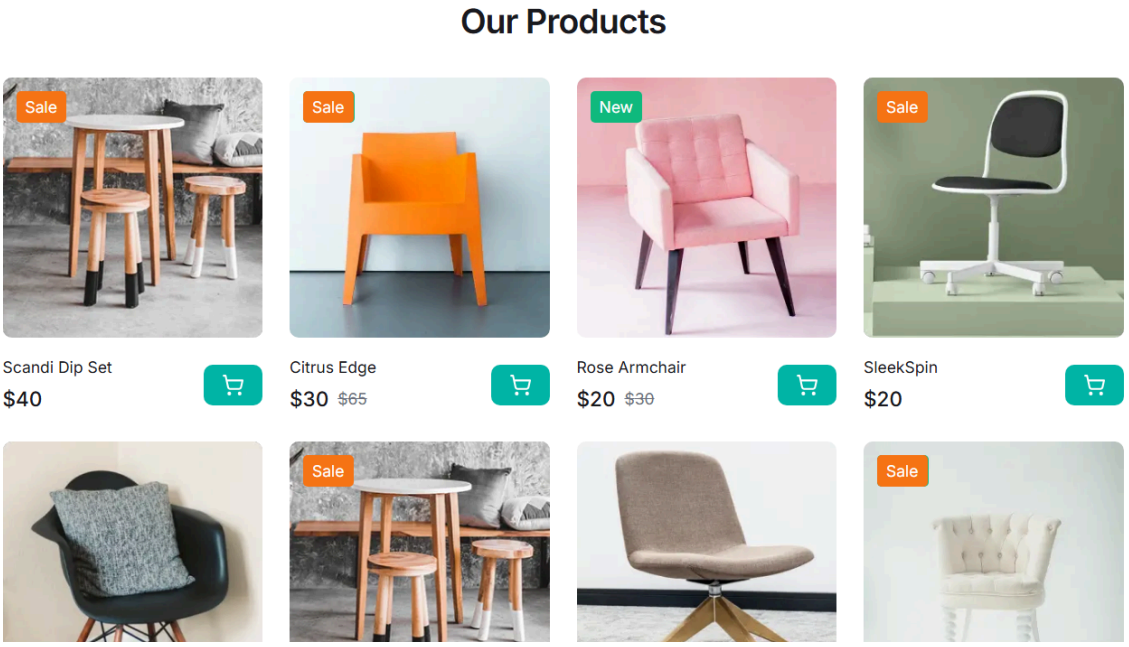
```
    // Convert the image to a buffer (binary format)
    const buffer = await response.arrayBuffer();

    // 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.");
```

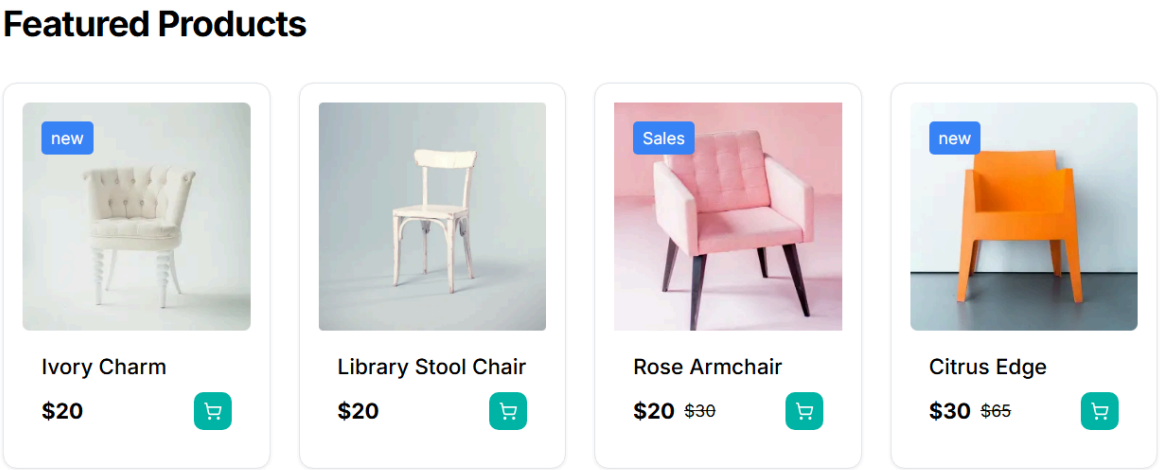
```
    const categoriesData = await categoriesResponse.json(); // Parse
response to JSON


    // 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: "products",

        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();
```

Data Successfully Displayed in the Frontend:





---

5. Results

- Successfully added product data manually to Sanity CMS.
- Fetched data dynamically from Sanity using its API in Next.js.
- Ensured accurate data rendering in the UI with schema-API compatibility.

---

6. Task Status

- Sanity Data Entry: ✓
- Dynamic API Fetching: ✓
- Frontend Rendering: ✓
- Schema Compatibility: ✓

---

7. Achievements

This task enhanced my practical skills in:

1. Configuring and using Sanity CMS with Next.js.
2. Creating dynamic and scalable data pipelines for modern web applications.
