Day 3: API Integration and Data Migration

Today's task involved API integration and data migration into Sanity CMS. The focus was on importing product data from an external API into Sanity CMS and then fetching it dynamically for the frontend. This was a multi-step process, and I enjoyed every bit of it! 🚀

What I Did Today

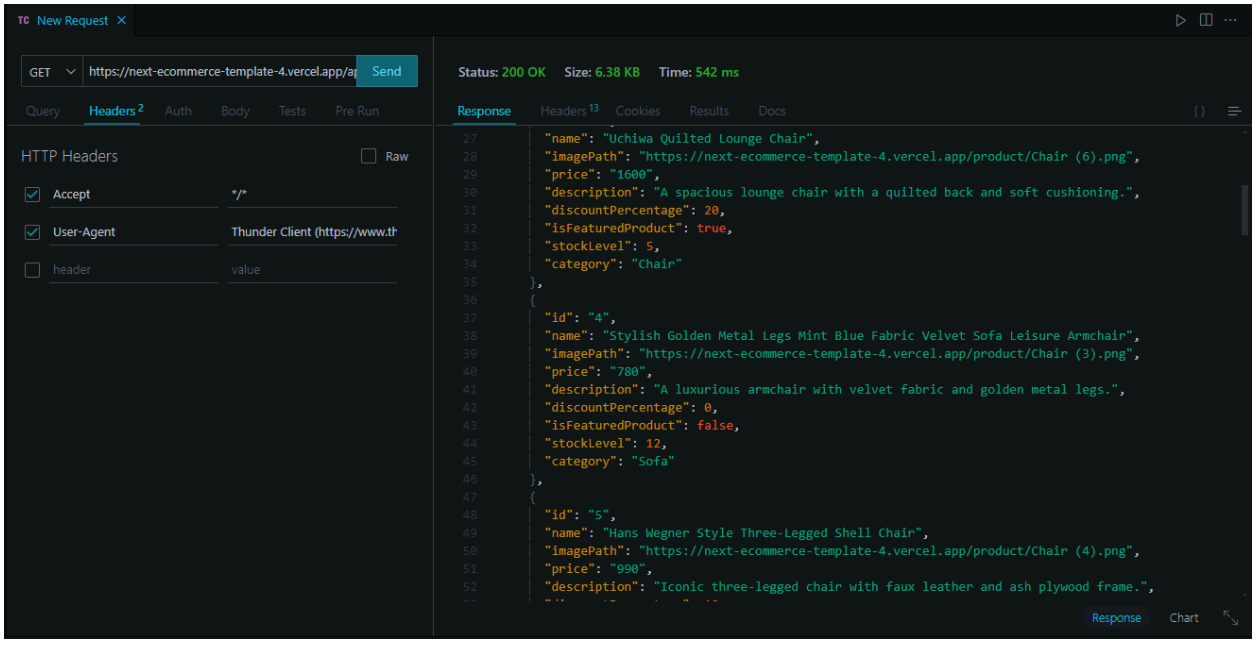
1. API Testing with Thunder Client

Before importing data, I tested the API endpoints using Thunder Client to ensure everything was working correctly.

Example Endpoint Tested:

- /products

Response Example:



---

2. Imported Data into Sanity CMS

To populate Sanity CMS with API data, I used the provided data migration script. This script fetched data from the external API and imported it into Sanity CMS using its APIs.

Script Workflow:

1. Fetch data from the external API.
2. Map the API fields to the Sanity schema fields.
3. Push the transformed data into Sanity CMS.

During Import:

- Ensured that API field names (like product_title) matched Sanity schema fields (like name).
- Validated data before pushing it into Sanity CMS.

2. Updated the Sanity CMS Schema

I made significant updates to the **Product Schema** in Sanity CMS to enhance its functionality and adaptability.

Key Changes I Made:

- Introduced discountPercentage, isFeaturedProduct, tags, and color for added flexibility.
- Replaced static category options with dynamic references to a Category Schema.

3. Data Fetching from Sanity CMS

I created a utility function getAllProducts.ts in my project library to fetch data dynamically from Sanity CMS.

4. Frontend Integration

I integrated the getAllProducts function into various components and pages, such as:

- Featured Products Section
- Latest Products on the Shop Page
- Dynamic Product Details Page using Routing

Example for Dynamic Routing (Single Product Page):

What's Next:

Moving forward, I'll work on further enhancing the user experience and adding additional features like product filtering, sorting, and SEO optimizations.

This journey has been incredibly fulfilling so far, and I'm excited to see what Day 4 brings! 🚀

Let me know your thoughts, feedback, or suggestions. 😊

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

export default defineType({
  name: "product",
  type: "document",
  title: "Product",
  fields: [
    defineField({
      name: "name",
      type: "string",
      title: "Name",
      validation: (Rule) => Rule.required().error("Name is required"),
    }),
    defineField({
      name: "image",
      type: "image",
      title: "Image",
      options: {
        hotspot: true,
      },
      description: "Upload an image of the product.",
    }),
    defineField({
      name: "price",
      type: "string",
      title: "Price",
      validation: (Rule) => Rule.required().error("Price is required"),
    }),
    defineField({
      name: "description",
      type: "text",
      title: "Description",
      validation: (Rule) =>
        Rule.max(150).warning("Keep the description under 150 characters."),
    }),
    defineField({
      name: "discountPercentage",
      type: "number",
      title: "Discount Percentage",
      validation: (Rule) =>
        Rule.min(0).max(100).warning("Discount must be between 0 and 100."),
    }),
    defineField({
      name: "isFeaturedProduct",
      type: "boolean",
      title: "Is Featured Product",
    }),
    defineField({
      name: "stockLevel",
      type: "number",
      title: "Stock Level",
      validation: (Rule) =>
        Rule.min(0).error("Stock level must be a positive number."),
    }),
    defineField({
      name: "category",
      type: "reference",
      title: "Category",
      to: [{ type: "category" }],
      validation: (Rule) => Rule.required().error("Category is required"),
    }),
    defineField({
      name: "color",
      title: "Color",
      type: "array",
      of: [{ type: "string" }],
    }),
    defineField({
      name: "tags",
      title: "Tags",
      type: "array",
      description: "Add tags for SEO",
      of: [{ type: "string" }],
    }),
  ],
});

```

```
import {client } from "@/sanity/lib/client"
export const getAllProducts = async () => {
  const query = `*[_type == "product"]{
    _id,
    name,
    image{
      asset->{
        url
      }
    },
    price,
    description,
    discountPercentage,
    isFeaturedProduct,
    stockLevel,
    category->{
      title
    },
    color,
    tags,
    _createdAt
  }`;

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

```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../../.env') });

const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2025-01-15',
  useCdn: false,
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading Image : ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop(),
    });
    console.log(`Image Uploaded Successfully : ${asset._id}`);
    return asset._id;
  }
  catch (error) {
    console.error('Failed to Upload Image:', imageUrl, error);
    return null;
  }
}

async function importData() {
  try {
    console.log('Fetching Product Data From API ...');

    const response = await axios.get("https://next-ecommerce-template-4.vercel.app/api/product")
    const products = response.data.products;

    for (const item of products) {
      console.log(`Processing Item: ${item.name}`);

      let imageRef = null;
      if (item.imagePath) {
        imageRef = await uploadImageToSanity(item.imagePath);
      }

      const sanityItem = {
        _type: 'product',
        name: item.name,
        category: item.category || null,
        price: item.price,
        description: item.description || '',
        discountPercentage: item.discountPercentage || 0,
        stockLevel: item.stockLevel || 0,
        isFeaturedProduct: item.isFeaturedProduct,
        image: imageRef
          ? {
              _type: 'image',
              asset: {
                _type: 'reference',
                _ref: imageRef,
              },
            }
          : undefined
```