DAY 3 - API INTEGRATION AND DATA MIGRATION

Prepared by
Urooj Sadiq

Prepared by: **Urooj Sadiq**

API INTEGRATION AND DATA MIGRATION –

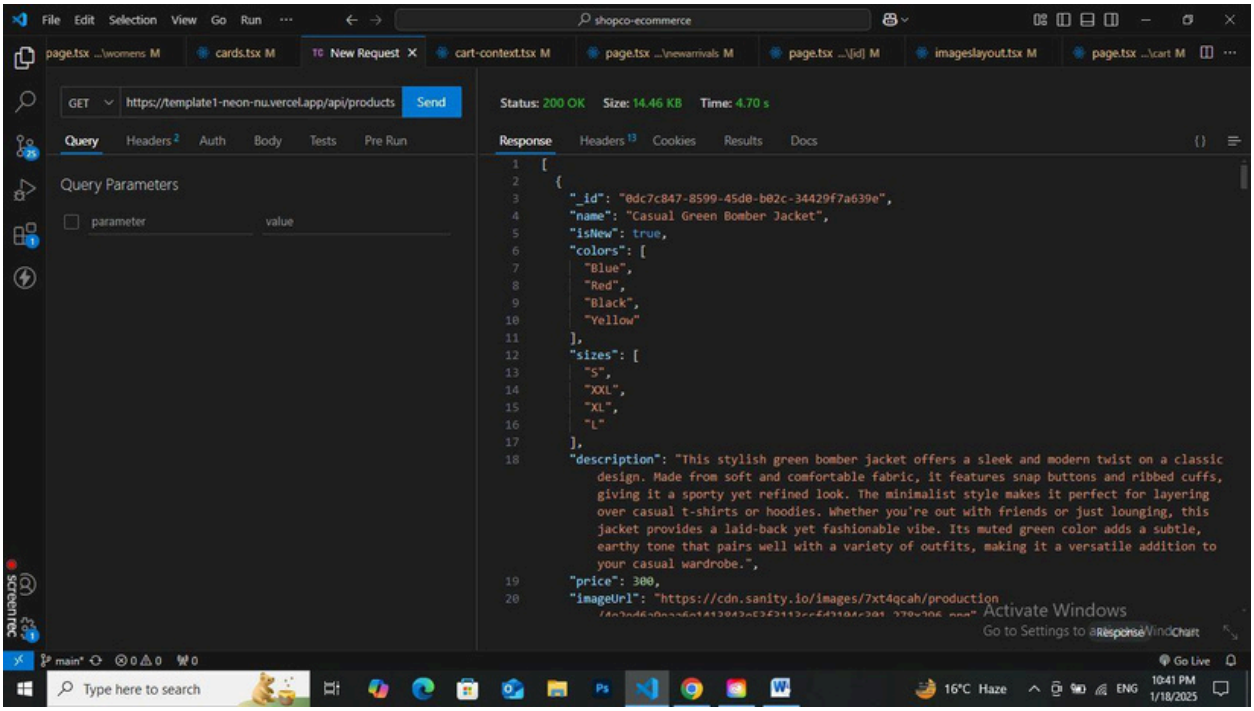
[SHOP.COM WEBSITE]

In this project, I built a simple and efficient way to migrate and import data from an external API into Sanity CMS. The process begins by fetching data through a Next.js application, then transforming it to match the structure of Sanity CMS. Once the data is ready, I send it to Sanity using its API.

I created reusable functions to make the code easy to scale. This solution not only makes data migration easier but also ensures that the data fits perfectly into the Sanity CMS structure. Below is a step-by-step guide on how I achieved this.

1. API Integration Process.

To test the API endpoint and verify that the data was being returned correctly, I used Thunder Client. In my Next.js project, I created utility functions to handle the process of fetching data from the API. I utilized the fetch method to make GET requests to the API endpoints and stored the responses in variables.



2. Adjustments Made to Schemas

After fetching the data, the next step was to compare the structure of the API data with the Sanity CMS schema. The Sanity schema defines how content is organized and stored within the CMS. For this project, a product schema was created to handle product data, with fields such as:

- Name (String)
- Price (Number)
- Description (Text)
- Image (Image)
- Category (String with predefined options)
- Discount Percent (Number)
- New (Boolean)
- Colors (Array of Strings)
- Sizes (Array of Strings)

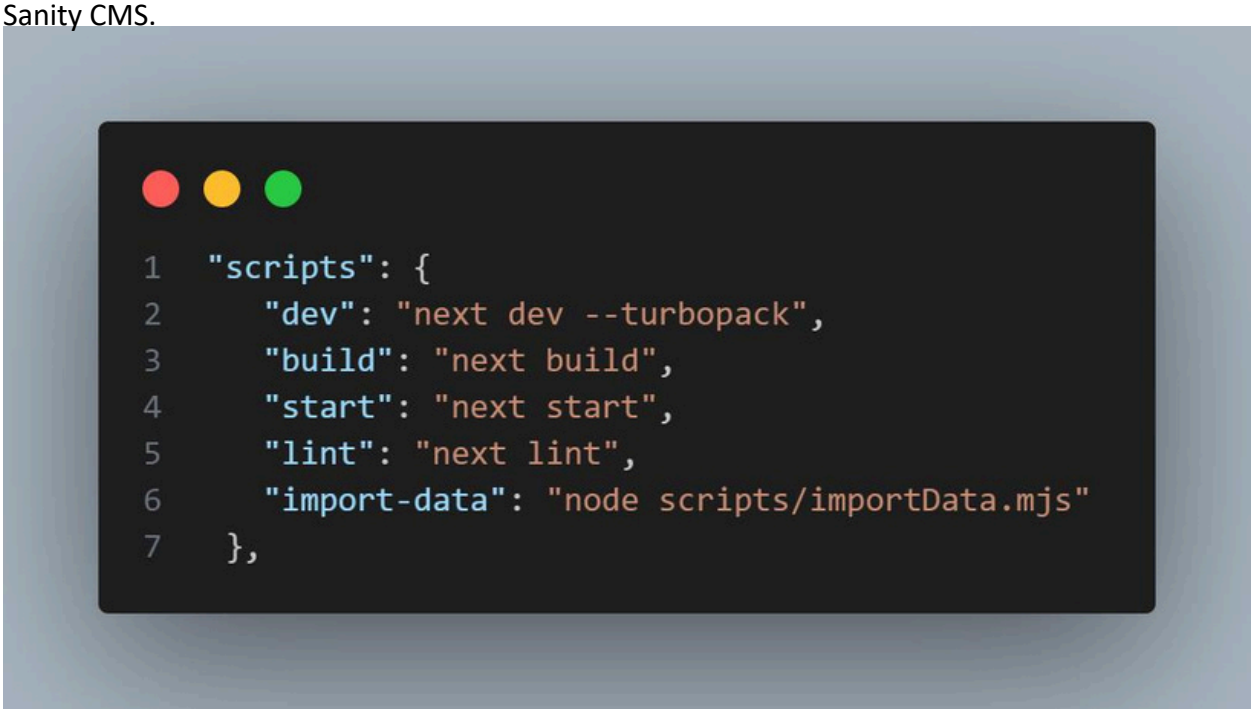
This comparison was essential to ensure that the data fetched from the API conformed to the structure specified in the Sanity schema. By validating that each field in the API response had a corresponding field in the Sanity schema, I ensured smooth data integration and compatibility.

```
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: 'Hoddie', value: 'hoodie'} ,
                    {title: 'Shirt', value: 'shirt'} ,
                ]
            }
        },
        {
            name:"discountPercent",
            title:"Discount Percent",
            type: 'number',
        },
        {
            name:"new",
            type: 'boolean',
            title:"New",
        },
        {
            name:"colors",
            title:"Colors",
            type: 'array',
            of:[
                {type: 'string'}
            ]
        },
        {
            name:"sizes",
            title:"Sizes",
            type: 'array',
            of:[
                {type: 'string'}
            ]
        }
    ],
})
```

3. Migration Steps and Tools Used

Here’s how I imported data from the API into Sanity CMS:

- First, I used the provided API to fetch the data and wrote a simple script to bring it into Sanity CMS.

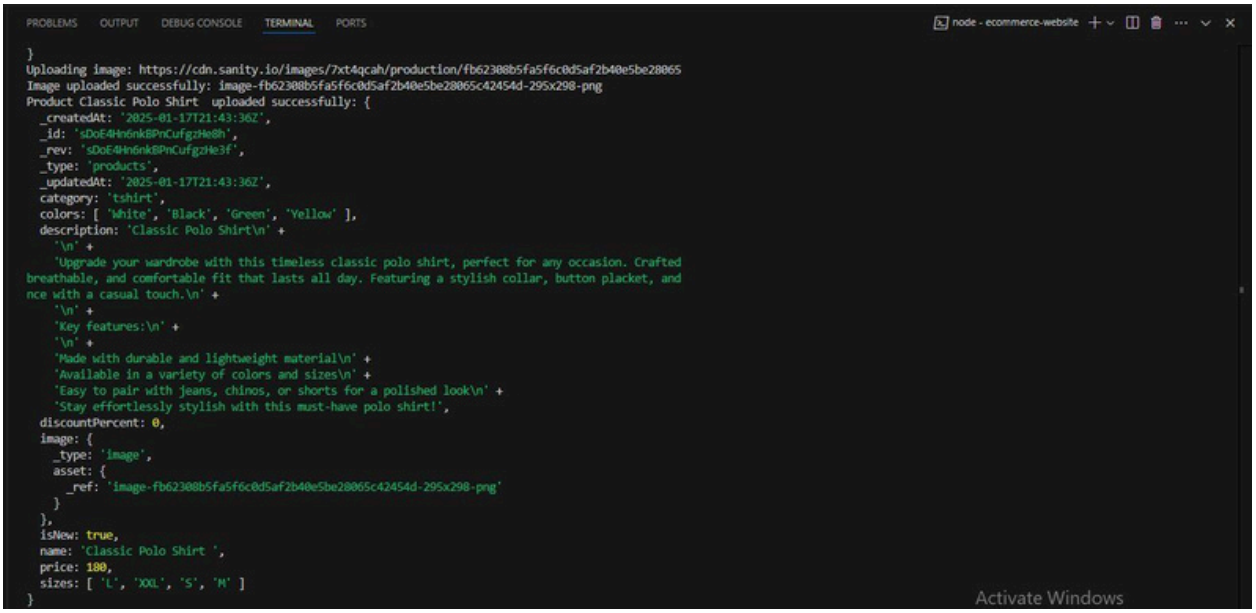


```
"scripts": {
   "dev": "next dev --turbopack",
   "build": "next build",
   "start": "next start",
   "lint": "next lint",
   "import-data": "node scripts/importData.mjs"
 },
```

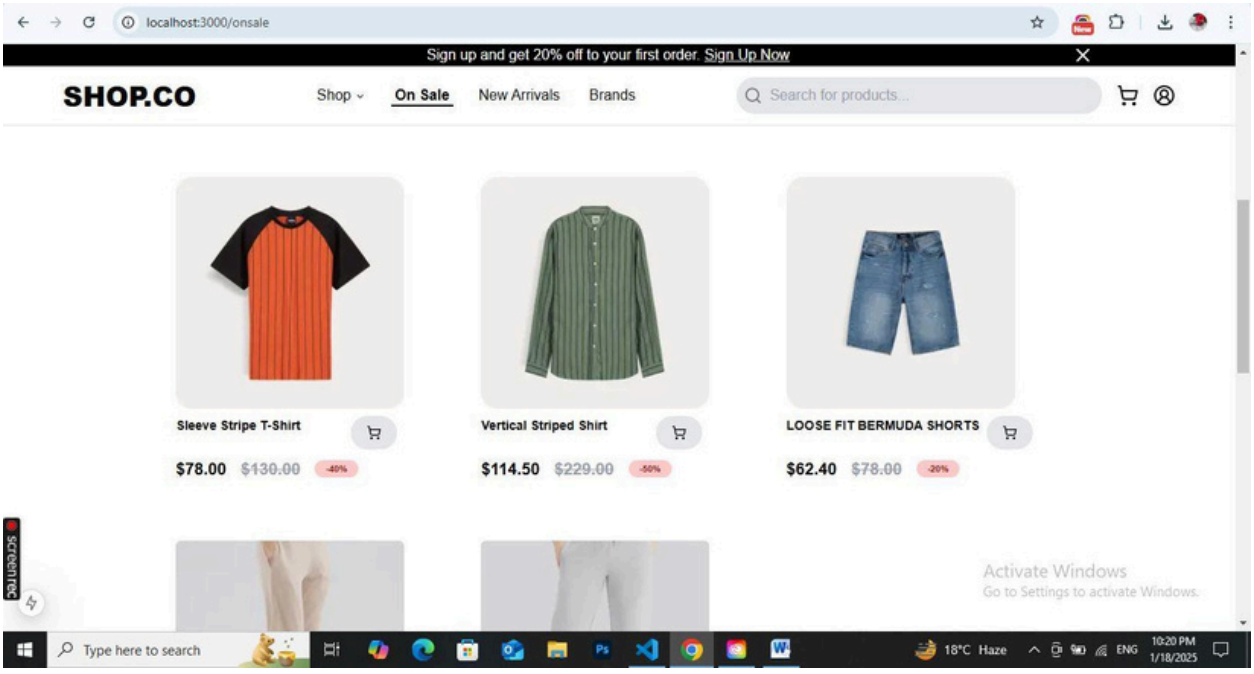
- I created an importData (.mjs) file in a special folder to fetch the data from the API and adjust it to fit Sanity’s format.

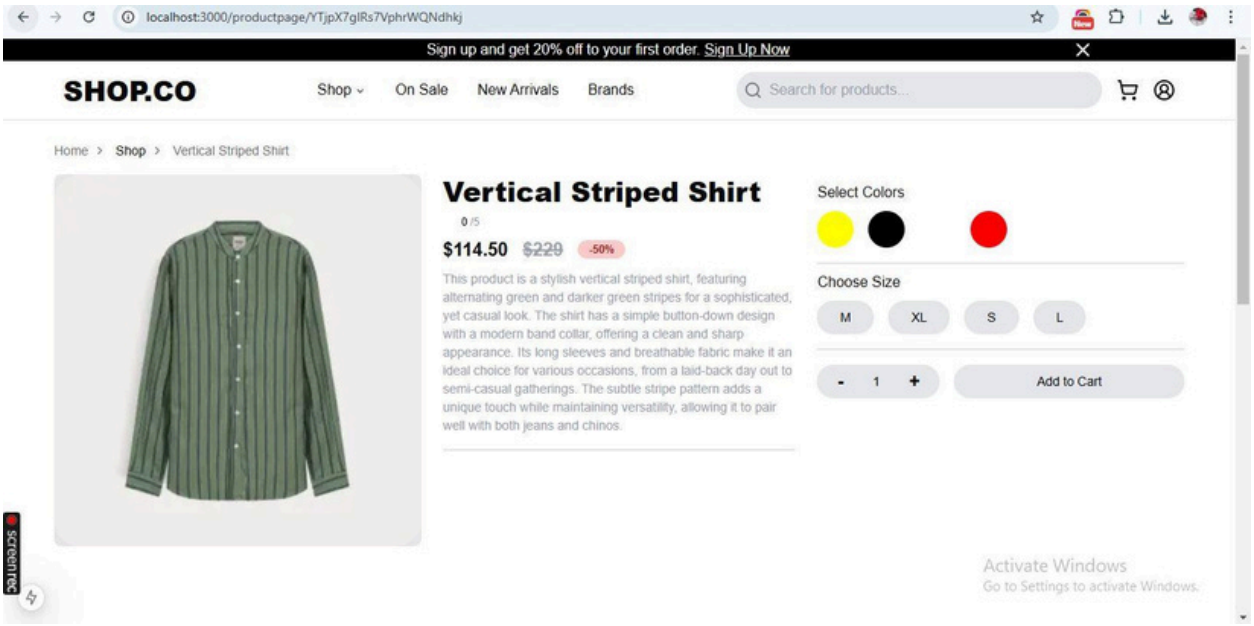
```
import { createClient } from '@sanity/client'
import dotenv from 'dotenv'
import { fileURLToPath } from 'url'
import path from 'path'

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url)
const __dirname = path.dirname(__filename)
dotenv.config({ path: path.resolve(__dirname, '../.env.local') })

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31'
})

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      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 uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: 'products',
        name: product.name,
        description: product.description,
        price: product.price,
        image: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        category: product.category,
        discountPercent: product.discountPercent,
        isNew: product.isNew,
        colors: product.colors,
        sizes: product.sizes
      };

      const createdProduct = await client.create(document);
      console.log(`Product ${product.name} uploaded successfully:`, createdProduct);
    } else {
      console.log(`Product ${product.name} skipped due to image upload failure.`);
    }
  } catch (error) {
    console.error('Error uploading product:', error);
  }
}

async function importProducts() {
  try {
    const response = await fetch('https://template1-neon-nu.vercel.app/api/products');

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error('Error fetching products:', error);
  }
}

importProducts();
```

- Using Sanity's client library, I uploaded the transformed data to the CMS. I ran the script npm run import-data to import all the product details, categories, and other relevant information.



- After the import, I checked the Sanity dashboard to make sure everything was correct and all fields were filled properly.





Self-Validation Checklist:

Task	Status
API Understanding	✓
Schema Validation	✓
Data Migration	✓
API Integration in Next.js	✓
Submission Preparation	✓