Day 3 - API Integration Report - **Comforty Chair Marketplace**

### Introduction

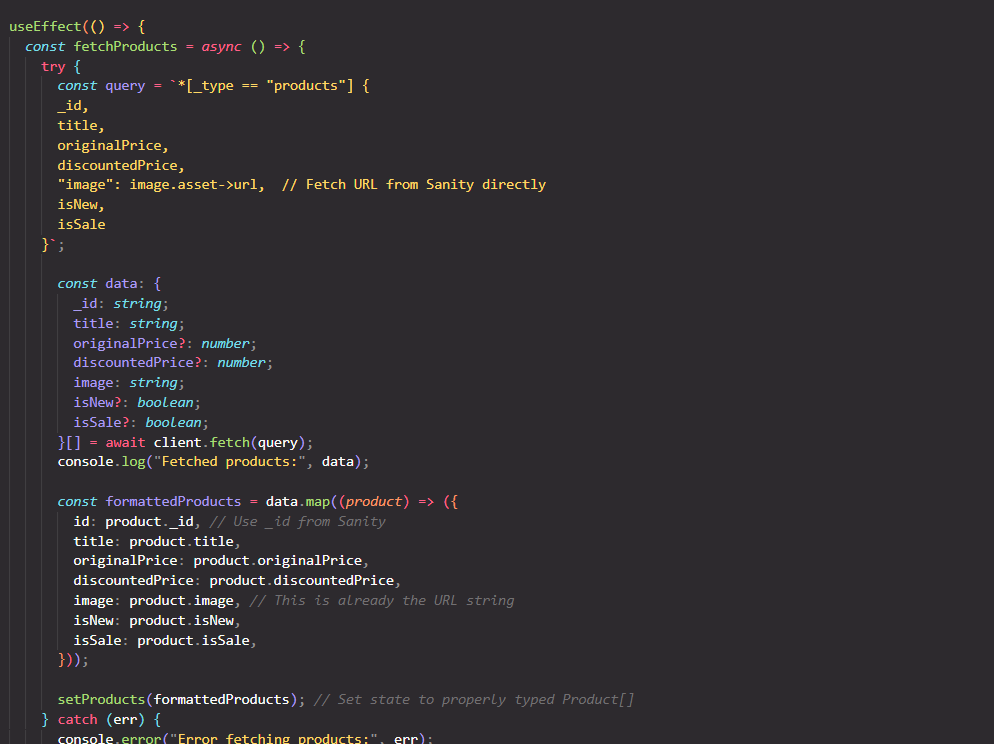
This report outlines the process of integrating APIs, making necessary adjustments to the schemas, and migrating data to the Sanity CMS for **Comforty Chair Marketplace**. The goal was to enable seamless API interaction, integrate external product data, and ensure the CMS is populated and functional.

### API Integration Process

#### 1. ****API Integration****

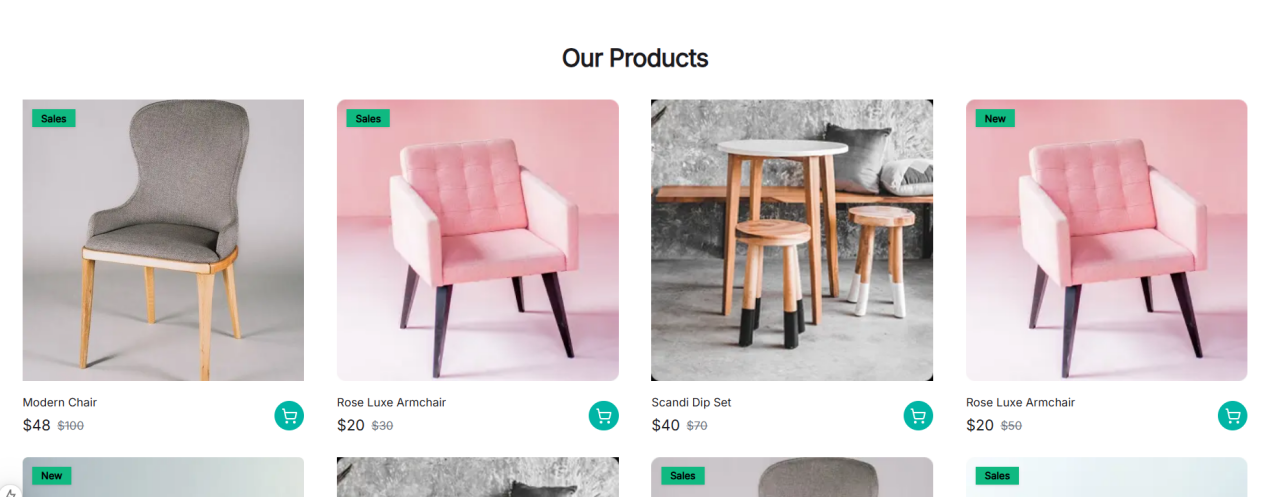
The API integration involved fetching product data from an external source and populating it in Sanity CMS. The following steps were followed:

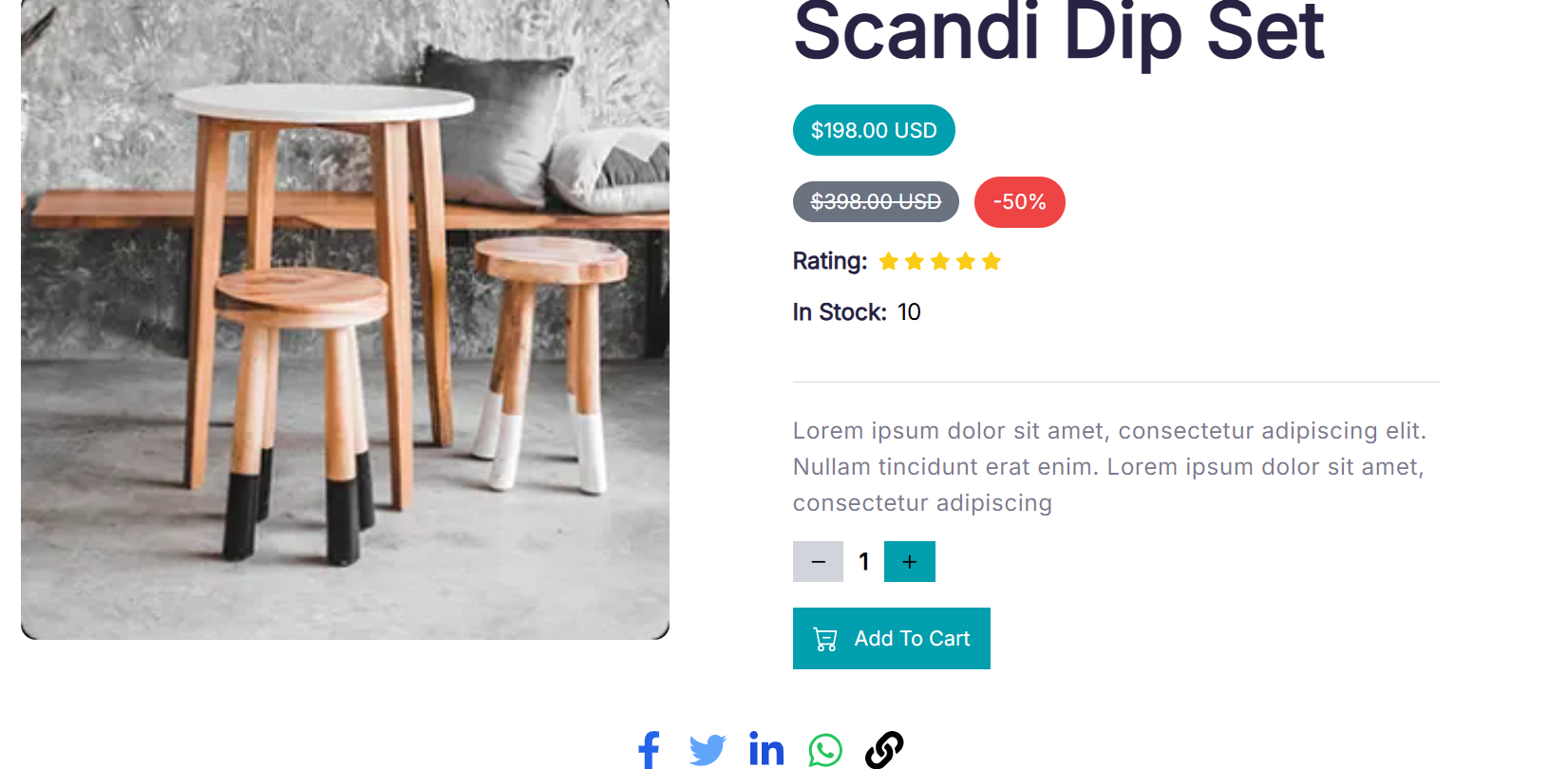
1. **Identifying the API Endpoints**:
   1. The external API that provides product data was identified, including endpoints for product information, images, and categories.
2. **Creating API Fetch Functions**:
   1. We created a fetch function using fetch() or an alternative like axios to retrieve product data from the external API.
3. **Integrating API Calls in the Project**:
   1. API calls were made during the data fetching process in Next.js. The API was connected to the frontend and used in pages like Product Detail.



**Frontend Display of API Data**:

* Once the data is fetched, it is displayed in the frontend. For example, on the Product Detail page, we display the product details like title, price, description, and inventory status fetched from the API.





· **Handling Data**:

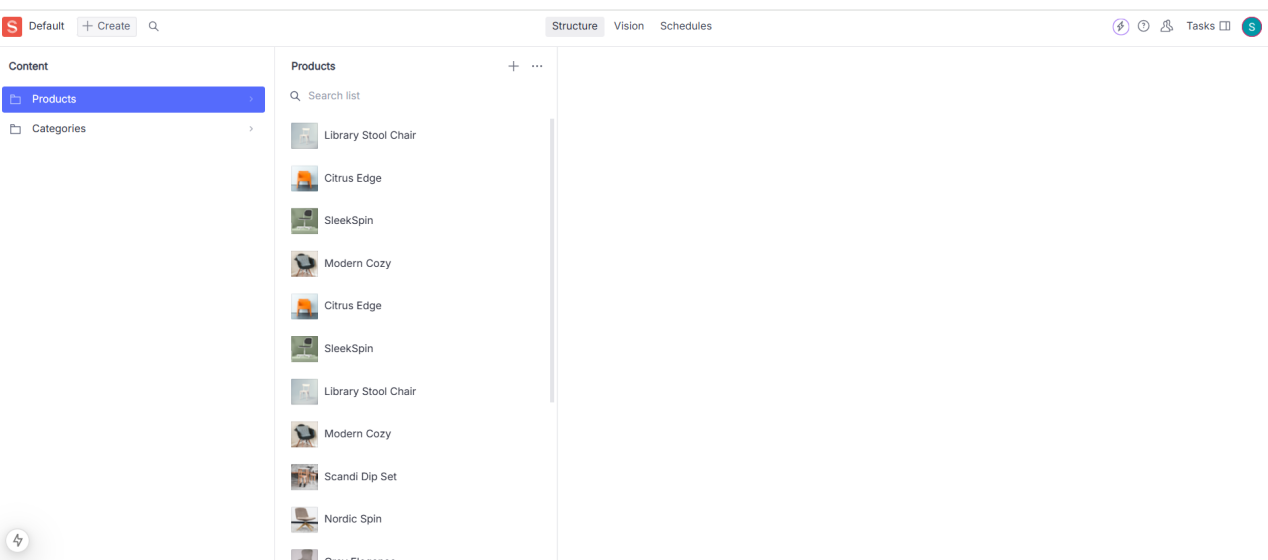
· Upon successful API calls, product data was parsed and displayed in the frontend. The product details were rendered on the Product Detail page.

· **Error Handling**:

· A fallback mechanism was implemented to handle any errors during the API fetch process (e.g., when the product isn't found).



### Adjustments Made to Schemas

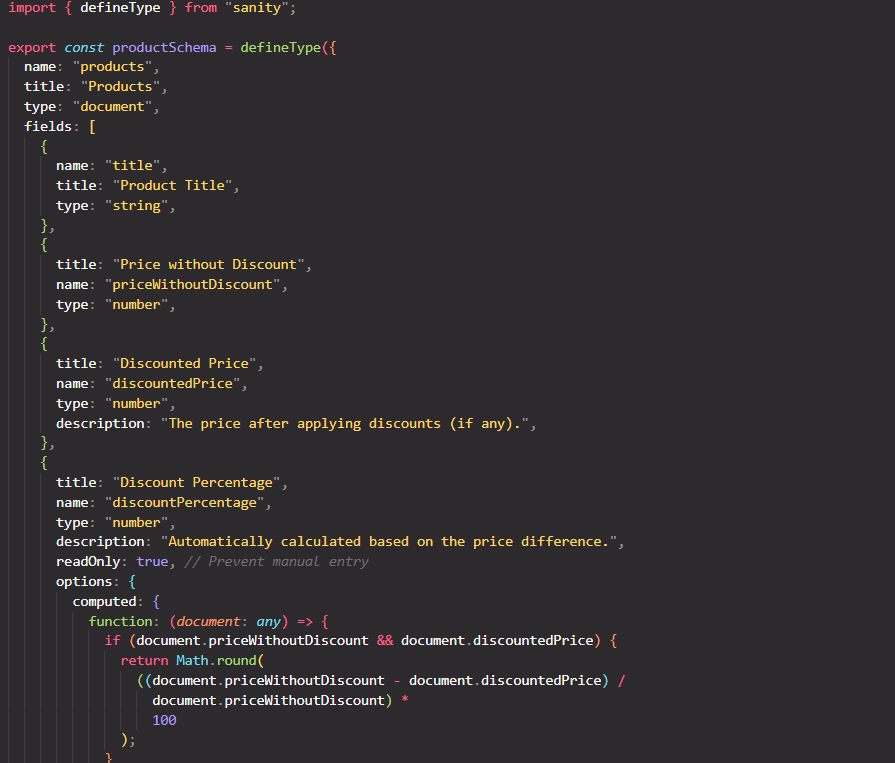


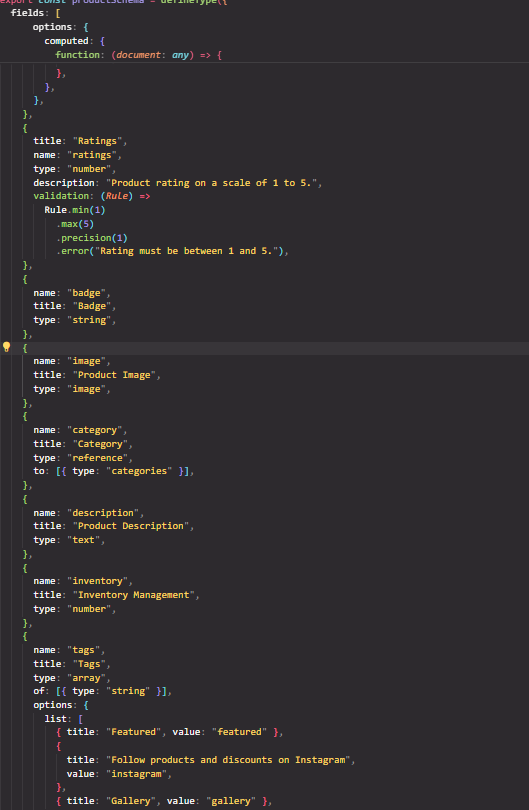
#### 2. ****Schema Adjustments****

To integrate and store data effectively, the following adjustments were made to the Sanity CMS schemas:

1. **Product Schema**:

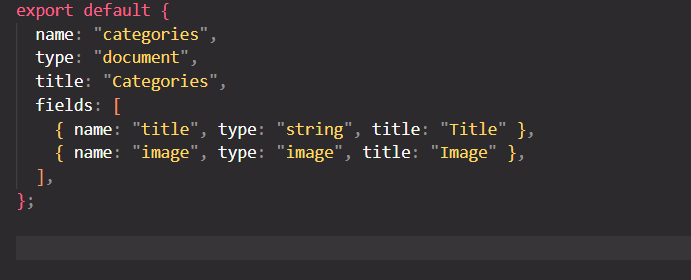
New fields were added to the product schema to handle API data, such as priceWithoutDiscount, tags, and rating.





1. **Category Schema**:

A category reference was added in the product schema to link products to specific categories.



**3.Image Handling**:

* The product images fetched via the API were connected with Sanity’s image asset reference field to ensure proper handling of image data.

### Migration Steps and Tools Used

#### 3. ****Data Migration****

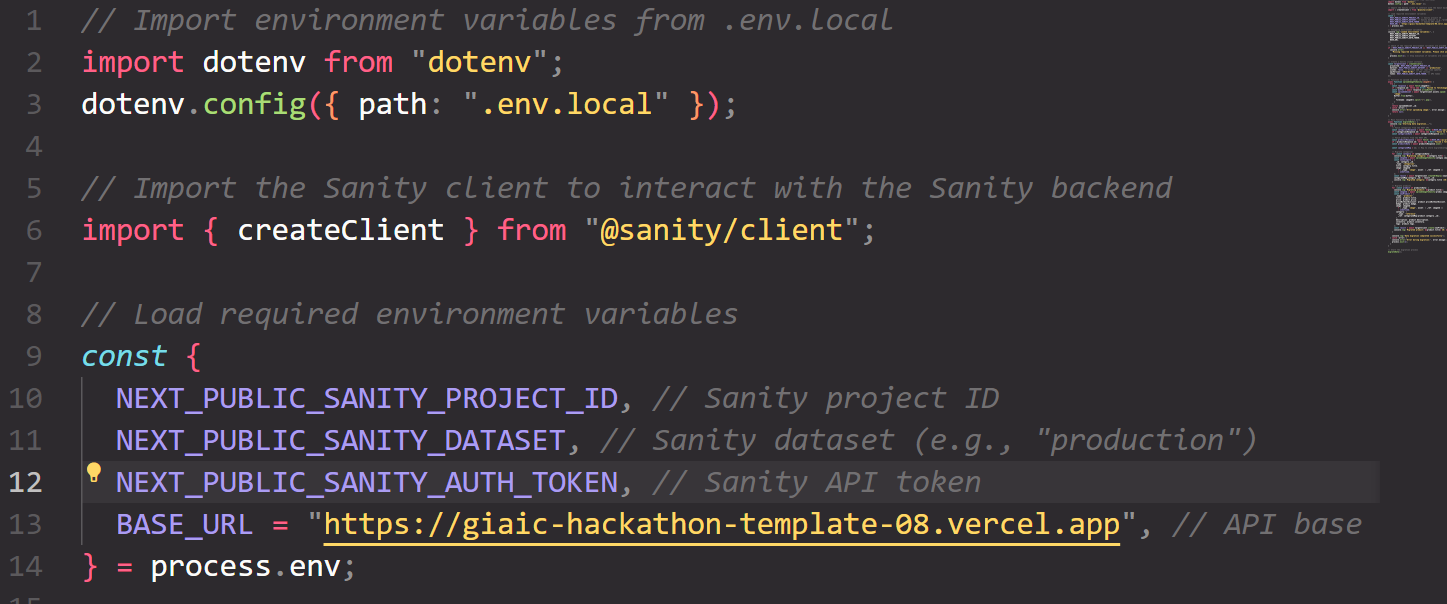
The product data was migrated from the external API into the Sanity CMS using the following steps:

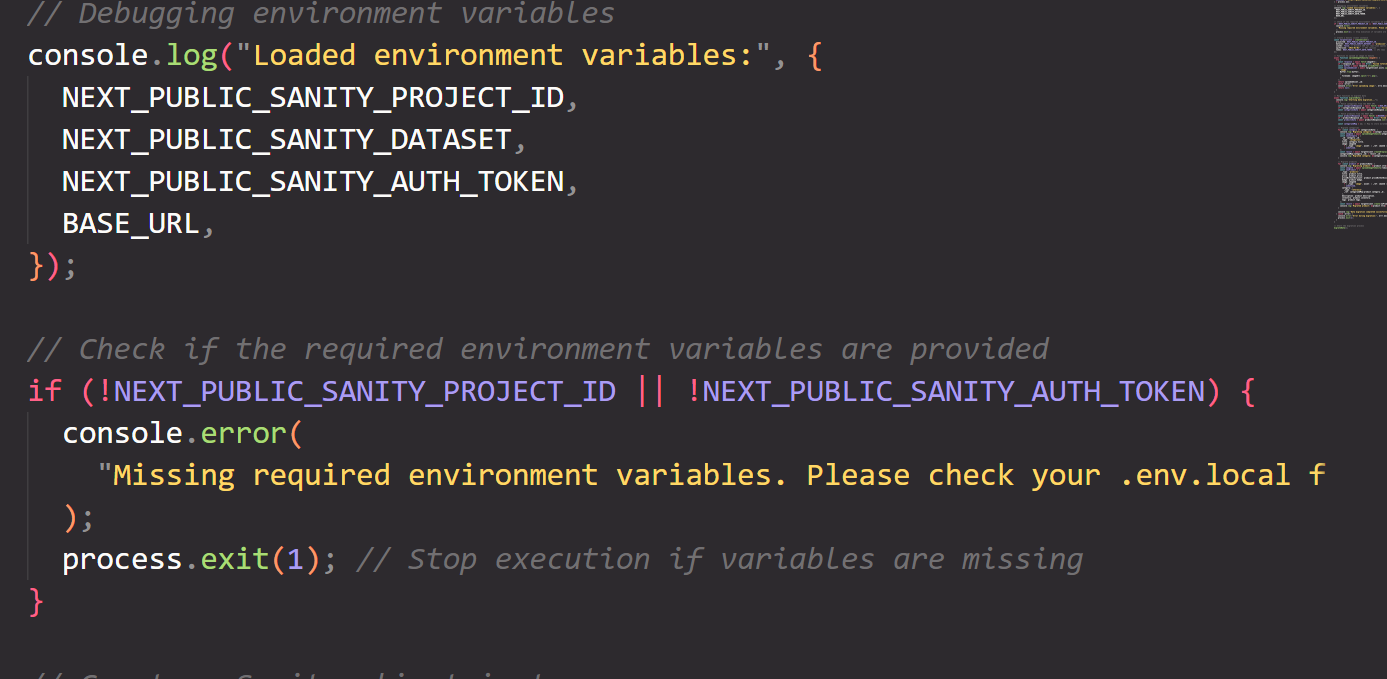
**Preparing the Migration Script**:

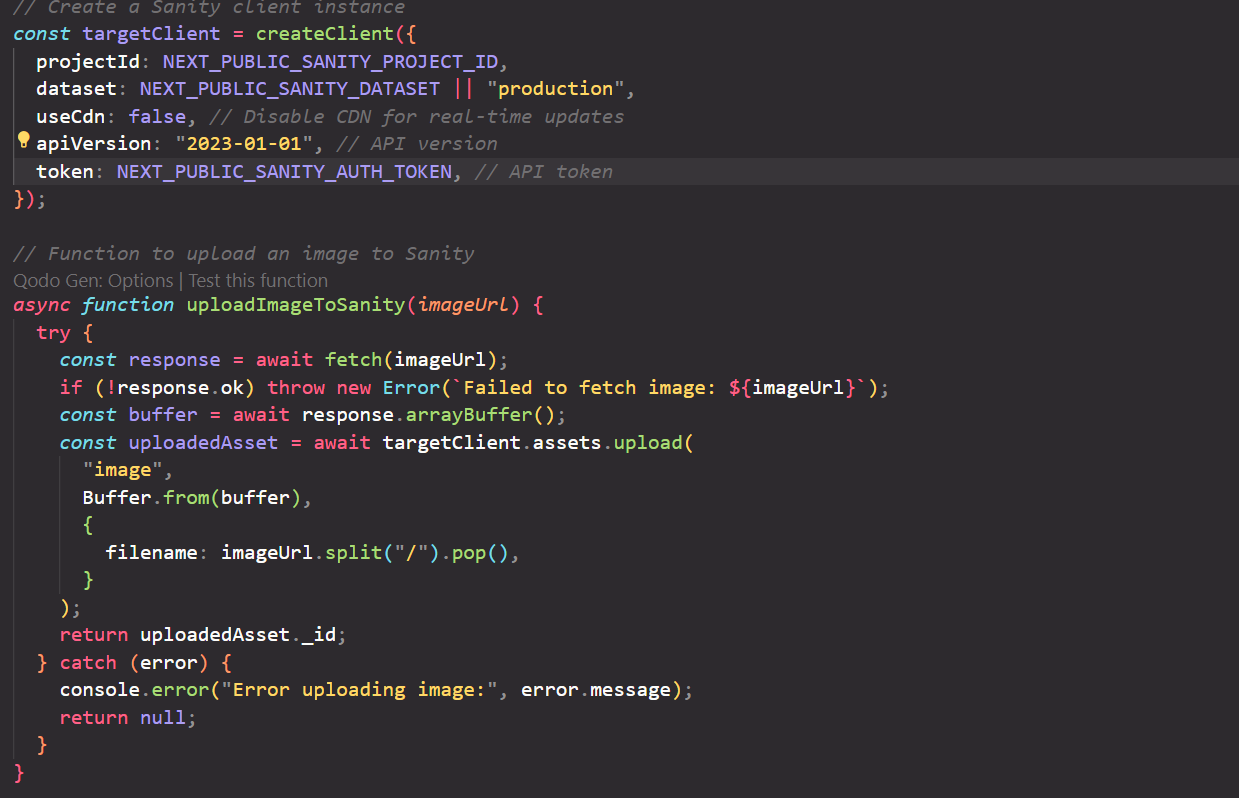
* 1. A Node.js script was written to automate the migration process. This script used Sanity’s client to create documents in the CMS with data from the external API.

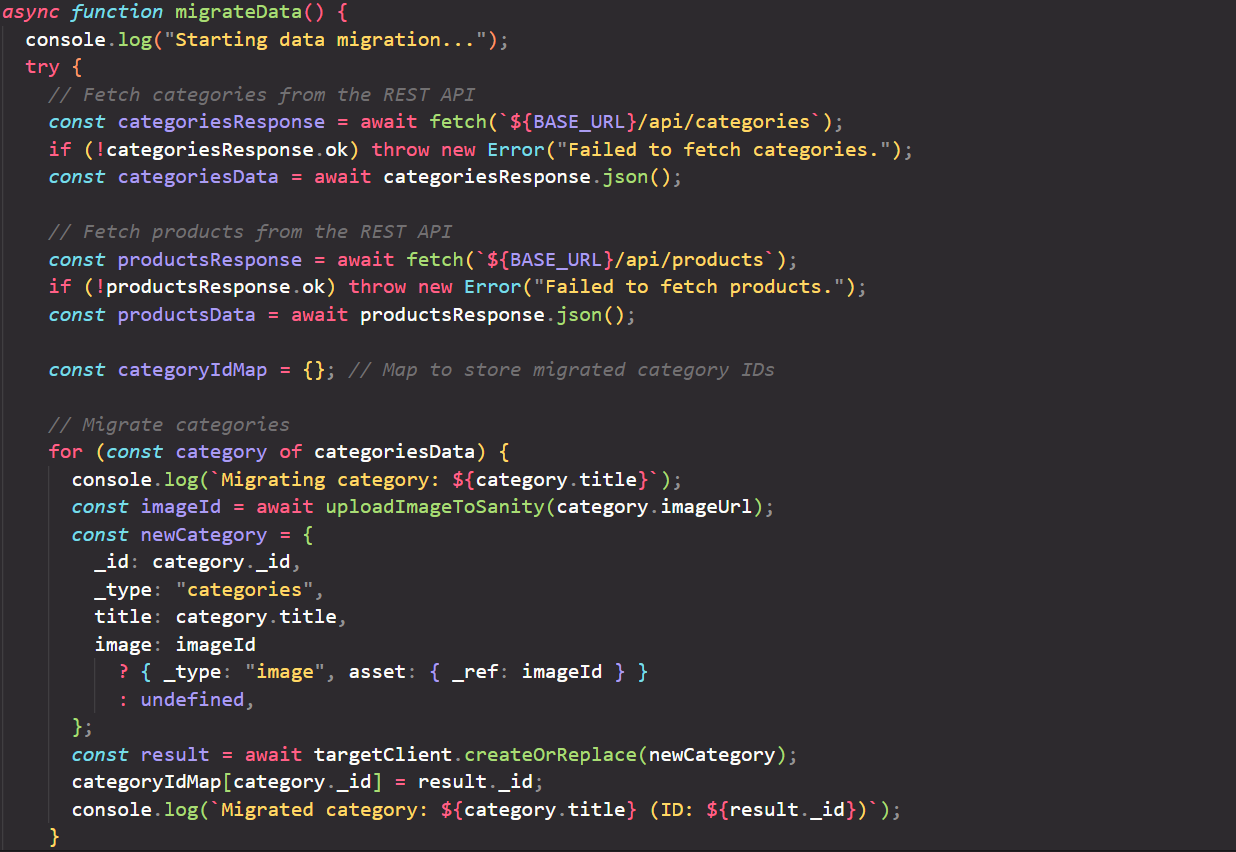
**Script to Migrate Product Data**:

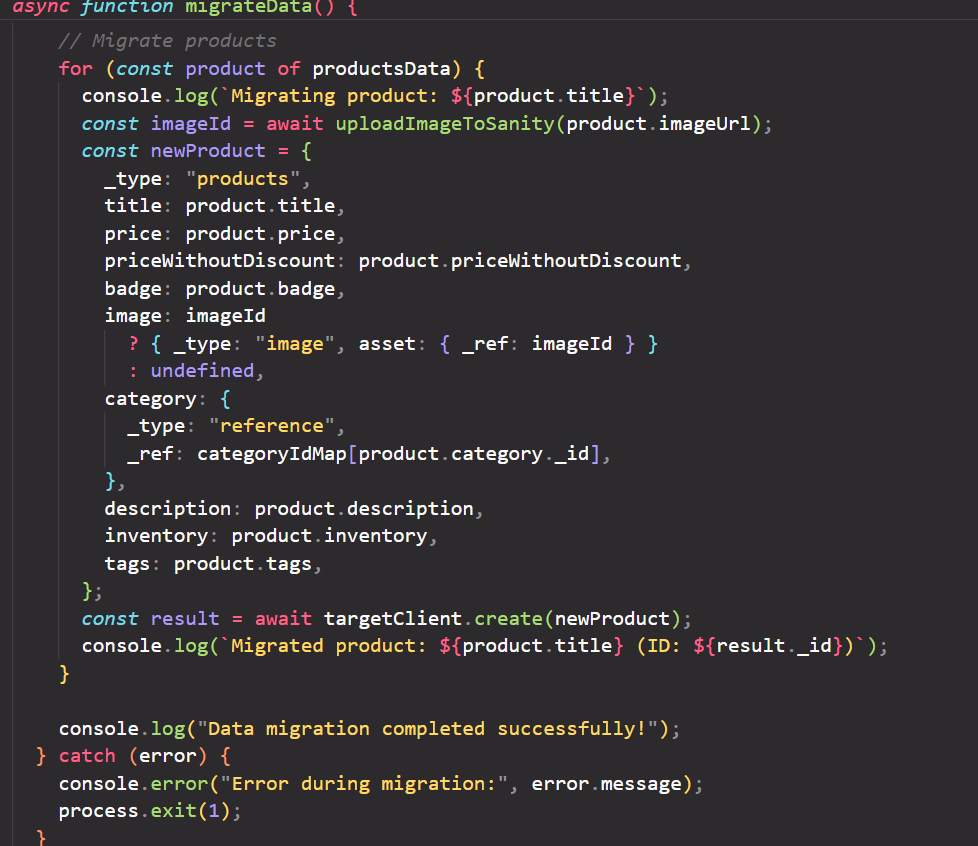
* 1. The script iterated over the product data and pushed it to Sanity’s CMS using the following code:

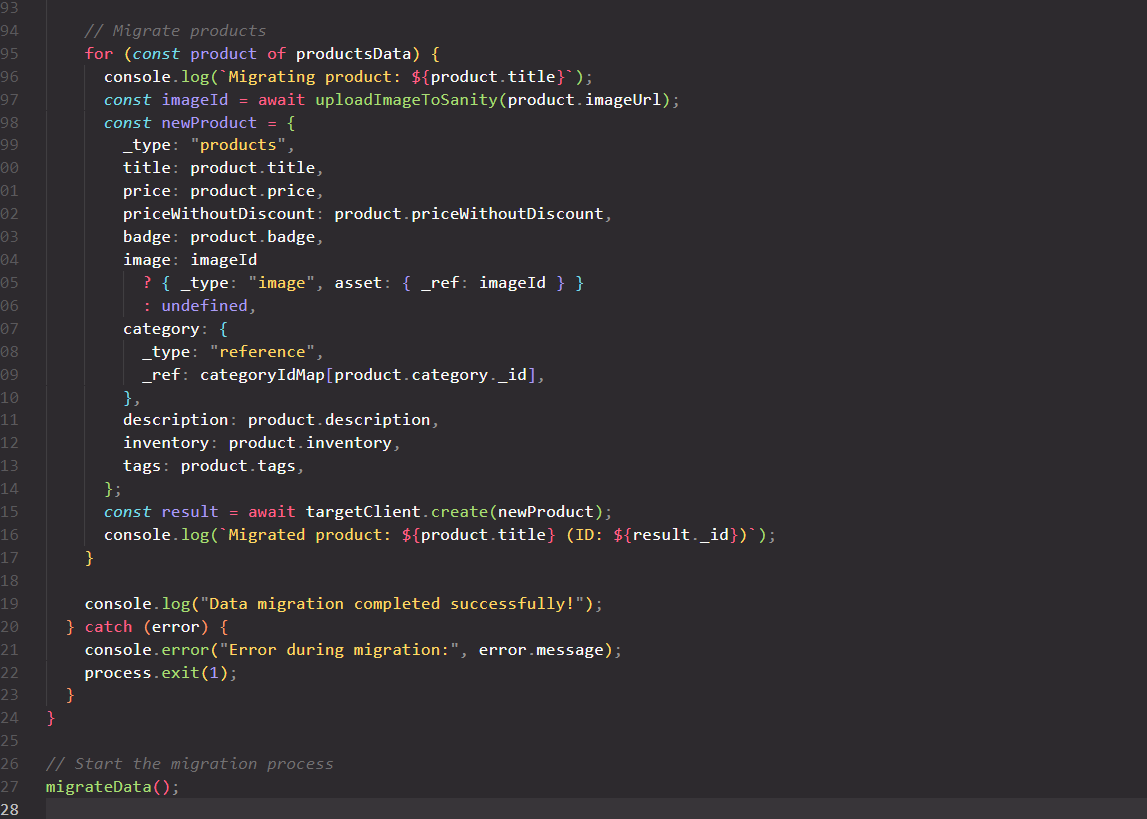












**Sanity Studio**:

* 1. Sanity Studio was used to verify that the data was correctly migrated and populated. We could check the product documents and their fields to ensure that all relevant information was stored.

**Verification**:

* 1. After migration, the data was verified in the Sanity Studio interface, ensuring that all product data, including pricing, description, and categories, were correctly added.

### Screenshots

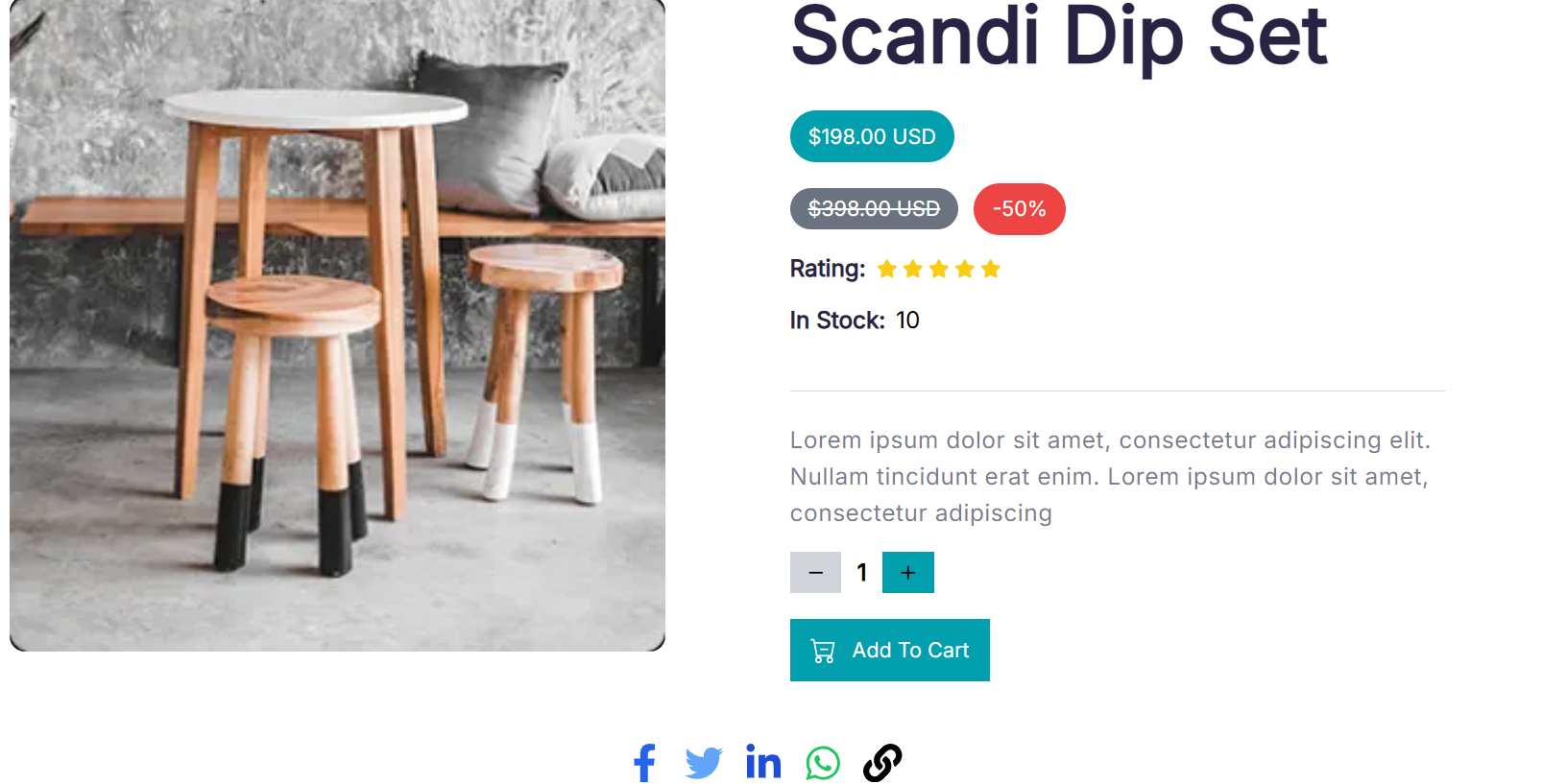
**API Calls**:

* 1. Screenshots of successful API calls made to fetch product data.



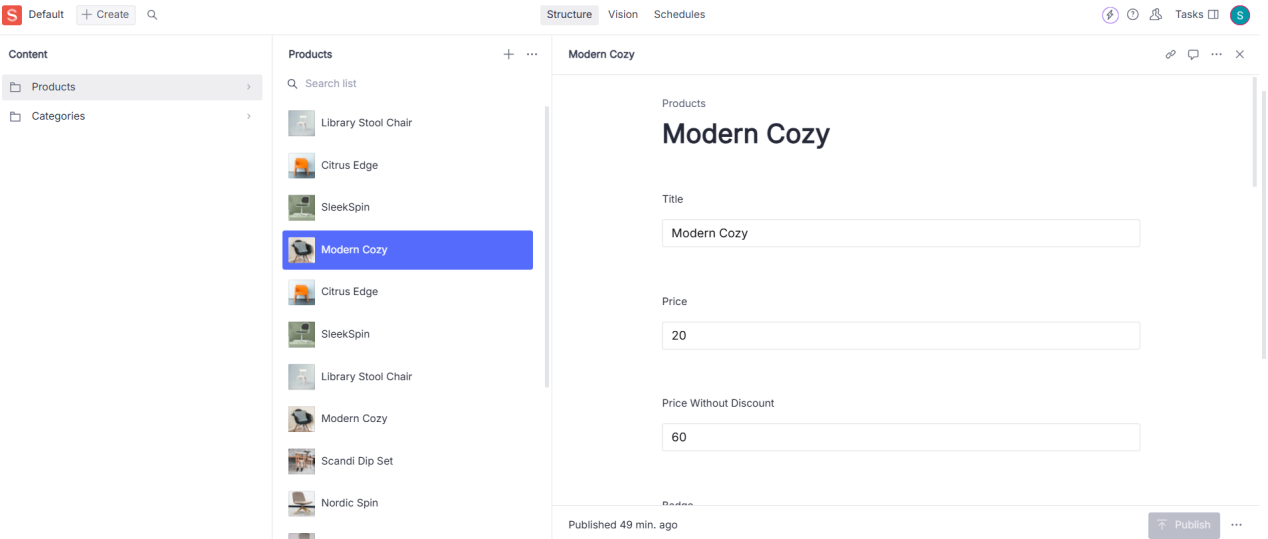
**Frontend Display**:

* 1. A screenshot showing the data displayed on the frontend (e.g., product details page with information from the API).



**Populated Sanity CMS Fields**:

* 1. A screenshot of Sanity Studio showing the product document with populated fields, such as price, description, and tags.



### Conclusion

The API integration process was successfully completed, with the product data being fetched, stored, and displayed on the frontend. Necessary adjustments were made to the Sanity CMS schema to accommodate the new product information. Data migration was automated using a custom script, ensuring a smooth transition from the external API to the CMS.