# Day-3 API Integration Report

**Introduction**

**Project Overview:** In this project, I have successfully integrated an e-commerce API with a frontend application, leveraging Sanity CMS to streamline clothing product management. By dynamically uploading product details, including images, via the API, I've ensured seamless storage and accessibility within Sanity. This integration optimizes data handling and powers a responsive and intuitive frontend, complete with advanced filtering, sorting, and categorization capabilities to enhance the user experience.

**API Integration**

**API Used:** The information is sourced from a specialized API endpoint designed to deliver detailed data on a wide range of products. The API URL is as follows:

- API URL

This API offers comprehensive details, including:

- Product ID
- Product Name
- Product Description
- Product Price
- Product Image
- Product Category
- Product Discount Percentage
- Product New or Old
- Product Colors
- Product Sizes

**Code for Fetch:** Here's the custom code used to fetch product data and upload images to Sanity. You can place this code in importData.js:

javascript

```
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();
```

**GROQ Query**

In this project, I have used the following GROQ query to retrieve clothing product data from Sanity. The query fetches details like product name, price, description, and image URL:

javascript

```
*[_type=="products"]{
 _id,
 name,
```

```
 description,
 price,
 "imageUrl": image.asset->url,
 category,
 discountPercent,
 "isNew": new,
 colors,
 sizes
}
```

**Schema Definition**

Below is the schema definition used for the products document in Sanity:

javascript

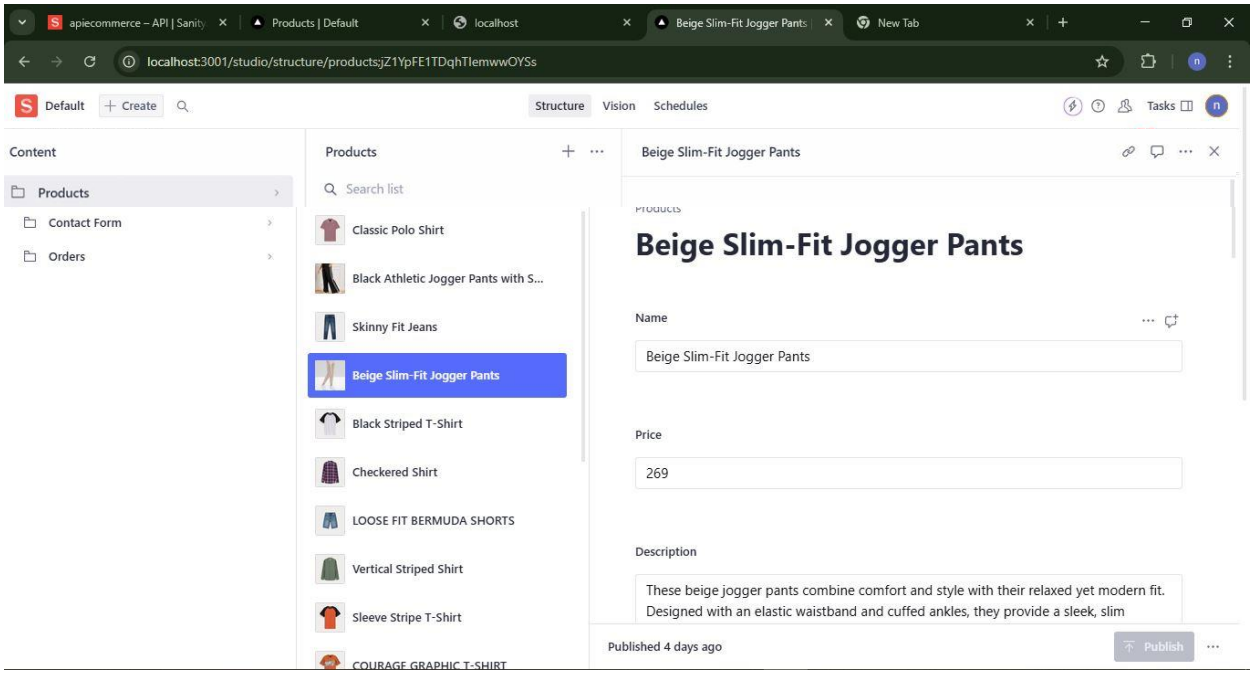
```
import { defineType } from "sanity"

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' }]
  }
]
```

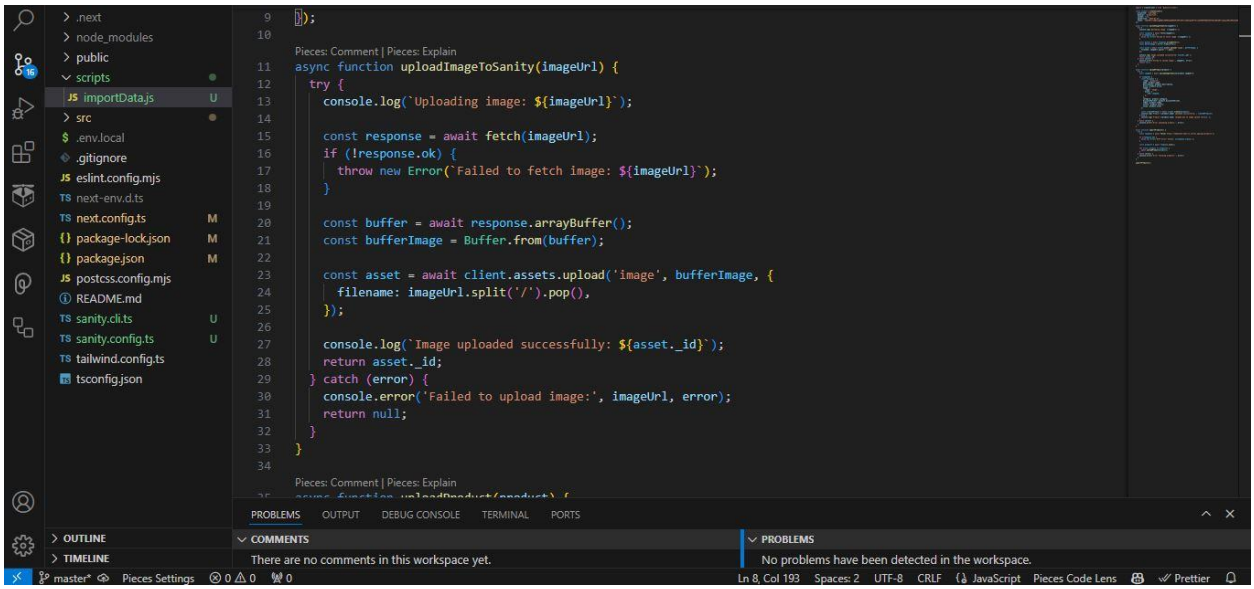
})

Sanity Studio interface

Once the data is imported, your Sanity Studio interface will look like this:

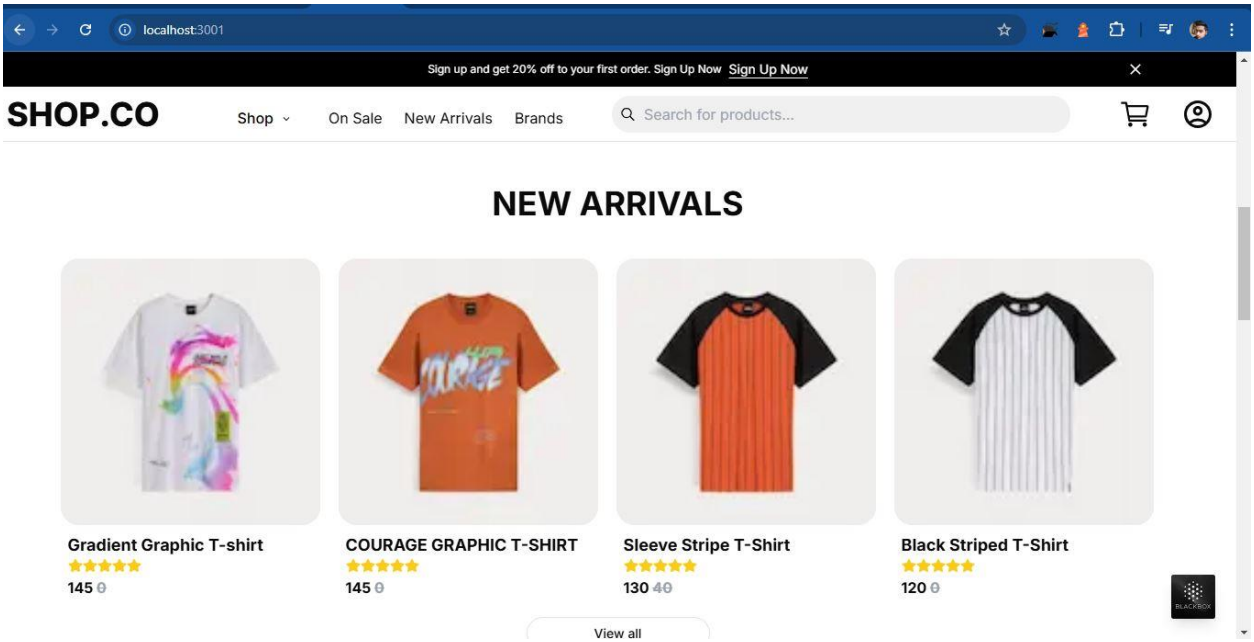



Code for Fetch: Here’s the custom code used to fetch product data and upload images to Sanity. You can place this code importData.j



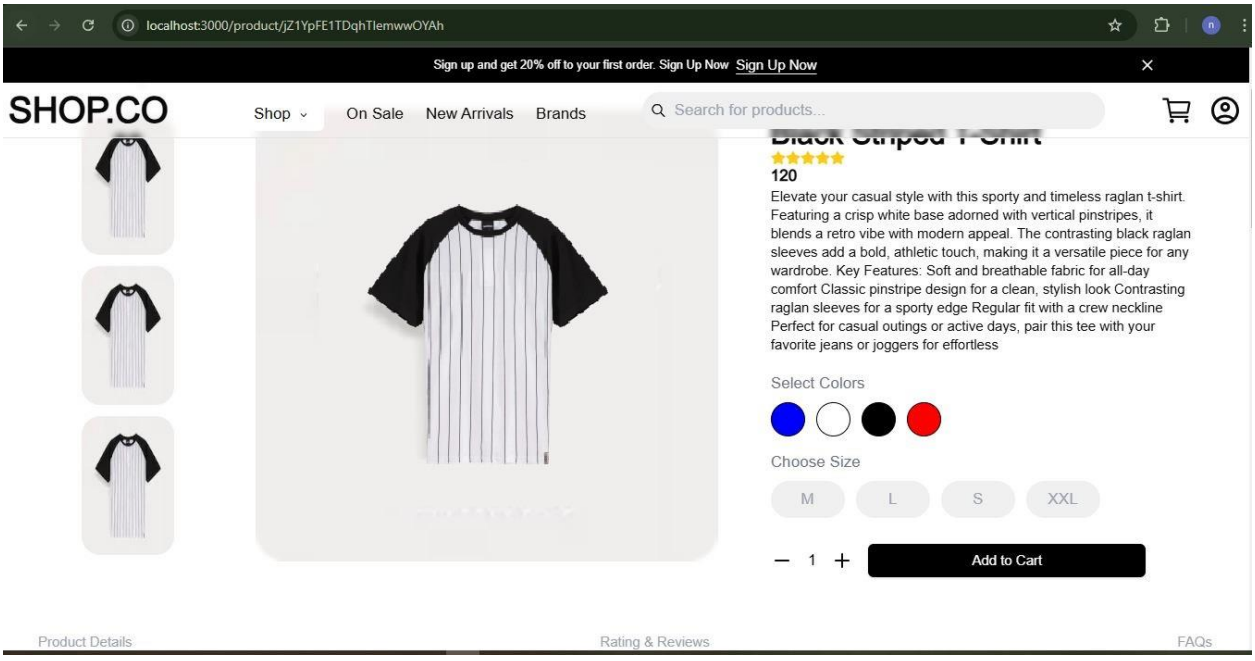
### Frontend Page

The frontend page showcases the clothing product details fetched from Sanity. Below is an example of how the page appears, displaying images, product title, price, and discount.




### Dynamic Page

The dynamic page further showcases additional details like colours and sizes.
If you need further modifications or have specific requirements, please let me know!



**Conclusion**

The integration of the e-commerce API with the frontend application through Sanity has been successfully implemented. The API dynamically retrieves clothing product details and images, while Sanity acts as the backend for storing and managing this data. This setup streamlines the handling of product information and elevates the user experience by providing an intuitive, flexible, and engaging frontend for seamless browsing and shopping.