INTRODUCTION

This report provides an overview of the process of incorporating an external API into a marketplace project. The primary goal was to retrieve product data, including images, from the API and store it within Sanity CMS. It details the modifications made to Sanity's schema, the data migration process, and the tools utilized during implementation.

1. API Integration Process

I integrated the API for Template 6 to fetch the required data. The process involved the following steps:

Identified the endpoints provided by the API for fetching product and inventory data.

Set up API calls in the Next.js application using fetch() to retrieve data dynamically.

Ensured proper error handling and logging to manage failed requests

2. Adjustments Made to Schemas
To align the fetched API data with Sanity CMS:

Updated the Product schema to include additional fields such as API_ID, images, and price.

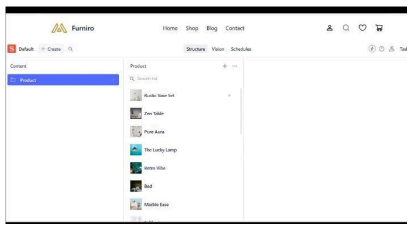
Modified the Category schema to support new relationships between products and their categories.

3. Migration Steps and Tools Used
For importing API data into Sanity CMS:

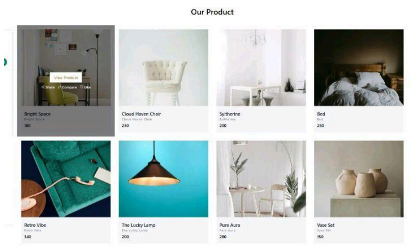
Used Sanity's import tool and custom scripts written in Node.js to transform API data into the required schema format.

Ensured unique _id for all imported data to prevent duplication.
Validated data consistency after import.

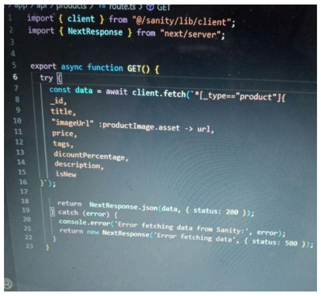
4. Screenshots
API Call Response: Screenshot of successful API call in the browser console or Postman.



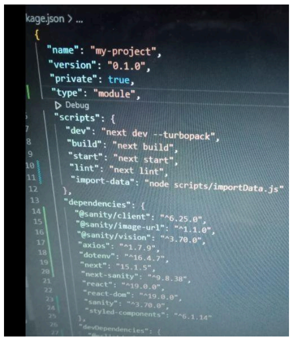
Frontend Display: Screenshot showing the data displayed on the product listing page in the frontend.



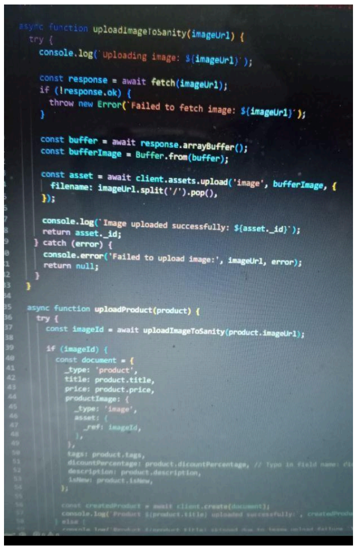
Coding screenshot:



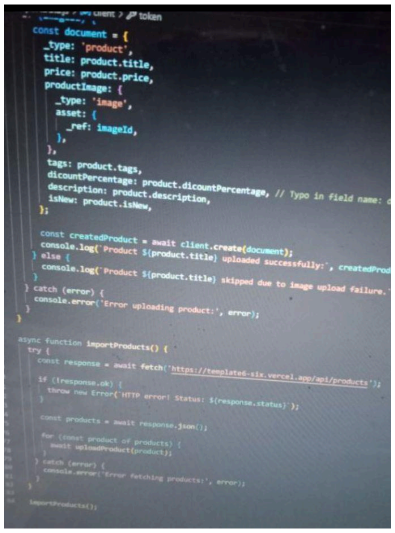
```
import { client } from "@/sanity/lib/client";
import { NextResponse } from "next/server";


export async function GET() {
  try {
    const data = await client.fetch(`*[_type=="product"]{
    _id,
    title,
    "imageUrl" :productImage.asset -> url,
    price,
    tags,
    dicountPercentage,
    description,
    isNew
  }`);

    return  NextResponse.json(data, { status: 200 });
  } catch (error) {
    console.error('Error fetching data from Sanity:', error);
    return new NextResponse('Error fetching data', { status: 500 });
  }
}
```



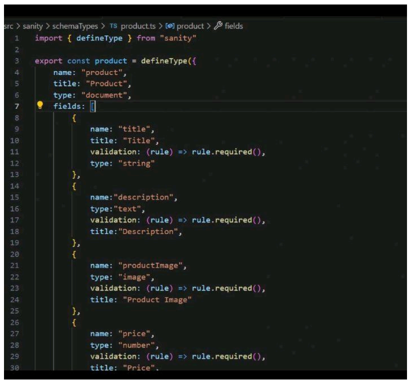
```
{
  "name": "my-project",
  "version": "0.1.0",
  "private": true,
  "type": "module",
  "scripts": {
    "dev": "next dev --turbopack",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data": "node scripts/importData.js"
  },
  "dependencies": {
    "@sanity/client": "^6.25.0",
    "@sanity/image-url": "^1.1.0",
    "@sanity/vision": "^3.70.0",
    "axios": "^1.7.9",
    "dotenv": "^16.4.7",
    "next": "15.1.5",
    "next-sanity": "^9.8.38",
    "react": "^19.0.0",
    "react-dom": "^19.0.0",
    "sanity": "^3.70.0",
    "styled-components": "^6.1.14"
  },
  "devDependencies": {
```



```
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: 'product',
        title: product.title,
        price: product.price,
        productImage: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        tags: product.tags,
        dicountPercentage: product.dicountPercentage, // Typo in field name:
        description: product.description,
        isNew: product.isNew,
      };

      const createdProduct = await client.create(document);
      console.log(`Product ${product.title} uploaded successfully:`, createdProdu
    } else {
```



```
client > token
      const document = {
        _type: 'product',
        title: product.title,
        price: product.price,
        productImage: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        tags: product.tags,
        dicountPercentage: product.dicountPercentage, // Typo in field name: d
        description: product.description,
        isNew: product.isNew,
      };

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

async function importProducts() {
  try {
    const response = await fetch('https://template6-six.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();
```



```
src > sanity > schemaTypes > product.ts > product > fields
import { defineType } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        {
            name: "title",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        },
        {
            name:"description",
            type:"text",
            validation: (rule) => rule.required(),
            title:"Description",
        },
        {
            name: "productImage",
            type: "image",
            validation: (rule) => rule.required(),
            title: "Product Image"
        },
        {
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
```

Conclusion:

The API integration for Template 6 was successfully completed, ensuring that the data fetched from the API is accurately displayed in both the frontend and the Sanity CMS backend. Necessary adjustments were made to the schemas to align with the API structure, and migration scripts were implemented to import data seamlessly into SanityDay 3 - API Integration Report