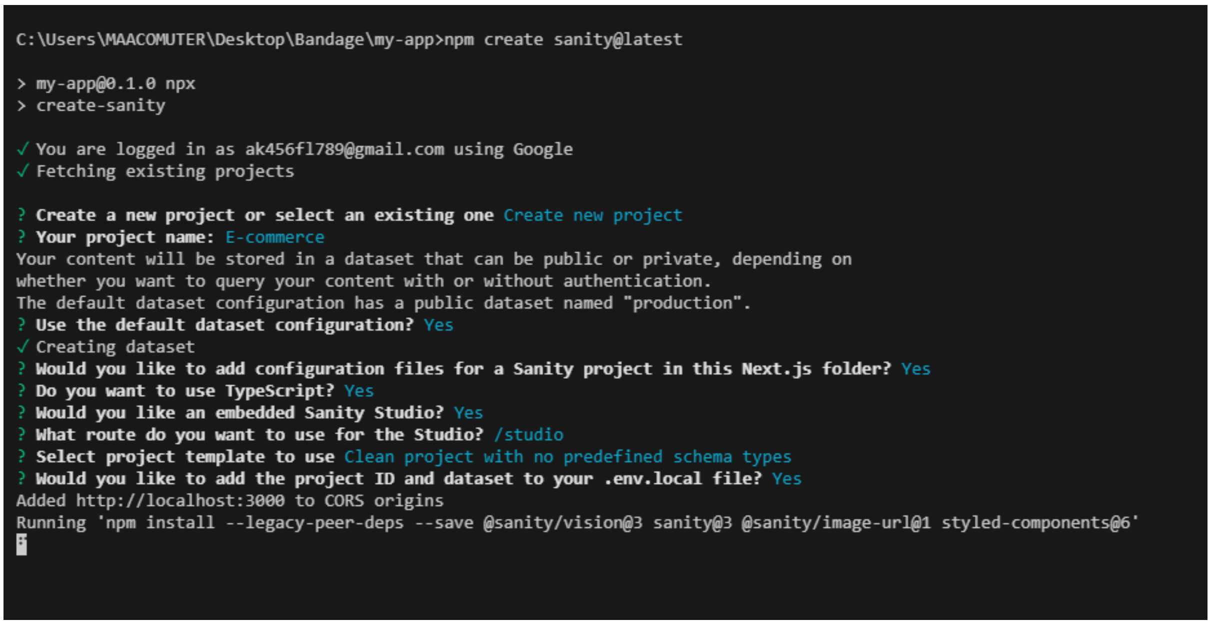
Name: Aafifa Afaq

Rollno:0007335

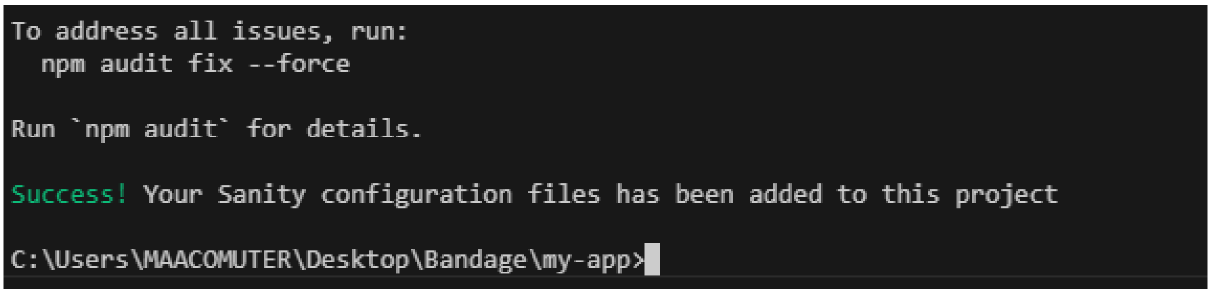
Day 3 - API Integration and Data Migration Report - [Bandage-E-Commerce Marketplace]

Overview

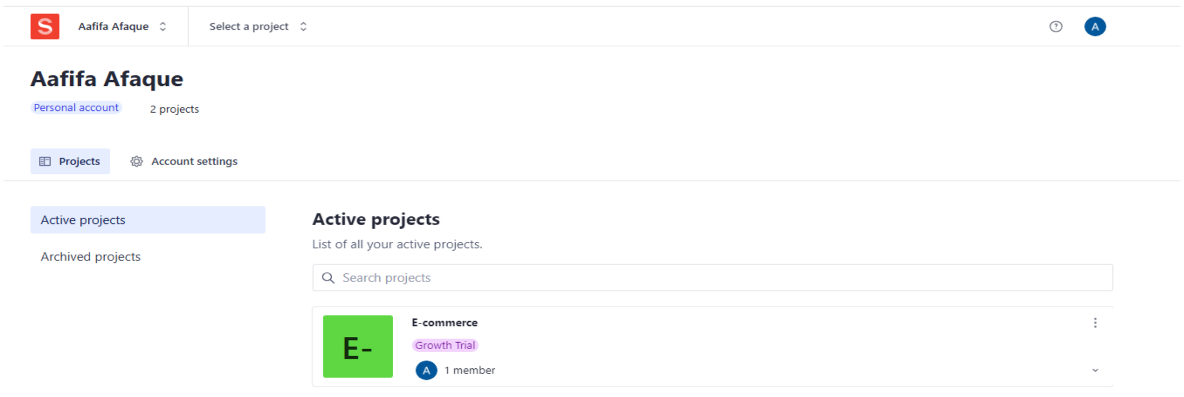
This document summarizes the process I followed to integrate APIs and migrate data into the Sanity CMS while ensuring a functional Next.js frontend. The objective was to populate the CMS with accurate data, align schemas, and build a robust frontend for data display.



```
C:\Users\MAACOMUTER\Desktop\Bandage\my-app>npm create sanity@latest

> my-app@0.1.0 npx
> create-sanity

✓ You are logged in as ak456fl789@gmail.com using Google
✓ Fetching existing projects

? Create a new project or select an existing one Create new project
? Your project name: E-commerce
Your content will be stored in a dataset that can be public or private, depending on
whether you want to query your content with or without authentication.
The default dataset configuration has a public dataset named "production".
? Use the default dataset configuration? Yes
✓ Creating dataset
? Would you like to add configuration files for a Sanity project in this Next.js folder? Yes
? Do you want to use TypeScript? Yes
? Would you like an embedded Sanity Studio? Yes
? What route do you want to use for the Studio? /studio
? Select project template to use Clean project with no predefined schema types
? Would you like to add the project ID and dataset to your .env.local file? Yes
Added http://localhost:3000 to CORS origins
Running 'npm install --legacy-peer-deps --save @sanity/vision@3 sanity@3 @sanity/image-url@1 styled-components@6'
```



```
To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.

Success! Your Sanity configuration files has been added to this project

C:\Users\MAACOMUTER\Desktop\Bandage\my-app>
```



Aafifa Afaque

Personal account · 2 projects

Projects · Account settings

Active projects · Archived projects

Active projects

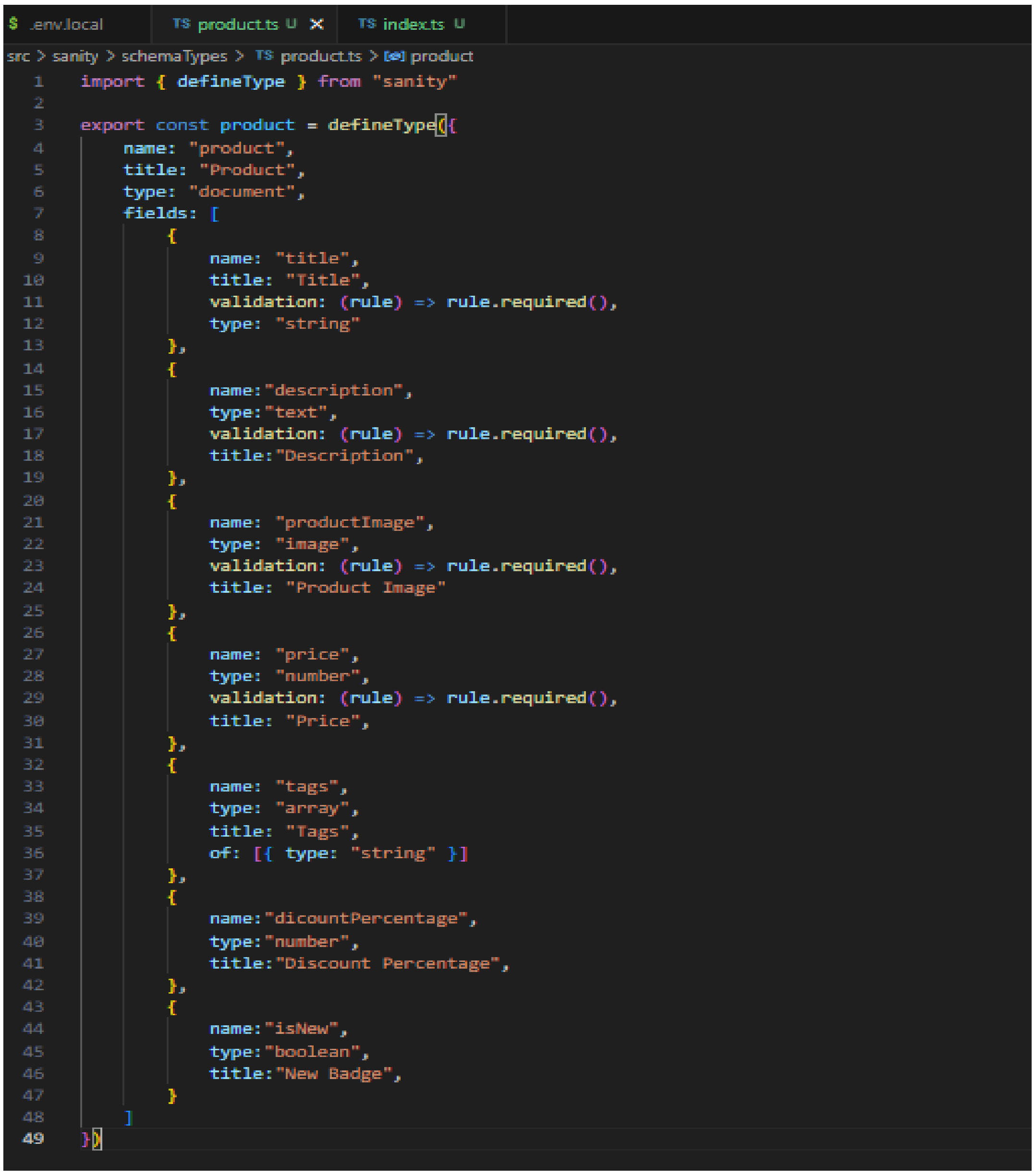
List of all your active projects.

Search projects

E-commerce · Growth Trial · 1 member

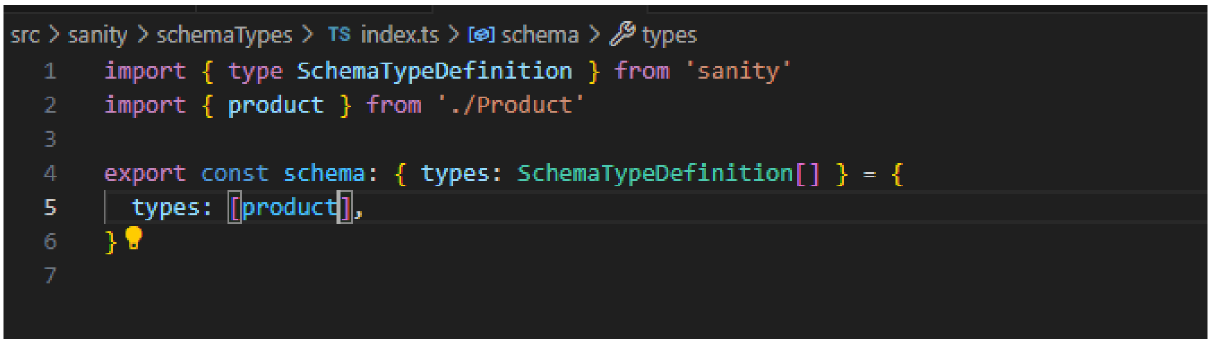
Process Overview

1. **Understanding the Provided API**
   - **API Documentation Review**:
     Identified key endpoints:
     - /products: For product listings.
   - **Tools Used**: browser developer tools to test endpoints and responses.
2. **Schema Validation and Adjustments**
   - Compared the Sanity CMS schema with the API data structure.
   - **Example Adjustments**:
     - **API Field**: slug
       Schema Field: slug



```ts
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",
        },
        {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }]
        },
        {
            name:"dicountPercentage",
            type:"number",
            title:"Discount Percentage",
        },
        {
            name:"isNew",
            type:"boolean",
            title:"New Badge",
        }
    ]
})
```

Action: Added slug field to the schema to align with the API data and ensure proper URL routing.

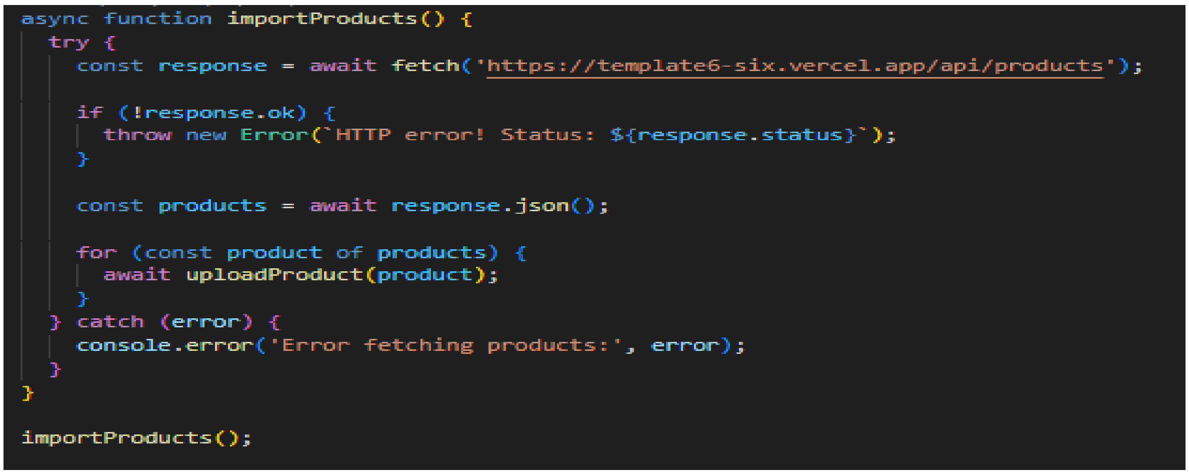


```
import { type SchemaTypeDefinition } from 'sanity'
import { product } from './Product'

export const schema: { types: SchemaTypeDefinition[] } = {
  types: [product],
}
```

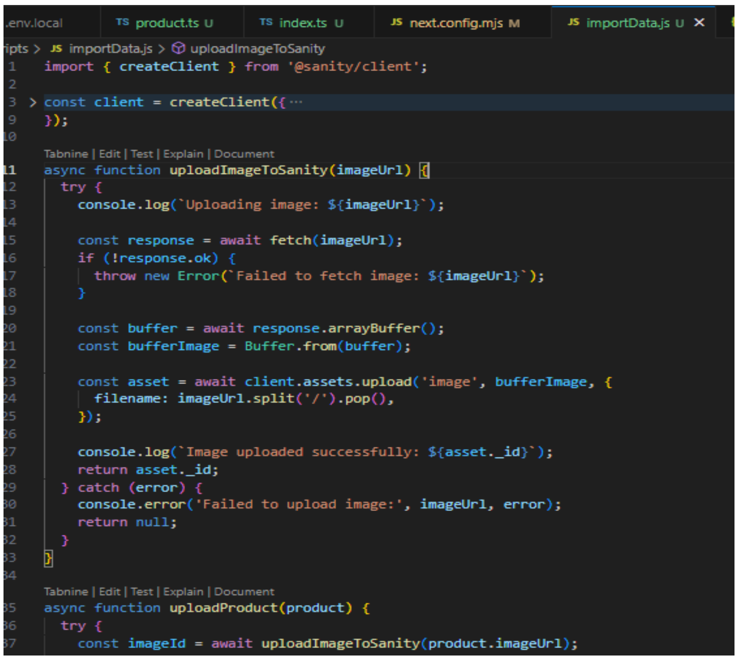
3. **Data Migration Methods**

   - **Provided API**:
     Wrote scripts to fetch, transform, and upload data into Sanity CMS.

4. **API Integration in Next.js**

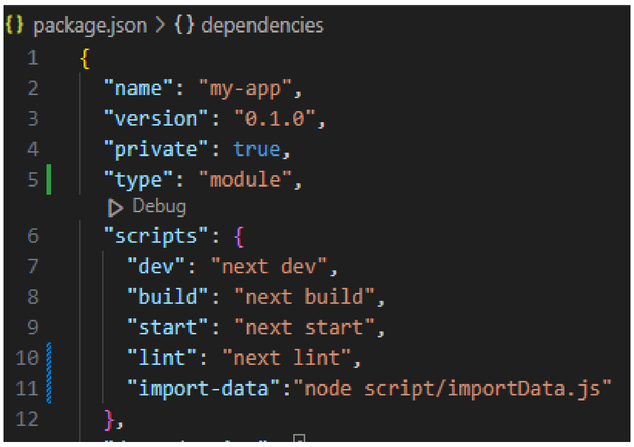


```
async function importProducts() {
  try {
    const response = await fetch('https://template6-six.vercel.app/api/products');

    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();
```



.env.local
product.ts U
index.ts U
next.config.mjs M
importData.js U
ipts > importData.js > uploadImageToSanity
Tabnine | Edit | Test | Explain | Document

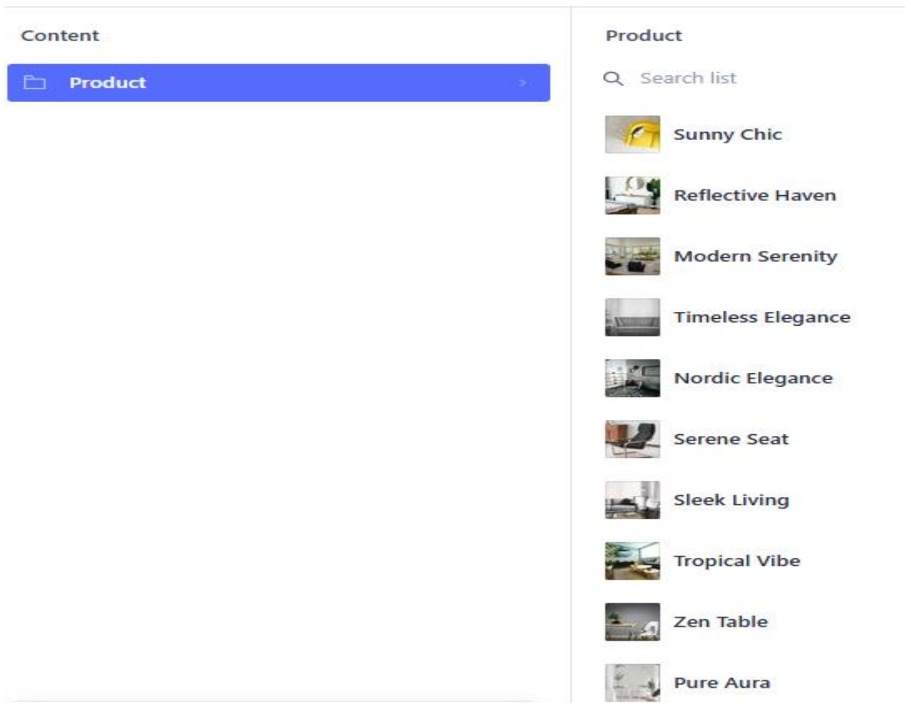
```
import { createClient } from '@sanity/client';

const client = createClient({…
});

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);
```

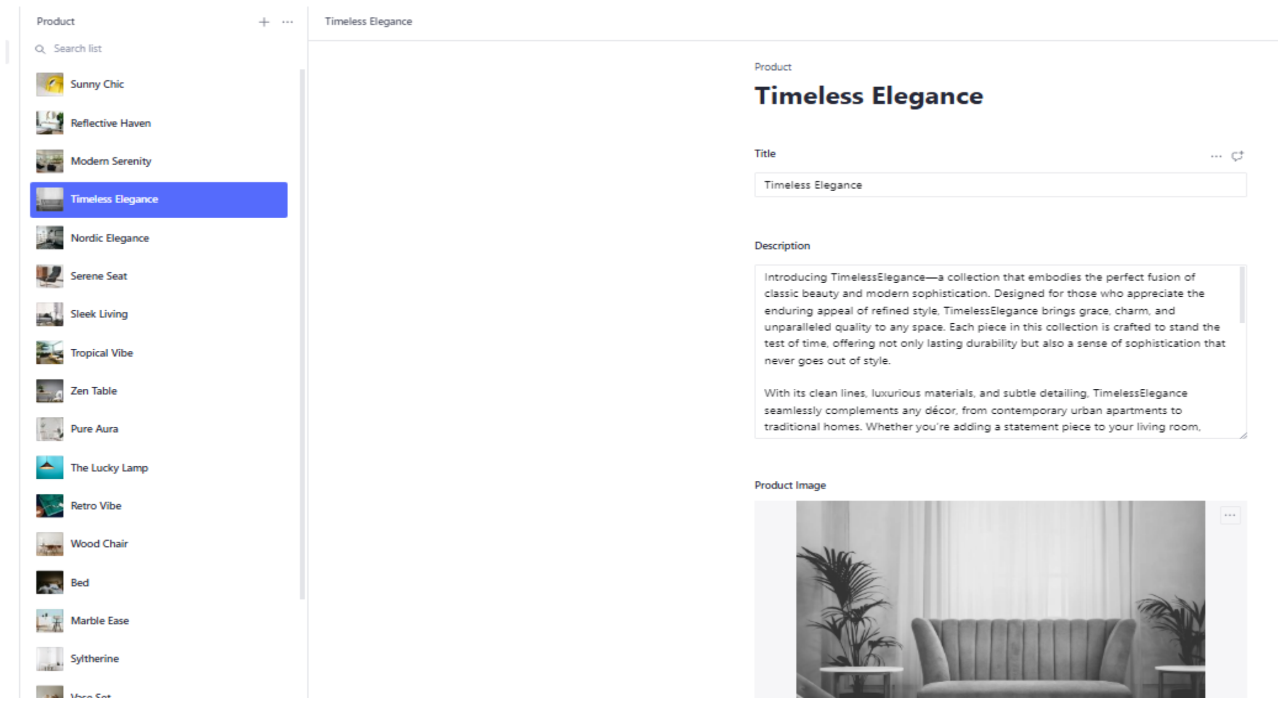


```
{
  "name": "my-app",
  "version": "0.1.0",
  "private": true,
  "type": "module",
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data":"node script/importData.js"
  },
```

Results

1. **Sanity CMS**: Successfully populated with data using automated and manual methods.
2. **Next.js Frontend**: Functional API integration displaying product listings and categories with fallback mechanisms.





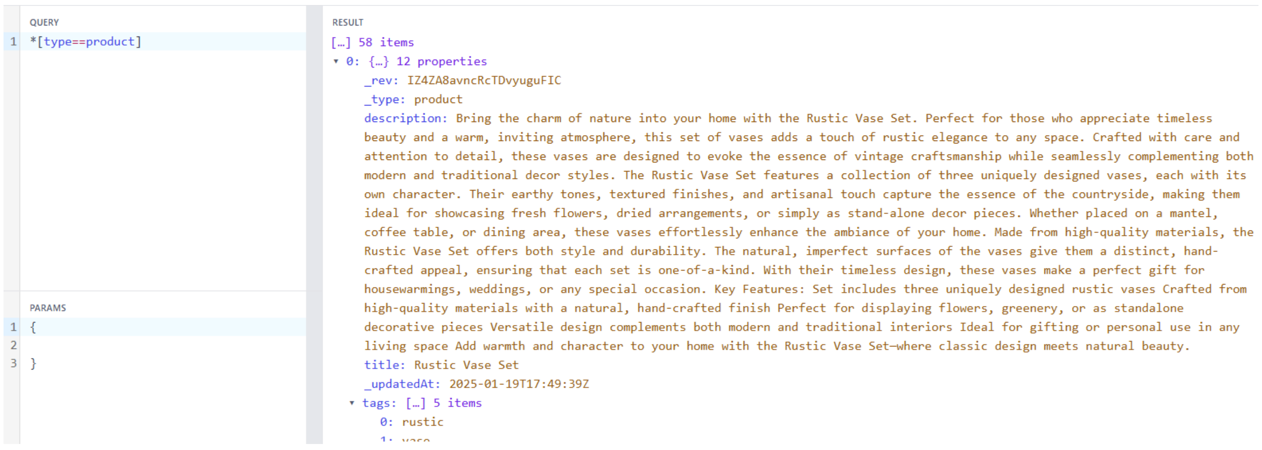
Timeless Elegance

Title: Timeless Elegance

Description:

Introducing TimelessElegance—a collection that embodies the perfect fusion of classic beauty and modern sophistication. Designed for those who appreciate the enduring appeal of refined style, TimelessElegance brings grace, charm, and unparalleled quality to any space. Each piece in this collection is crafted to stand the test of time, offering not only lasting durability but also a sense of sophistication that never goes out of style.

With its clean lines, luxurious materials, and subtle detailing, TimelessElegance seamlessly complements any décor, from contemporary urban apartments to traditional homes. Whether you're adding a statement piece to your living room,



QUERY

```
*[type==product]
```

PARAMS

```
{

}
```

RESULT

```
[…] 58 items
▾ 0: {…} 12 properties
    _rev: IZ4ZA8avncRcTDvyuguFIC
    _type: product
    description: Bring the charm of nature into your home with the Rustic Vase Set. Perfect for those who appreciate timeless
    beauty and a warm, inviting atmosphere, this set of vases adds a touch of rustic elegance to any space. Crafted with care and
    attention to detail, these vases are designed to evoke the essence of vintage craftsmanship while seamlessly complementing both
    modern and traditional decor styles. The Rustic Vase Set features a collection of three uniquely designed vases, each with its
    own character. Their earthy tones, textured finishes, and artisanal touch capture the essence of the countryside, making them
    ideal for showcasing fresh flowers, dried arrangements, or simply as stand-alone decor pieces. Whether placed on a mantel,
    coffee table, or dining area, these vases effortlessly enhance the ambiance of your home. Made from high-quality materials, the
    Rustic Vase Set offers both style and durability. The natural, imperfect surfaces of the vases give them a distinct, hand-
    crafted appeal, ensuring that each set is one-of-a-kind. With their timeless design, these vases make a perfect gift for
    housewarmings, weddings, or any special occasion. Key Features: Set includes three uniquely designed rustic vases Crafted from
    high-quality materials with a natural, hand-crafted finish Perfect for displaying flowers, greenery, or as standalone
    decorative pieces Versatile design complements both modern and traditional interiors Ideal for gifting or personal use in any
    living space Add warmth and character to your home with the Rustic Vase Set—where classic design meets natural beauty.
    title: Rustic Vase Set
    _updatedAt: 2025-01-19T17:49:39Z
  ▾ tags: […] 5 items
      0: rustic
      1: vase
```

Best Practices Followed

- **Sensitive Data Management**: Stored API keys in .env files.
- **Clean Code**: Used modular functions and descriptive variables with comments.
- **Data Validation**: Ensured data alignment with the schema before migration.
- **Version Control**: Committed frequently with detailed messages.
- **Thorough Testing**: Addressed edge cases and validated endpoints with Postman.

Conclusion

Through meticulous planning and execution, I ensured the Sanity CMS was accurately populated and fully integrated into a functional Next.js application. Robust practices and thorough testing were key to achieving a scalable and efficient