Marketplace Builder Hackathon

Day 3: API Integration and Data Migration

API Integration Report

1. **Introduction**
   This report details the process of integrating product and category data from an external API into the backend system of E-commerce, an online clothing store. The integration utilizes Sanity CMS for content management and Next.js for frontend rendering.

Key Objectives of the Integration:

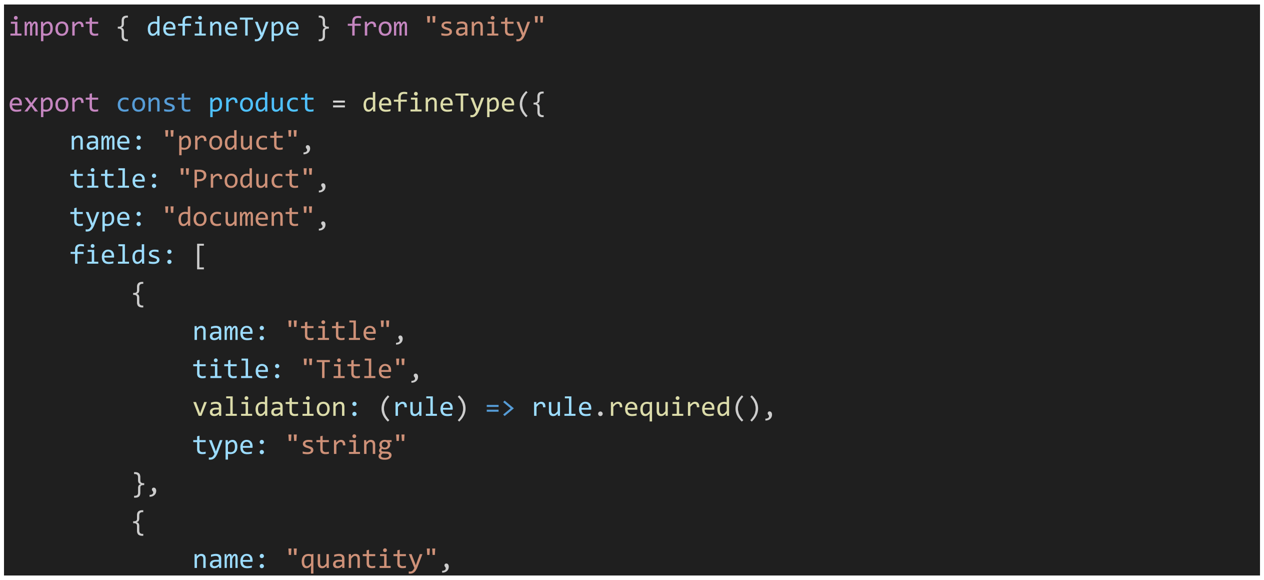
1. Effectively store and organize product and category data within **Sanity CMS**.
2. Fetch live data from an **external API.**
3. Display the retrieved data dynamically on the **frontend** to ensure a smooth shopping experience.

API Integration and Data Migration with Sanity and Schema

Step 1: Migrating Data to Sanity CMS

The fetched data was formatted and saved in **Sanity CMS** utilizing its **JavaScript client library.**

1.



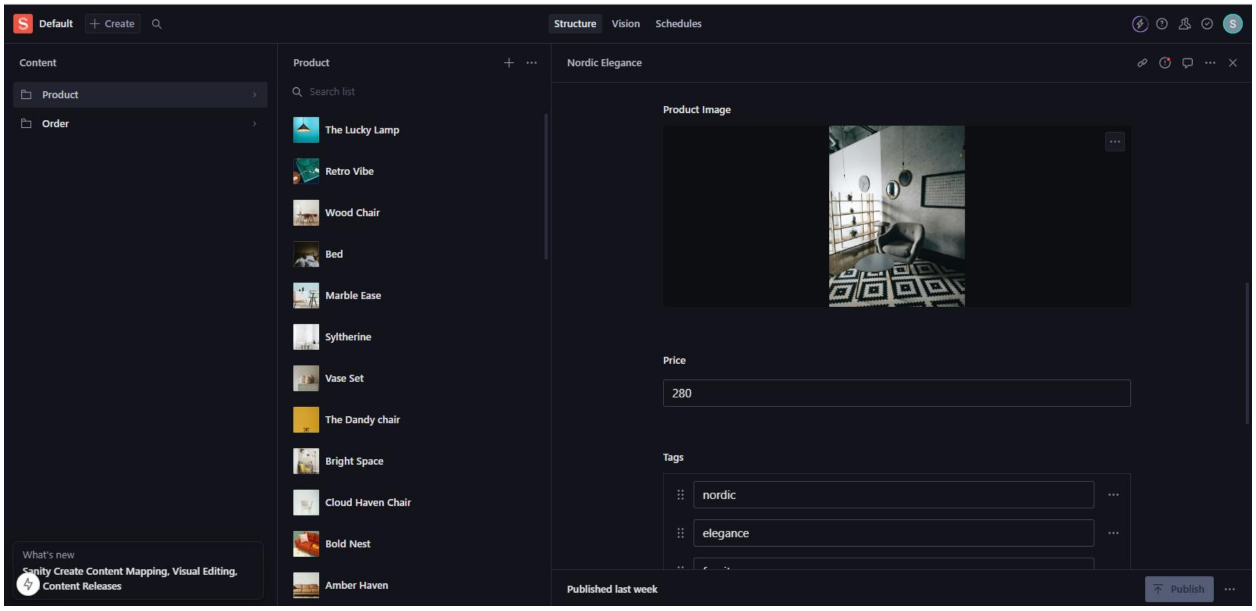
```
import { defineType } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        {
            name: "title",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        },
        {
            name: "quantity",
            title: "Quantity",
            validation: (rule) => rule.required(),
            type: "number"
        },
        {
            name: "inventory",
            title: "Inventory",
            validation: (rule) => rule.required(),
            type: "number"
        },
        {
            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",
        }
    ]
```

```
})
```

2.

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

const client = createClient({
  projectId: 'npvmw2bo',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-13',
  token:
'skYidbOtzgo7QG1TWptyxzWnJXR5PwM1BrxYVm3YCTwZ4pJsWr7cK6AKbcXRmEAMm5hxouIrU3dVHBB6
zSeK1I5IhvVuU2U0ssEidPvfBNPx0sdkvPLVCp1DjINSs8xvzedtV0IYohgayj5q71Hd53dpO0g9qgyVM
rqszzR5SnuhII3S2lml',
});

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,
        discountPercentage: product.discountPercentage, // Typo fixed
        description: product.description,
        isNew: product.isNew,
      };

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

Sanity Dashboard Screenshot Showing Stored Products



Product

- The Lucky Lamp
- Retro Vibe
- Wood Chair
- Bed
- Marble Ease
- Syltherine
- Vase Set
- The Dandy chair
- Bright Space
- Cloud Haven Chair
- Bold Nest
- Amber Haven

Nordic Elegance

Product Image

Price
280

Tags
nordic
elegance

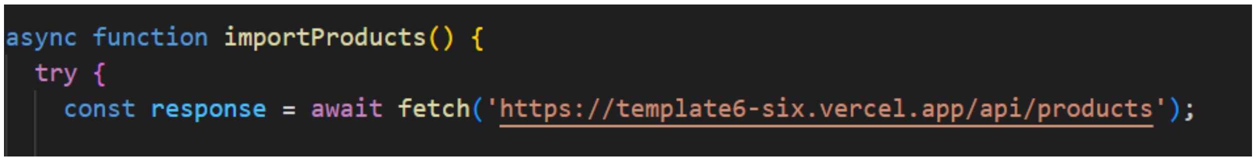
Step 2: Retrieving Data from the API

The API offers specific endpoints for accessing data, including:

- **Products Endpoint**: This endpoint delivers key product information, including:
  - Product name
  - Pricing details
  - Descriptions
  - Categories
  - Stock status
  - Product images

Base API URL:

https://template6-six.vercel.app/api/products

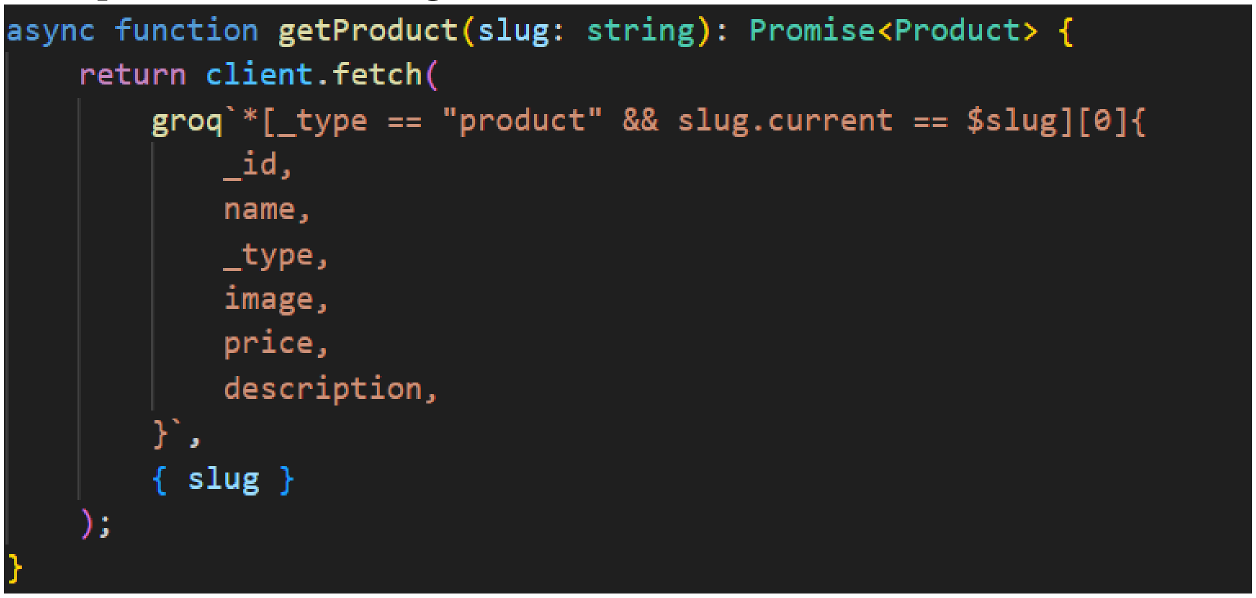


```
async function importProducts() {
  try {
    const response = await fetch('https://template6-six.vercel.app/api/products');
```

4. Frontend Integration

The stored product data was retrieved from **Sanity CMS** and showcased on the **E-Commerce** frontend. Utilizing **Next.js,** the data was dynamically rendered to populate the product listing interface, ensuring a seamless and interactive shopping experience.

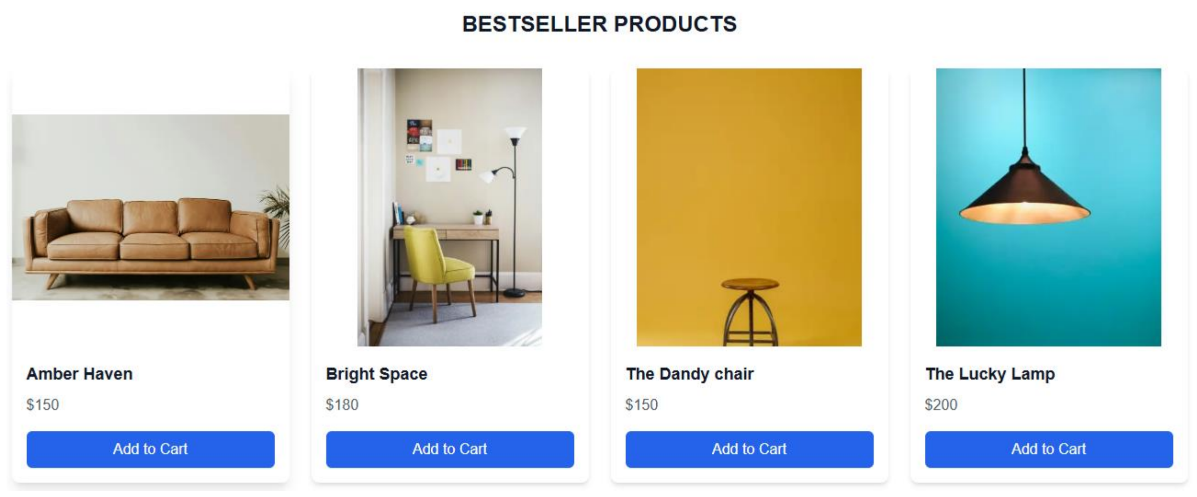
Example Code for Fetching Data on the Frontend:



```
async function getProduct(slug: string): Promise<Product> {
    return client.fetch(
        groq`*[_type == "product" && slug.current == $slug][0]{
            _id,
            name,
            _type,
            image,
            price,
            description,
        }`,
        { slug }
    );
}
```

Product Display:

Below is an image of the product listing page, where products are dynamically displayed on the frontend.



4. Conclusion

The API integration for E-Commerce was successfully implemented. The process involved retrieving data, storing it in Sanity CMS, and dynamically displaying it on the frontend. Attached screenshots confirm the successful execution at each stage. This integration optimizes product and category management, ensuring a smooth experience for both administrators and users.