

# Market Place Builder Hackathon

## Day 3: API Integration and Data Migration

### Report on API Integration

#### 1. Introduction

This document outlines the process of incorporating product and category data from an external API into the backend system of E-commerce, an online clothing store. The integration employs Sanity CMS for content management and Next.js for frontend rendering.

#### Primary Goals 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 Using Sanity and Schema

### Step 1: Transferring Data to Sanity CMS

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

#### 1.

```
import { defineType, defineField } from "sanity"

export const product = defineType({
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    defineField({
      name: "category",
      title: "Category",
      type: "reference",
      to: [{
        type: "category"
      }]
    })
  ]
})
```

```
}
),
defineField({
  name: "name",
  title: "Title",
  validation: (rule) => rule.required(),
  type: "string"
}),
defineField({
  name: "slug",
  title: "Slug",
  validation: (rule) => rule.required(),
  type: "slug"
}),
defineField({
  name: "inventory",
  title: "Inventory",
  validation: (rule) => rule.required(),
  type: "number"
}),
defineField({
  name: "image",
  type: "image",
  validation: (rule) => rule.required(),
  title: "Product Image"
}),
defineField({
  name: "price",
  type: "number",
  validation: (rule) => rule.required(),
  title: "Price",
}),
defineField({
  name: "quantity",
  title: "Quantity",
  type: "number",
  validation: (rule) => rule.min(0),
}),
defineField({
  name: "tags",
  type: "array",
  title: "Tags",
  of: [{
    type: "string"
  }]
})
```

```

    })),
    defineField({
      name: 'description',
      title: 'Description',
      type: 'text',
      description: 'Detailed description of the product',
    }),
    defineField({
      name: 'features',
      title: 'Features',
      type: 'array',
      of: [{ type: 'string' }],
      description: 'List of key features of the product',
    }),
    defineField({
      name: 'dimensions',
      title: 'Dimensions',
      type: 'object',
      fields: [
        { name: 'height', title: 'Height', type: 'string' },
        { name: 'width', title: 'Width', type: 'string' },
        { name: 'depth', title: 'Depth', type: 'string' },
      ],
      description: 'Dimensions of the product',
    }),
  ],
})),
})

```

## 2.

```

import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';
import slugify from 'slugify';

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,

```

```

    useCdn: false,
    token: process.env.SANITY_API_TOKEN,
    apiVersion: '2021-08-31'
  });

  async function uploadImageToSanity(imageUrl) {
    try {
      console.log(`Uploading image: ${imageUrl}`);
      const response = await axios.get(imageUrl, { responseType: 'arraybuffer'
    });

    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      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 createCategory(category, counter) {
    try {
      const categoryExist = await client.fetch(`*[_type=="category" &&
slug.current==$slug][0]`, { slug: category.slug });
      if (categoryExist) {
        return categoryExist._id;
      }
      const catObj = {
        _type: "category",
        _id: `${category.slug}-${counter}`,
        name: category.name,
        slug: {
          _type: 'slug',
          current: slugify(category.name || 'default-category', { lower:
true, strict: true })
        }
      };
      const response = await client.createOrReplace(catObj);
      console.log('Category created successfully', response);
      return response._id;
    } catch (error) {
      console.error('Failed to create category:', category.name, error);
      return null;
    }
  }

```

```

    }
}

async function importData() {
  try {
    console.log('Fetching products from API...');
    const response = await axios.get('https://hackathon-apis.vercel.app/api/products');
    const products = response.data;
    let counter = 1;

    for (const product of products) {
      console.log(`Processing product: ${product.name}`);
      let imageRef = null;
      let catRef = null;

      if (product.image) {
        imageRef = await uploadImageToSanity(product.image);
      }
      if (product.category?.name) {
        catRef = await createCategory(product.category, counter);
      }

      const sanityProduct = {
        _id: `product-${counter}`,
        _type: 'product',
        name: product.name,
        slug: {
          _type: 'slug',
          current: slugify(product.name, { lower: true, strict: true })
        },
        price: product.price,
        category: catRef ? {
          _type: 'reference',
          _ref: catRef
        } : undefined,
        tags: product.tags || [],
        quantity: 50,
        image: imageRef ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef
          }
        } : undefined,

```

```

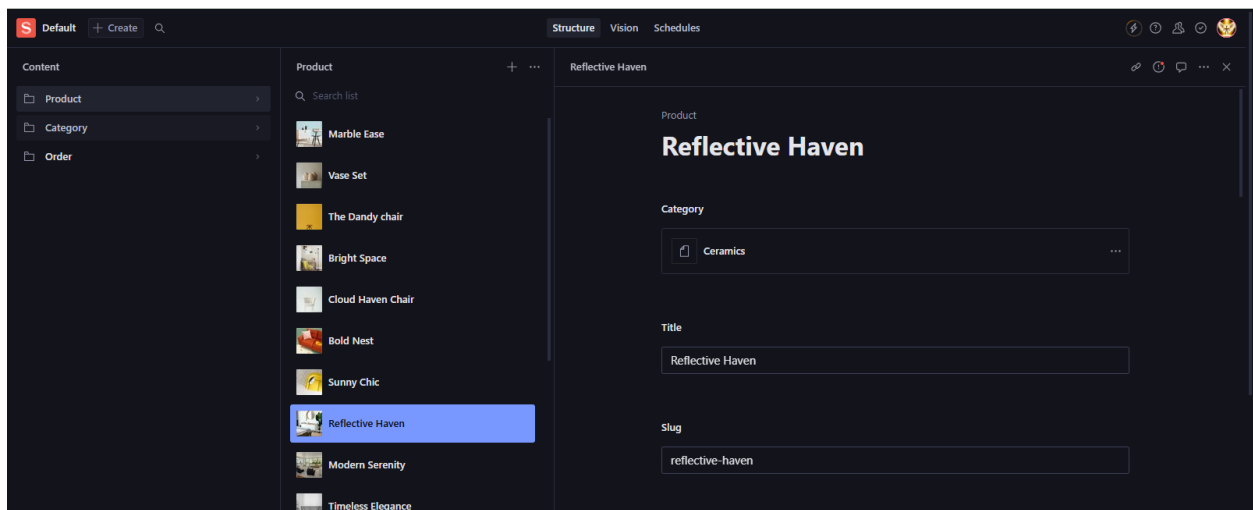
        description: product.description || "Default product
description",
        features: product.features || ["Premium material", "Handmade
upholstery"],
        dimensions: product.dimensions || {
            _type: 'dimensions',
            height: "110cm",
            width: "75cm",
            depth: "50cm"
        }
    };

    await client.createOrReplace(sanityProduct);
    console.log(`✔ Imported product: ${sanityProduct.name}`);
    counter++;
}
console.log('✔ Data import completed!');
} catch (error) {
    console.error('Error importing data:', error);
}
}

importData();

```

## Sanity Dashboard Screenshot Displaying Stored Products



## Step 2: Fetching Data from the API

The API provides dedicated endpoints for retrieving data, such as:

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

Base api url:

<https://hackathon-apis.vercel.app/api/products>

```
async function importData() {  
  try {  
    console.log('Fetching products from API...');  
    const response = await axios.get('https://hackathon-apis.vercel.app/api/products');  
  }  
}
```

## 4. Frontend Integration

The stored product data was fetched from **Sanity CMS** and displayed on the **E-Commerce** frontend. With **Next.js**, the data was dynamically rendered to populate the product listing interface, providing a smooth and engaging shopping experience.


### Code Example for Retrieving 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 a screenshot of the product listing page, showcasing products dynamically rendered on the frontend.

New Ceramics




The Poplar suede sofa  
£980

Add to Cart




Tropical Vibe  
£550

Add to Cart



Sleek Living  
£300

Add to Cart



Serene Seat  
£350

Add to Cart

## 4. Conclusion

The API integration for E-Commerce was successfully carried out. The process included fetching data, storing it in Sanity CMS, and dynamically rendering it on the frontend. Attached screenshots validate the successful execution of each step. This integration enhances product and category management, ensuring a seamless experience for both administrators and users.