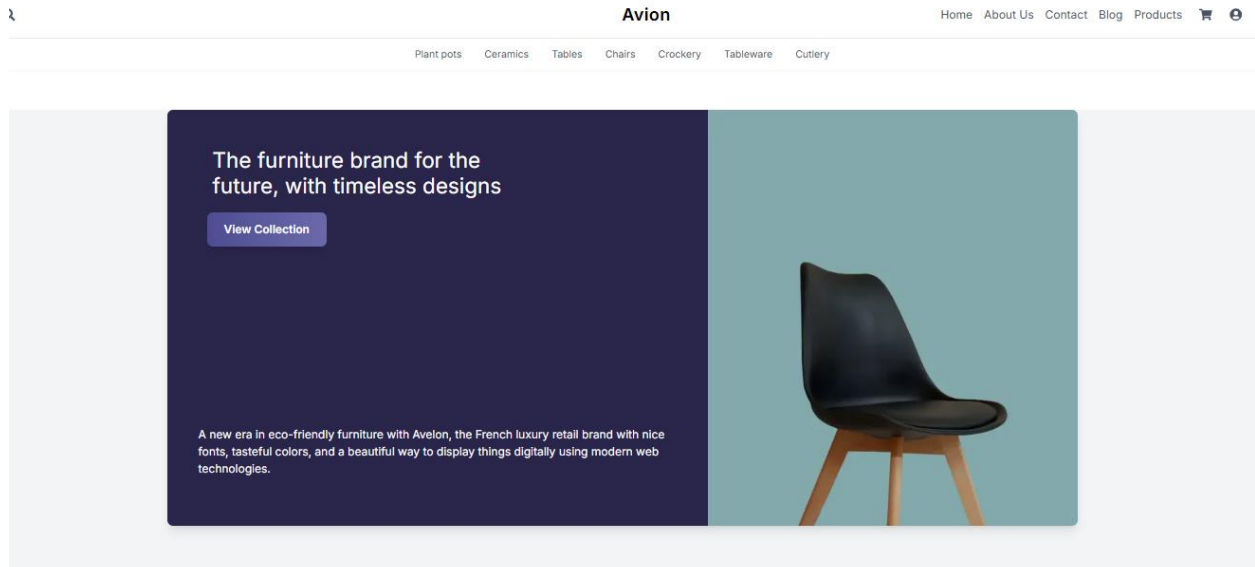


# HECKATHON 3

## DAY 3 - API INTEGRATION AND DATA MIGRATION

### [E- COMMERCE]



### ✓ *API integration process:*

#### **Requirement: -**

The task was to integrate the given API in to our marketplace. And send that Data to sanity CMS and after that export that data or fetch that data to the frontend in to the marketplace which is [E- Commerce].

#### **Step 1: Understand the API**

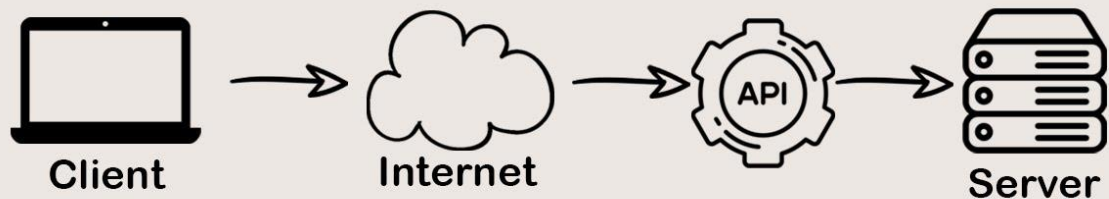
- **Action:** The first step in API integration is to thoroughly review the API documentation provided by the service. This documentation contains critical information on available endpoints, data formats, request methods (GET, POST, PUT, DELETE), and authentication methods.
- **Example:** For Sanity CMS, the API documentation outlines how to interact with the CMS for creating and managing content. This includes endpoints like `/products`, `/categories`, etc.

## Step 2: Set Up Authentication

- **Obtain API Keys:** Many APIs require an authentication key (API key, OAuth token) to make requests. Obtain this key from the API provider.
- **Set Up Authentication:** Implement the required authentication method in your code (e.g., include the API key in headers or use OAuth).

**Example:** For Sanity CMS, an API key or OAuth token is needed to authenticate API requests. This key must be included in the header of each request to ensure the request is authorized.

# What is an API?



## Step 3: Make API Requests to External Service

- Send a GET or POST request to the external API.
- Retrieve the required data (e.g., product listings).
- Format the data to match Sanity's structure.
- Send the formatted data to Sanity.

#### **Step 4: Process the Data**

- Action: After receiving the data, clean, transform, and map it to the correct fields in your system.

Example: Map product details (name, price, description) to the corresponding fields in your schema.

#### **Step 5: Send Data to Sanity CMS**

- Action: Send processed data to Sanity using a POST request to create or a PUT request to update content.

### **✓ Adjustment Made to Schemas:**

#### **Step 1: Add New Fields for API Data**

Action: Add fields to store API data, like an image URL for product images.

#### **Step 2: Adjust Data Types**

Action: Ensure schema data types match API data, like converting price from string to number.

#### **Step 3: Create Relationships Between Tables**

Action: Link related data, such as products and categories, using references.

#### **Step 4: Normalize Data**

Action: Organize data by separating repeating info into related schemas, like product images or prices.

#### **Step 5: Versioning and Data Validation**

Action: Add fields for API versioning and validation to ensure accurate and up-to-date data.

## ✓ **Migration steps and tools used**

### → **Assess Current Schema**

Review the existing schema to identify changes needed for API data.

*Purpose: Understand current structure and plan necessary adjustments for API compatibility.*

### → **Prepare Data Mapping Plan**

Create a document mapping API data to your Sanity schema.

*Purpose: Ensure all relevant data is mapped accurately to the appropriate fields.*

### → **Select Migration Tools**

Choose tools like Talend, Flyway, or custom scripts for migration.

*Purpose: Automate the migration process and ensure efficiency and reliability.*

### → **Cleanse and Transform Data**

Remove duplicates and correct formatting to ensure clean, valid data.

*Purpose: Prepare the data for smooth migration and avoid errors.*

### → **Migrate Data**

Use tools/scripts to move data from the old system or API to Sanity.

*Purpose: Transfer data into the new schema or CMS effectively.*

### → **Verify Migration Success**

Test to ensure the data is correctly reflected in Sanity.

*Purpose: Confirm that data has been migrated correctly and appears as expected.*

## → Post-Migration Adjustments

Perform cleanup, add indexes, and optimize schema performance.

*Purpose: Improve the system's performance and ensure efficient data handling.*

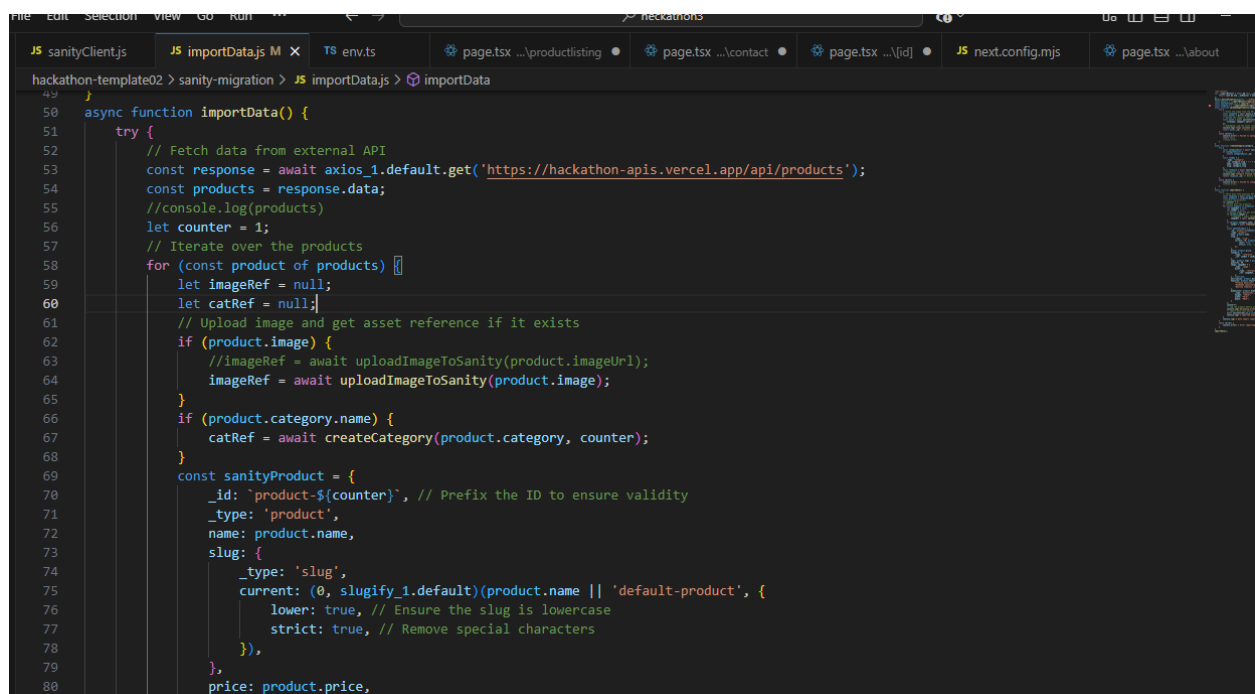
## → Test and Optimize

Test API integration and optimize for smooth operation.

*Purpose: Ensure smooth data retrieval and system performance after migration.*

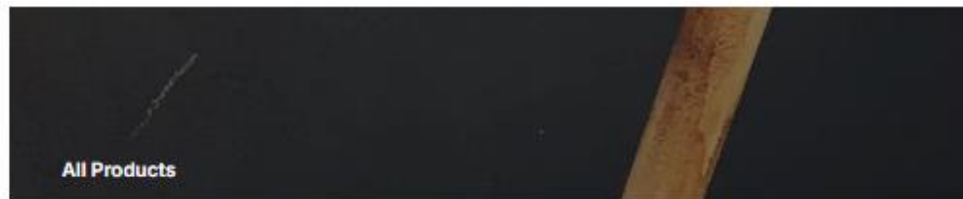
## IMAGES

API calls: -



```
File Edit Selection View Go Run Hackathon03
JS sanityClient.js JS importData.js M X TS env.ts page.tsx ...productlisting page.tsx ...contact page.tsx ...id JS next.config.mjs page.tsx ...about
hackathon-template02 > sanity-migration > JS importData.js > importData
49
50 async function importData() {
51   try {
52     // Fetch data from external API
53     const response = await axios_1.default.get('https://hackathon-apis.vercel.app/api/products');
54     const products = response.data;
55     //console.log(products)
56     let counter = 1;
57     // Iterate over the products
58     for (const product of products) {
59       let imageRef = null;
60       let catRef = null;
61       // Upload image and get asset reference if it exists
62       if (product.image) {
63         //imageRef = await uploadImageToSanity(product.imageUrl);
64         imageRef = await uploadImageToSanity(product.image);
65       }
66       if (product.category.name) {
67         catRef = await createCategory(product.category, counter);
68       }
69       const sanityProduct = {
70         _id: `product-${counter}`, // Prefix the ID to ensure validity
71         _type: 'product',
72         name: product.name,
73         slug: {
74           _type: 'slug',
75           current: (0, slugify_1.default)(product.name || 'default-product', {
76             lower: true, // Ensure the slug is lowercase
77             strict: true, // Remove special characters
78           }),
79         },
80         price: product.price,
```

Data successfully displayed in the frontend: -



The Poplar suede sofa  
\$580



Tropical Vibe  
\$250



Sleek Living  
\$300



Serene Seat  
\$350



Nordic Elegance  
\$280



TimberCraft  
\$320



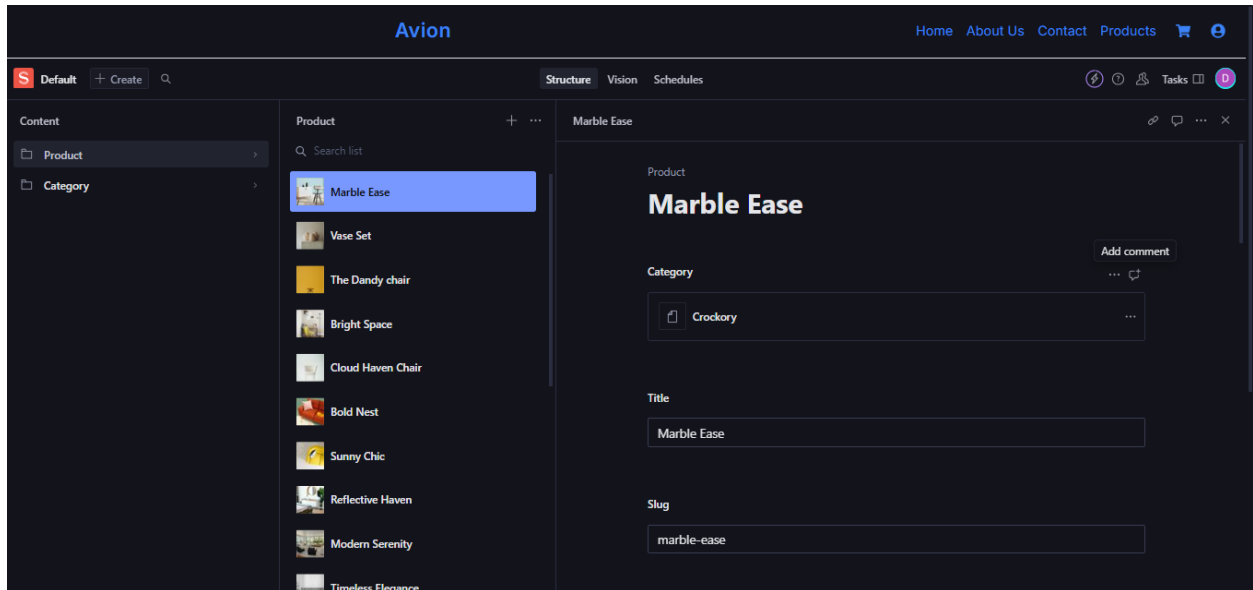
Timeless Elegance  
\$320



Modern Serenity  
\$480



## Populated Sanity CMS fields: -



## Code snippets for API integration and migration scripts: -

```
File Edit Selection View Go Run ...  heckathod
sanityClient.js importData.js M TS importData.js X TS envs  pagetx ...productlisting pagetx ...contact pagetx ...[id] nest.config.mjs pa ...
hackathon-template02 > sanity-migration > TS importData.js > createCategory
1  import axios from 'axios';
2  import { client } from './sanityClient.js';
3  import slugify from 'slugify';
4
5  async function uploadImageToSanity(imageUrl: string): Promise<string|null> {
6
7
8  try {
9    // Fetch the image from the URL and convert it to a buffer
10    const response = await axios.get(imageUrl, { responseType: 'arraybuffer', timeout: 10000 });
11    const buffer = Buffer.from(response.data);
12
13    // Upload the image to Sanity
14    const asset = await client.assets.upload('image', buffer, {
15      filename: imageUrl.split('/').pop(), // Extract the filename from URL
16    });
17
18    // Debugging: Log the asset returned by Sanity
19    console.log('Image uploaded successfully:', asset);
20
21    return asset._id; // Return the uploaded image asset reference ID
22  } catch (error) {
23    console.error('❌ Failed to upload image:', imageUrl, error);
24    return null;
25    //throw error;
26  }
27
28  interface Category {
29    _id:string,
30    _type?:string,
31    name:string,
32    slug:string
```

```
File Edit Selection View Go Run ... heckathon3
sanityClient.js importData.js M TS importData.ts X TS envs page.tsx ...productListing page.tsx ...contact page.tsx ...[id] next.config.mjs pa ...
hackathon-template02 > sanity-migration > TS importData.ts > createCategory

34
35 async function createCategory(category:Category,counter:number) {
36
37   try {
38     const categoryExist = await client.fetch("[_type=='category' && slug=='$slug'][0]",(slug:category.slug))
39     if(categoryExist)
40     {
41       return categoryExist._id
42     }
43     const catObj = {
44       _type:"category",
45       _id:category.slug+"-"+counter,
46       name:category.name,
47       slug:category.slug
48     }
49     const response = await client.createOrReplace(catObj)
50
51
52     // Debugging: Log the asset returned by Sanity
53     console.log('Category created successfully', response);
54
55     return response._id; // Return the uploaded image asset reference ID
56   } catch (error) {
57     console.error("❌ Failed to category:", category.name, error);
58     //throw error;
59   }
60 }
61
62
63 async function importData() {
64   try {
```