

# Day-3 API Integration and Data Migration Report

## Introduction:

This document provides a concise overview of the process we follow to fetch data from an API, migrate the retrieved data into Sanity, and subsequently display the product information on the frontend. It outlines the key steps and methods used to ensure seamless integration and efficient data management across the system.

## API Integration Process:

**API Endpoints:** The provided API includes the /product endpoint.

**Fetching Data:** We utilized the Axios library to perform HTTP requests to the API.

**A Get Method is Called:** and the API URL is provided within the code to fetch data from Sanity. The response is then stored in a variable named response for future use.

## Adjustments Made to Schemas:

To changes in the product data structure, we incorporated additional fields into the API Schema:

**Category Field:** This field categorizes products, facilitating organized data retrieval and display.

**Tag Field:** Tags were added to allow for more flexible filtering and searching of products based on specific attributes.

**Size Field:** Inclusion of size information enables users to select products according to their dimensional preferences.

**Color Field:** Adding color details provide users with the option to choose products based on color variants.

These schema enhancements aim to improve the user experience by providing more detailed and organized product information.

```
        title: "Discount Percentage",
      },
      {
        name: "isNew",
        type: "boolean",
```

```

        title:"New Badge",
      }
    ]
  })
})

```

## Migration Steps:

### Installing:

First step: install all the libraries needed to import data from API such as axios Sanity setup.

### Sanity Setup:

Then install the sanity.io to make product schema in which we will put our data after fetching it from api and display in sanity studio

### Setup Sanity Client:

Create a sanityClient.js file inside a sanity-migration folder to configure the sanity client. The Client will be used to interact with your sanity backend for data migration.

```

src > sanity > lib > TS client.ts > client > token
1  import { createClient } from 'next-sanity'
2
3  import { apiVersion, dataset, projectId } from '../env'
4
5  export const client = createClient({
6    projectId,
7    dataset,
8    apiVersion,
9    useCdn: true,
10   token: "skIihA21wEoCjNTBgI53licK4r1KtxV7sr19AY5KDwaRlSvyfQTGpJ6TKFwEMirDrKppVVvvhAoQDKwVp9pnfAlKI00wRGn6ZZ6Nl1GqzAFJNkD9zLjRmzwph
11
12   // Set to false if statically generating pages, using ISR or tag-based revalidation
13 })
14

```

## Create importData.js for Data Migration:

Create an importData.js script to fetch data from the api using Axios. Transform the fetched data into a format compatible with you sanity schema and integrate into sanity .

```

sync 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: dicountPercentage -> discountPercentage
        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();

```

## ToolsUsed:

### Axios:

Used for fetching data from api,including sanity if needed for read operation.

Allow making HTTP requests (GET, POST etc) seamlessly.

**Dotenv:** Used to manage and store sensitive configuration details like API key, token and URLs in .env-file.

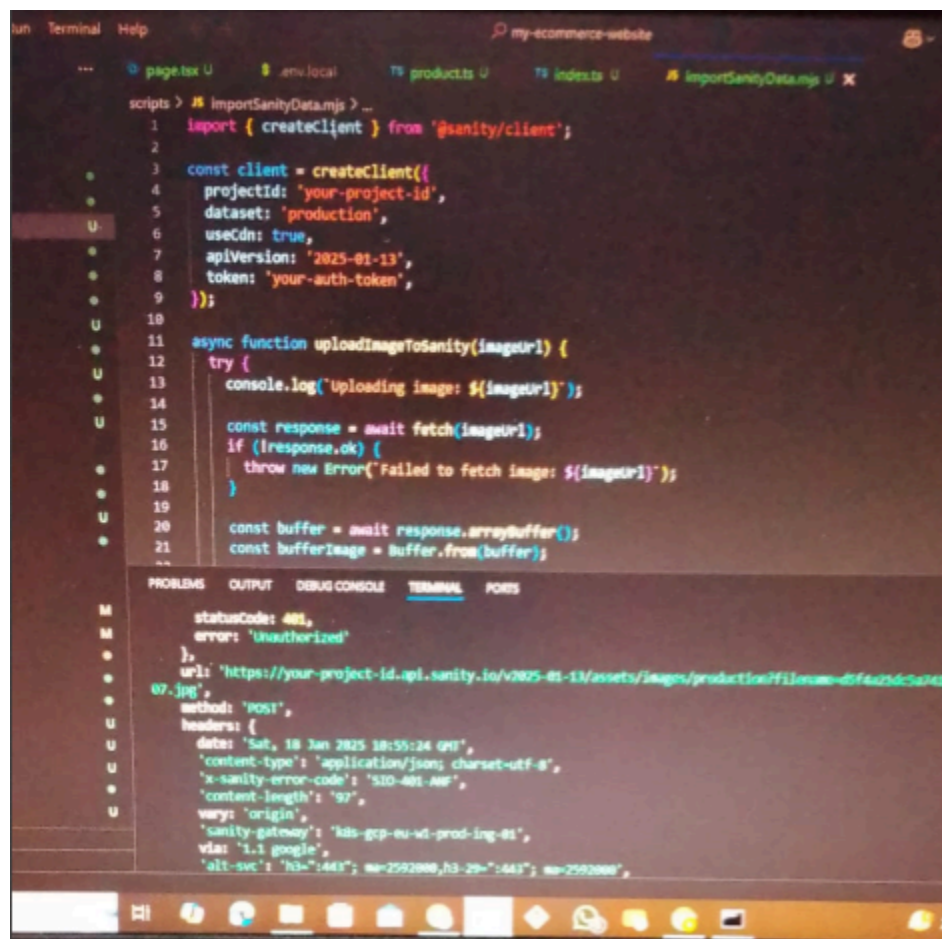
Keep environmental variables secure and out of the main codebase.

### Sanity Client:

A javascript client library for interacting with the sanity CMS.

Used for writing data to sanity fetching data and other CRUD operations

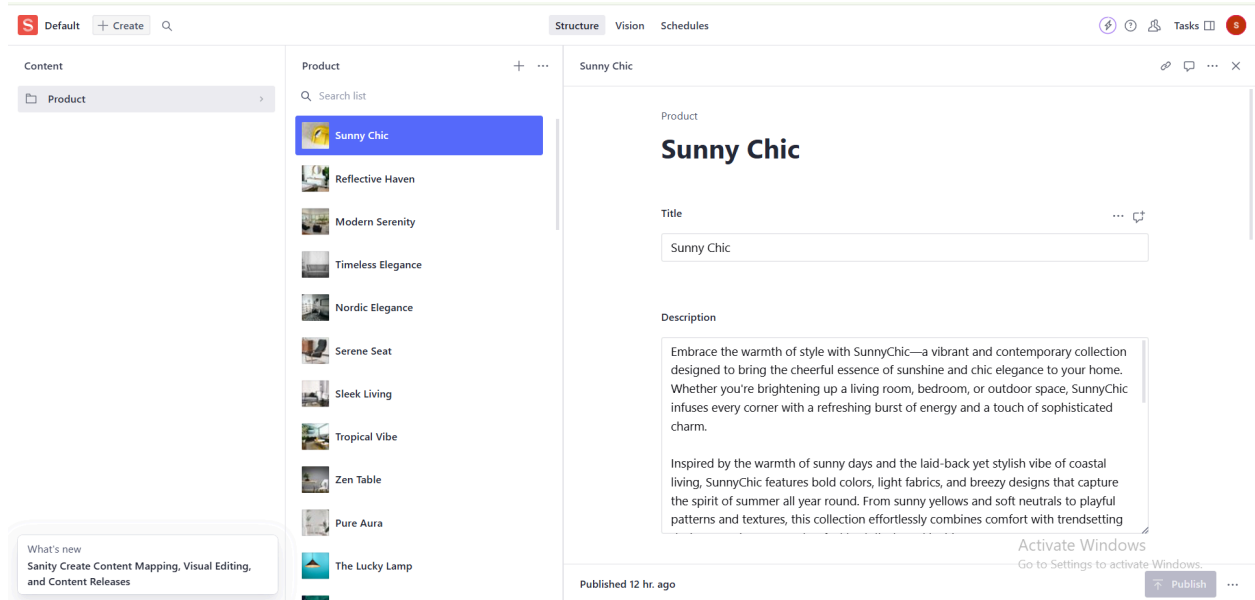
## 1-API Calls Output:



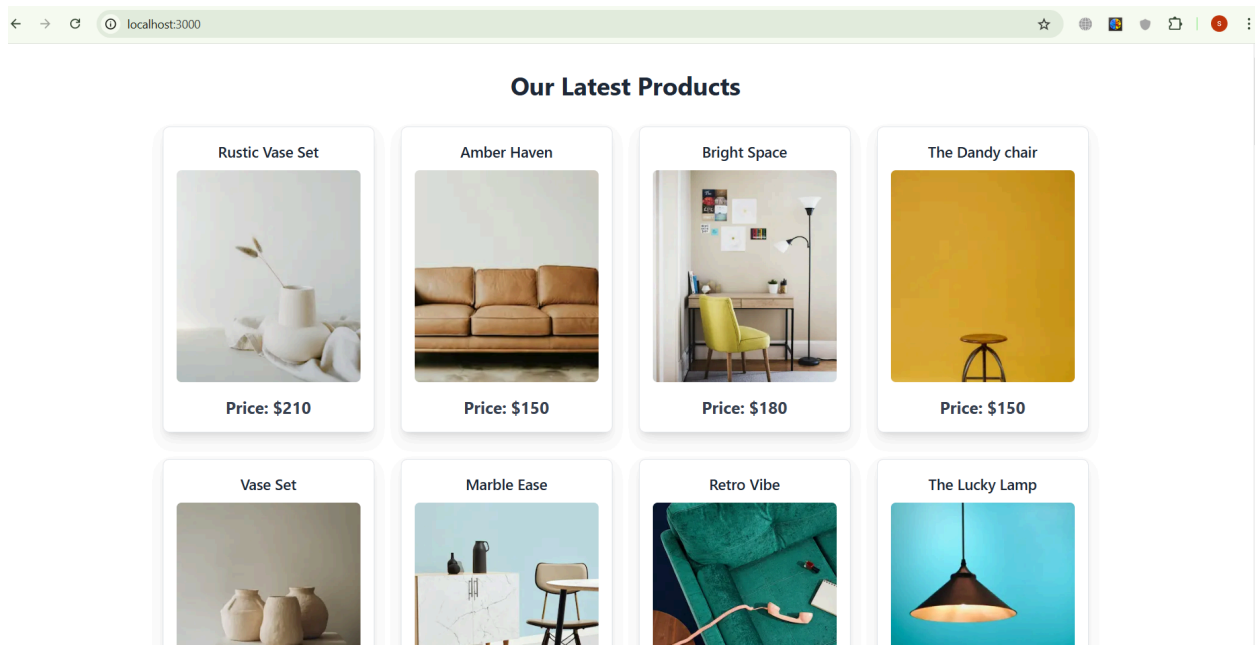
```
scripts > # importSanityData.mjs > ...
1 import { createClient } from '@sanity/client';
2
3 const client = createClient({
4   projectId: 'your-project-id',
5   dataset: 'production',
6   useCdn: true,
7   apiVersion: '2025-01-13',
8   token: 'your-auth-token',
9 });
10
11 async function uploadImageToSanity(imageUrl) {
12   try {
13     console.log('Uploading image: ${imageUrl}');
14
15     const response = await fetch(imageUrl);
16     if (!response.ok) {
17       throw new Error('Failed to fetch image: ${imageUrl}');
18     }
19
20     const buffer = await response.arrayBuffer();
21     const bufferImage = Buffer.from(buffer);
22   } catch (error) {
23     console.error('Error uploading image: ', error);
24   }
25 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
statusCode: 401,
error: 'Unauthorized'
},
url: 'https://your-project-id.api.sanity.io/v2025-01-13/assets/images/production/filename-d0fe425d5a7428
07.jpg',
method: 'POST',
headers: {
  date: 'Sat, 18 Jan 2025 18:55:24 GMT',
  'content-type': 'application/json; charset=utf-8',
  'x-sanity-error-code': '510-401-Auth',
  'content-length': '92',
  vary: 'origin',
  'sanity-gateway': 'kls-gcp-eu-w1-prod-ing-01',
  via: '1.1 google',
  'alt-svc': 'h3=":443"; ma=2592000,h3-29=":443"; ma=2592000',
}
```

## 2-Sanity Schema Fields:



### 3-Display In FrontendData :



### Conclusion:

We successfully fetched data from an external API, transformed it to match the required structure, and integrated it with sanity CMS. The data was migrated to sanity making it the data dynamically on the frontend.

