

# Hackathon Day 3

## API Integration and Data Migration to Sanity CMS

### 1. Overview

The main goal was to transfer product data from an external API to Sanity CMS. This includes getting product details like names, prices, descriptions, and images, then uploading them to Sanity so they can be used on the website.

---

### 2. Tools and Libraries

To complete this task, I used these tools:

- **Sanity Client:** This helped in connecting and interacting with Sanity CMS.
  - **Axios:** Used for making API requests and retrieving data from the external API.
  - **.env:** Used to securely store sensitive information, like the API token, in environment variables.
  - **Path and fileURLToPath:** These helped in setting the correct paths for environment files.
  - **Buffer:** Used for handling image files when uploading them to Sanity.
- 

### 3. Environment Setup

I stored the sensitive information, such as API tokens and project details, in a `.env.local` file to keep them secure. This file contains:

- **NEXT\_PUBLIC\_SANITY\_PROJECT\_ID:** Your unique project ID from Sanity.
  - **NEXT\_PUBLIC\_SANITY\_DATASET:** The dataset name you're using in Sanity.
  - **SANITY\_API\_TOKEN:** The token used to authenticate with Sanity.
- 

### 4. Uploading Images

To upload images:

- I fetched the image from the URL and converted it into binary data.
  - The image was then uploaded to Sanity, and I stored the image's reference in the product's data.
- 

### 5. Migrating Data

Here's the process I followed to migrate the data:

1. **Fetching Product Data:**

- The product data was fetched from the API (<https://template-03-api.vercel.app/api/products>).
- This data contains product details like name, price, category, description, and image URL.

## 2. Storing Data in Sanity:

- For each product, I uploaded the image , stored the image reference, and then created a product document in Sanity.
- Each product is added to Sanity with all its details, ready to be displayed on the website.

## 3. Error Handling:

- If there were any issues (such as problems fetching the data or uploading the image), the script would catch the errors and print an error message.

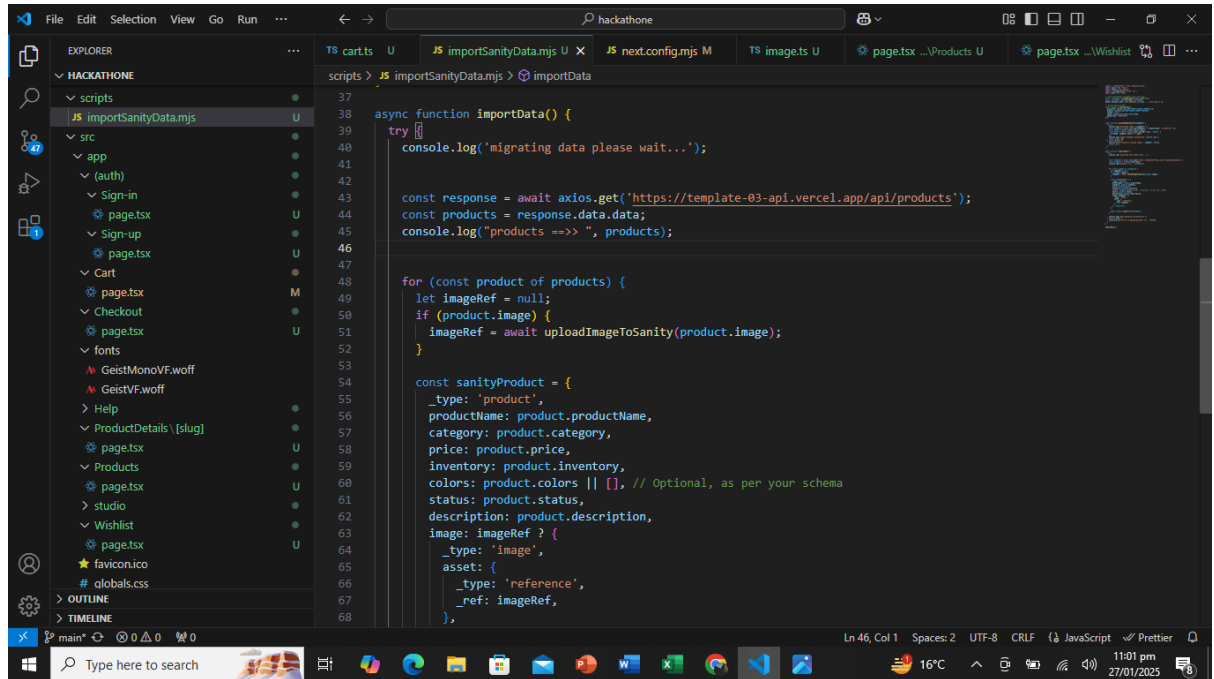
---

## 6. Screenshots & code snipts.

### . Api integration.

```
6 }
7
8 async function importData() {
9   try {
10     console.log('migrating data please wait...');
11
12     // API endpoint containing car data
13     const response = await axios.get('https://template-03-api.vercel.app/api/products');
14     const products = response.data.data;
15     console.log("products ==>> ", products);
16   }
17 }
```

- **Migrating Data.**



```

37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
async function importData() {
  try {
    console.log('migrating data please wait...');

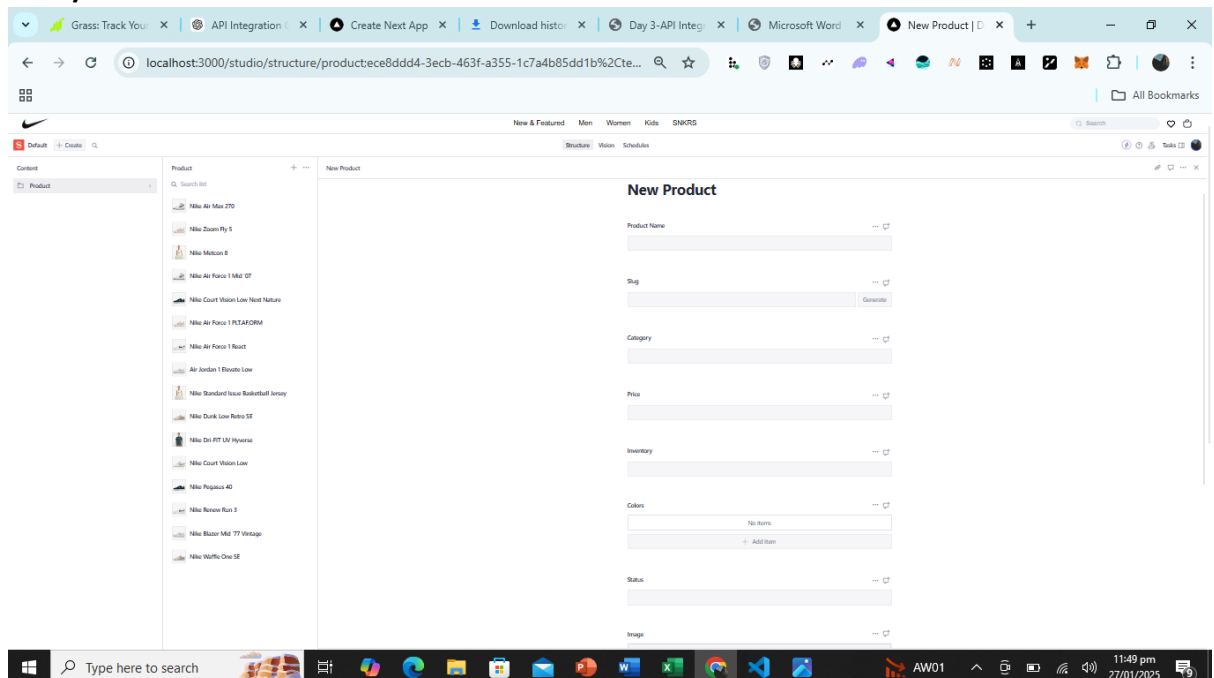
    const response = await axios.get('https://template-03-api.vercel.app/api/products');
    const products = response.data.data;
    console.log("products ==> ", products);

    for (const product of products) {
      let imageRef = null;
      if (product.image) {
        imageRef = await uploadImageToSanity(product.image);
      }

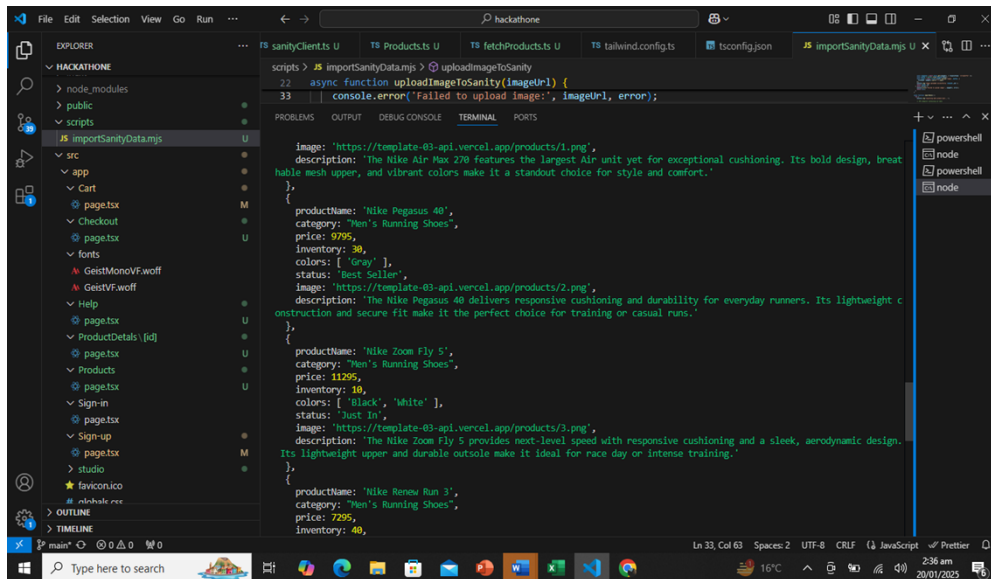
      const sanityProduct = {
        _type: 'product',
        productName: product.productName,
        category: product.category,
        price: product.price,
        inventory: product.inventory,
        colors: product.colors || [], // Optional, as per your schema
        status: product.status,
        description: product.description,
        image: imageRef ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        } : null,
      };
    }
  } catch (error) {
    console.error('Error migrating data:', error);
  }
}

```

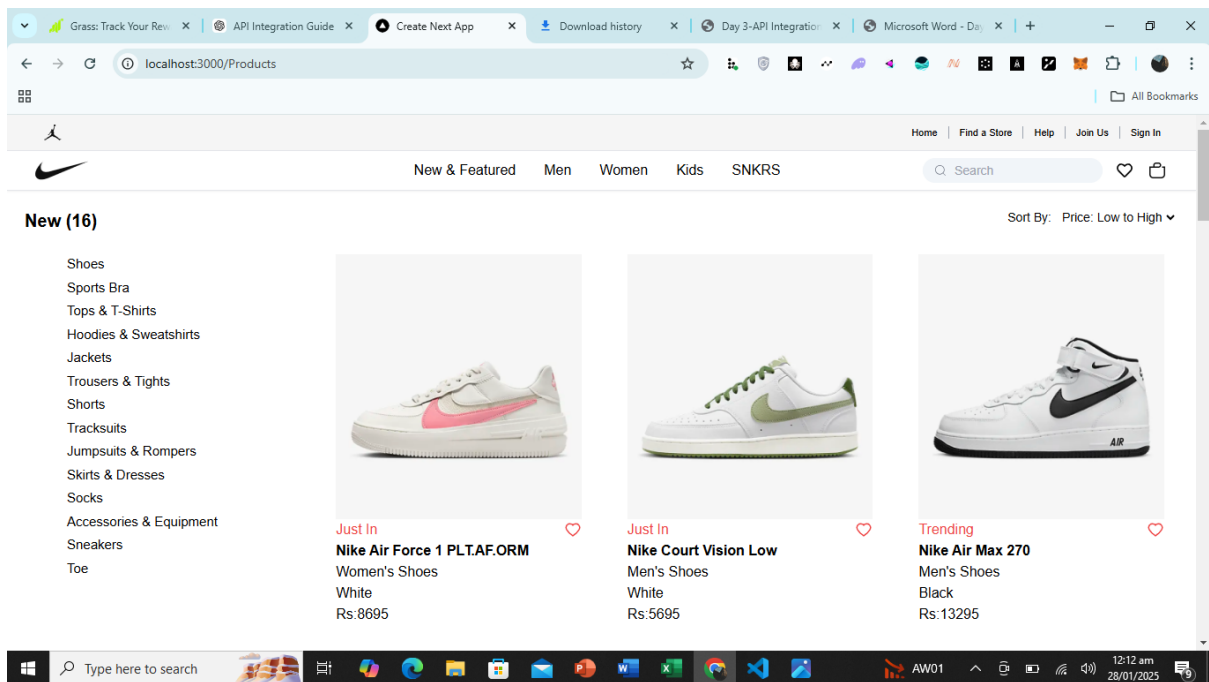
- **Sanity CMS:**



- Data Successfully imported in sanity.



- Frontend Display:



## 7. Final Thoughts

This process successfully moved all the product data from the external API into Sanity CMS, including handling images and ensuring all the data was structured properly. Now, the data final rendered.