

## **Hackathon Day 3: API Integration and Data Migration**

*On Day 3 of the hackathon, the primary focus was on API integration and migrating data into Sanity CMS to build a functional marketplace backend. Below is a detailed account of the steps I followed to complete the task successfully.*

### **Step 1: Understanding the Task and APIs**

*I started by thoroughly reviewing the provided API documentation and endpoints. The key endpoints included data for product listings, categories, and other relevant details. This step was crucial to understand the data structure and its compatibility with my existing Sanity CMS schema.*

### **Step 2: Schema Validation and Adjustments**

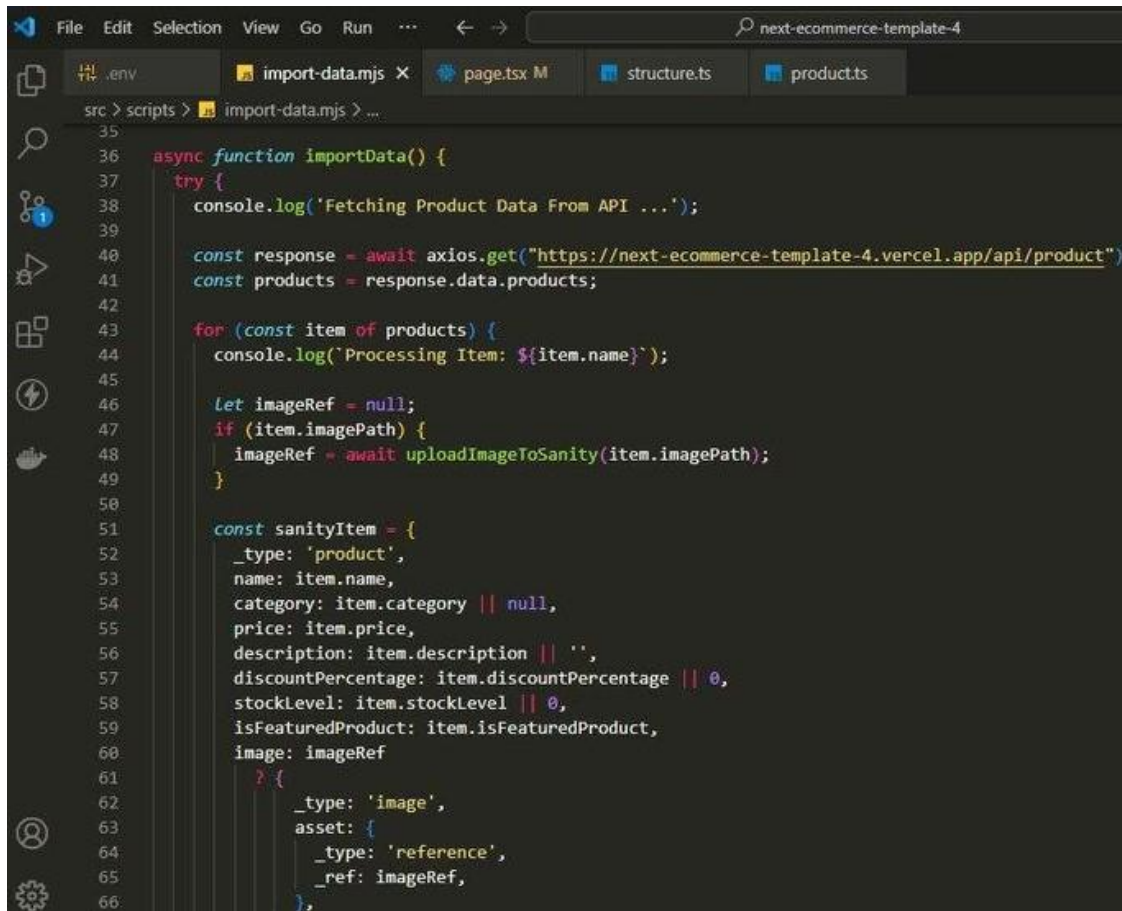
*Next, I validated my Sanity CMS schema against the API data. This involved comparing field names and data types to ensure compatibility. For instance, fields like name in the API were mapped to product\_title in my schema. Adjustments were made to align the schema with the incoming data.*

### **Step 3: Data Migration into Sanity CMS**

*I migrated the data into Sanity CMS using the following approach:*

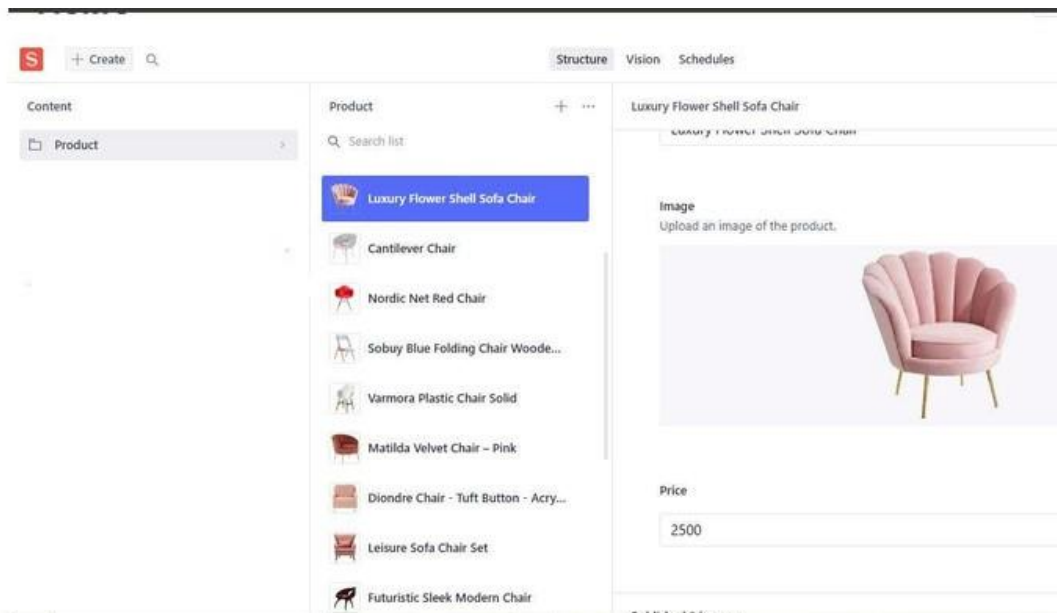
**.env file:** I copy my project Id and Token to .env file for security reasons that no one can access my Id or Token, this will ensure my data is safe.

*Script-based Migration:* I clone the repo which was provided by our faculty there was a script to fetch data from the provided API and transform it to match my schema. This script ensured the data was imported accurately and efficiently.

A screenshot of a code editor window titled 'next-e-commerce-template-4'. The editor shows a file named 'import-data.mjs' with the following JavaScript code:

```
35
36 async function importData() {
37   try {
38     console.log('Fetching Product Data From API ...');
39
40     const response = await axios.get("https://next-e-commerce-template-4.vercel.app/api/product");
41     const products = response.data.products;
42
43     for (const item of products) {
44       console.log('Processing Item: ${item.name}');
45
46       let imageRef = null;
47       if (item.imagePath) {
48         imageRef = await uploadImageToSanity(item.imagePath);
49       }
50
51       const sanityItem = {
52         _type: 'product',
53         name: item.name,
54         category: item.category || null,
55         price: item.price,
56         description: item.description || '',
57         discountPercentage: item.discountPercentage || 0,
58         stockLevel: item.stockLevel || 0,
59         isFeaturedProduct: item.isFeaturedProduct,
60         image: imageRef
61       };
62       if (imageRef) {
63         const imageItem = {
64           _type: 'image',
65           asset: {
66             _type: 'reference',
67             _ref: imageRef,
68           },
69         };
70       }
71     }
72   } catch (error) {
73     console.error('Error importing data:', error);
74   }
75 }
```

**Validation:** *After importing the data, I validated it within CMD and Sanity Studio to ensure all fields were correctly populated and aligned with the schema.*



```
C:\Windows\System32\cmd.exe
Uploading Image : https://next-ecommerce-template-4.vercel.app/product/Chair (22).png
Image Uploaded Successfully : image-c919c7d6e7b1d3ece287cde7c3fff40c97216ac3-374x374-png
Uploading Chair - Nordic Net Red Chair to Sanity !
Uploaded Successfully: vdNgKiFJm5ZxktgQt01dCN
-----

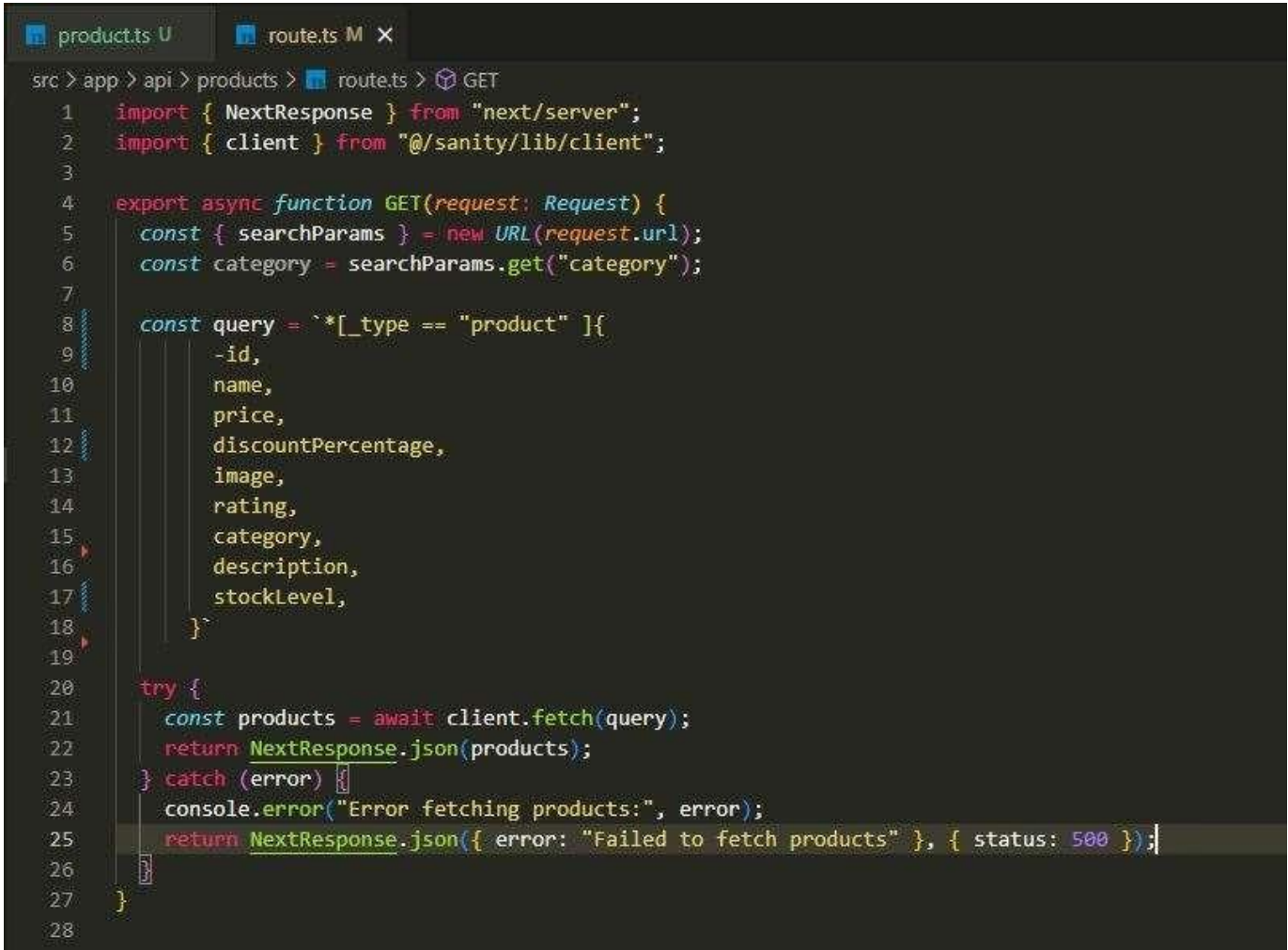
Processing Item: Cantilever Chair
Uploading Image : https://next-ecommerce-template-4.vercel.app/product/Chair (23).png
Image Uploaded Successfully : image-90a8d692777c7c2b80c7e49916bdce829702c35d-342x342-png
Uploading Chair - Cantilever Chair to Sanity !
Uploaded Successfully: lQ0y5s0sF1KY4sC8XATFU1
-----

Processing Item: Luxury Flower Shell Sofa Chair
Uploading Image : https://next-ecommerce-template-4.vercel.app/product/Chair (34).png
Image Uploaded Successfully : image-b36c424e4368829cc69cefdb42ba7b9d213e965e-1258x1258-png
Uploading Sofa - Luxury Flower Shell Sofa Chair to Sanity !
Uploaded Successfully: lQ0y5s0sF1KY4sC8XAYswZ
-----

Data Import Completed Successfully !

O:\GIAIC\Assinments\hackathone-e-commerce-q2\hackathone-3\next-ecommerce-template-4>
```

**Step 4: API Integration in Next.js**After completing the data migration, I focused on integrating the API into my Next.js project. This included:

A screenshot of a code editor with two tabs: 'products.ts U' and 'route.ts M X'. The 'route.ts' tab is active, showing a GET route handler. The code imports 'NextResponse' from 'next/server' and 'client' from '@sanity/lib/client'. It defines an async function GET(request: Request) that extracts search parameters from the request URL, specifically the 'category' parameter. A GraphQL query is constructed to fetch product details like id, name, price, discountPercentage, image, rating, category, description, and stockLevel. The code then attempts to fetch the data using the Sanity client. If successful, it returns the data as a JSON response. If an error occurs, it logs the error and returns a 500 status response with a message 'Failed to fetch products'.

```
src > app > api > products > route.ts > GET
1  import { NextResponse } from "next/server";
2  import { client } from "@sanity/lib/client";
3
4  export async function GET(request: Request) {
5    const { searchParams } = new URL(request.url);
6    const category = searchParams.get("category");
7
8    const query = `*[_type == "product" ]{
9      -id,
10     name,
11     price,
12     discountPercentage,
13     image,
14     rating,
15     category,
16     description,
17     stockLevel,
18   }`;
19
20   try {
21     const products = await client.fetch(query);
22     return NextResponse.json(products);
23   } catch (error) {
24     console.error("Error fetching products:", error);
25     return NextResponse.json({ error: "Failed to fetch products" }, { status: 500 });
26   }
27 }
28
```

1. **Creating Utility Functions:** I created reusable functions to fetch data from the API and Sanity CMS.
2. **Rendering Data:** The fetched data was displayed on the frontend, including product listings and categories.
3. **Error Handling:** Robust error-handling mechanisms were implemented to ensure a smooth user experience.

**Step 5: Testing and Validation**

To ensure everything was working as expected:

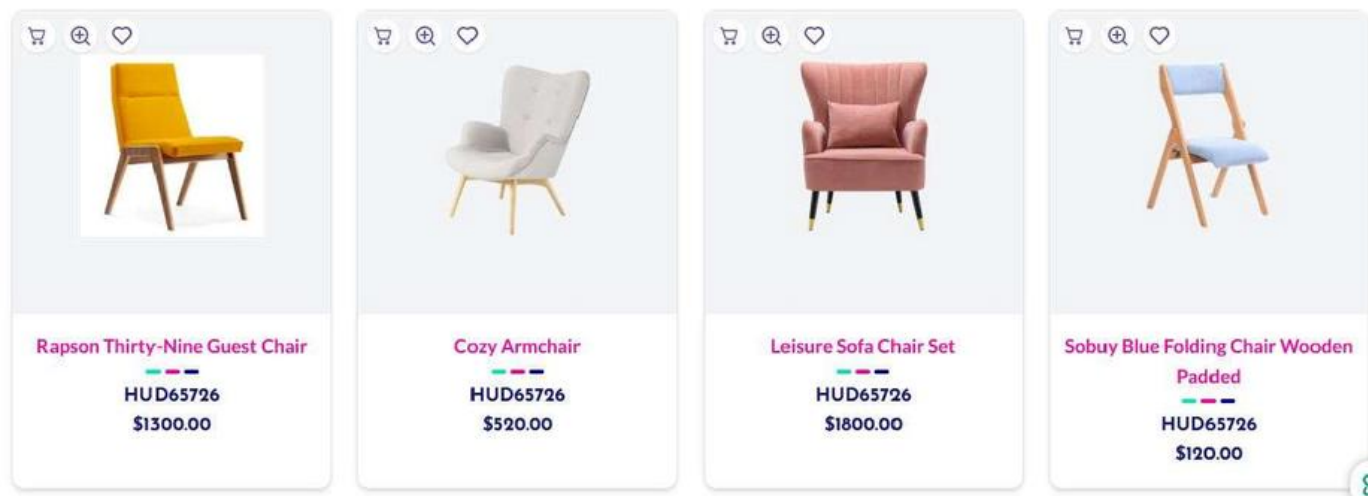
I used tools like Postman to test API endpoints.

*I logged responses to debug issues and confirm data accuracy. I tested the frontend to verify the correct display of data.*

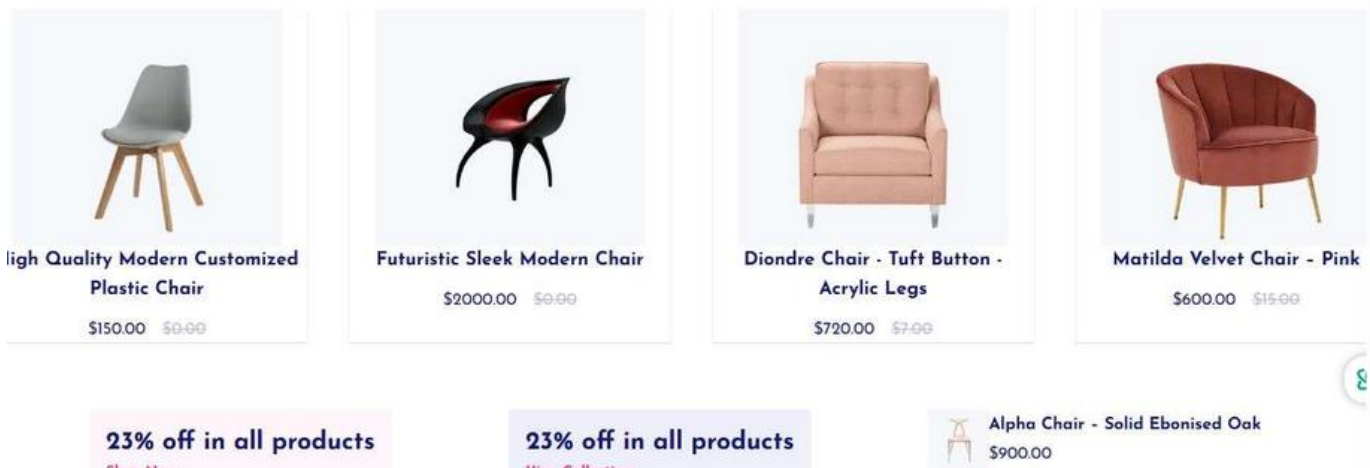
## **Final Outcome**

*The project resulted in:*

- 1. A fully populated Sanity CMS with imported data from the API.*
- 2. A functional API integration in Next.js, with data dynamically rendered on the frontend.*
- 3. Proper documentation of the entire process, including scripts, screenshots, and validation steps.*



## Trending Products



*This exercise provided hands-on experience in integrating APIs and managing data migration, preparing me to handle real-world challenges in marketplace development.*

## **Self-Validation Checklist**

Tasks	✓	✗
API Understanding	✓	
Schema Validation	✓	
Data Migration	✓	
API Integration in Next.js	✓	
Submission Preparation	✓	