

# Day 3 API INTEGRATION AND DATA MIGRATION

<b>Name</b>	<b>Shakir Hussain</b>
<b>Roll Number</b>	<b>00461483</b>
<b>Class Day</b>	<b>Tuesday</b>
<b>Timing</b>	<b>2pm to 5pm</b>

Content












📁 Product >

What's new  
Sanity Create Content Mapping, Visual Editing,  
and Content Releases

localhost:3000/studio/structure/product

Product + ...

🔍 Search list

-  Nike Waffle One SE
-  Nike Metcon 8
-  Nike Blazer Mid '77 Vintage
-  Nike Renew Run 3
-  Nike Zoom Fly 5
-  Nike Pegasus 40
-  Nike Air Max 270
-  Nike Court Vision Low
-  Nike Dri-FIT UV Hyverse
-  Nike Dunk Low Retro SE
-  Nike Standard Issue Basketball Jersey



Structure

Vision

Schedules

Tasks

Product

Search list

Nike Waffle One SE

Nike Metcon 8

Nike Blazer Mid '77 Vintage

Nike Renew Run 3

Nike Zoom Fly 5

Nike Pegasus 40

Nike Air Max 270

Nike Court Vision Low

Nike Dri-FIT UV Hyverse

Nike Dunk Low Retro SE

Nike Standard Issue Basketball Jersey

Air Jordan 1 Elevate Low

Nike Air Force 1 React

Published 1 hr. ago

Nike Blazer Mid '77 Vintage

Product

Nike Blazer Mid '77 Vintage

Product Name

Nike Blazer Mid '77 Vintage

Category

Men's Shoes

Price

8495

Inventory

22

Colors

Published 1 hr. ago

Publish

wKv1AMm1jkttyWGXVfs1

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

On Day 3 of the hackathon, the focus was on integrating a furniture-related API and migrating data into **Sanity CMS** to set up the backend for a fully functional furniture e-commerce website. Below is a detailed breakdown of the steps I followed to complete this task.

### Step 1: Understanding the Furniture API and Data Structure

I began by analyzing the documentation of the furniture API, which provided details about products such as **sofas**, **tables**, **chairs**, and other furniture items. The key endpoints included data for product details, categories (e.g., living room, office furniture), pricing, stock levels, and images. This initial review was essential to understand the structure of the API and ensure it aligned with my e-commerce website's requirements.

### Step 2: Validating and Customizing the Sanity CMS Schema

I validated my existing Sanity CMS schema to confirm compatibility with the furniture API data. During this process, I ensured that fields such as **name**, **category**, **price**, **imagePath**, and **description** matched the incoming data.

```
1 export const productSchema = {
2   ⚠ name: 'product',
3   title: 'Product',
4   type: 'document',
5   fields: [
6     {
7       name: 'productName',
8       title: 'Product Name',
9       type: 'string',
10    },
11    {
12      name: 'category',
13      title: 'Category',
14      type: 'string',
15    },
16    {
17      name: 'price',
18      title: 'Price',
19      type: 'number',
20    },
21    {
22      name: 'inventory',
23      title: 'Inventory',
24      type: 'number',
```

### Step 3: Fetching Furniture Data and Inserting into Sanity

Using **Axios**, I fetched data from the furniture API and wrote a script to migrate it into the Sanity CMS.

Below is the API integration and migration logic implemented in my Next.js project:

#### API ROUTE for Data Insertion:

```
async function importData() {
  try {
    console.log('migrating data please wait...');

    // API endpoint containing car data
    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,
          },
        } : undefined,
      };
    }
  }
}
```

By running this API route, all furniture data was successfully fetched and inserted into the Sanity CMS.

## **Step 4: Verifying and Displaying Data**

After migrating the data, I verified it in the Sanity Studio under the “NikeProduct” section. To ensure the data was displayed correctly on the website, I implemented a page to fetch and render it dynamically:

### **Fetch & Displaying Data in Next.js:**

## **Step 5: Testing and Feedback**

I tested the implementation by running the Next.js app and verified the functionality on [HTTP://LOCALHOST:3000/sHoP](http://localhost:3000/sHoP). All products, along with their details, were displayed successfully, providing a seamless user experience for browsing shoes, men shoes, shirt, and other shoes.

### **Final Thoughts:**

This process successfully set up the backend for a shoes e-commerce website using SanityCMS and integrated data from a shoes API.

The flexibility of Sanity CMS allowed me to customize the schema for shoes-store, while Next.js made it easy to fetch and display the data dynamically.