

# Hackathon Day 3 **API integration**

Nike General E Commerce website marketplace

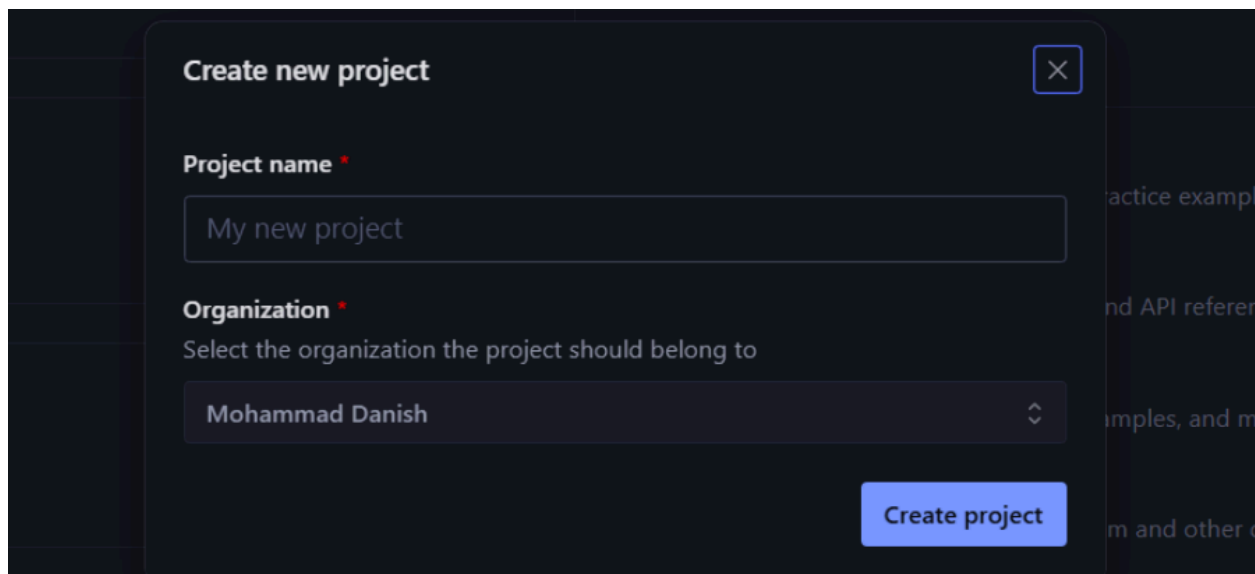
## API INTEGRATION PROCESS:

integrating APIs and managing data migration are essential skills in modern web development.

Recently, I worked on a project that required migrating product data using **Sanity**, a powerful headless CMS. This guide will walk you through the process I followed to ensure a smooth and efficient migration.

### Step 1: Setting Up a Project in Sanity

The first step was to create a project in Sanity. I visited [Sanity.io](https://sanity.io), signed in, and set up a new project using their intuitive interface. This step established the foundation for integrating product data seamlessly.



The screenshot shows the 'Create new project' dialog in the Sanity interface. It features a dark theme with a light blue 'Create project' button. The form includes a 'Project name' field with the placeholder text 'My new project' and an 'Organization' dropdown menu currently showing 'Mohammad Danish'. A close button (X) is located in the top right corner of the dialog.

Create new project

Project name \*

My new project

Organization \*

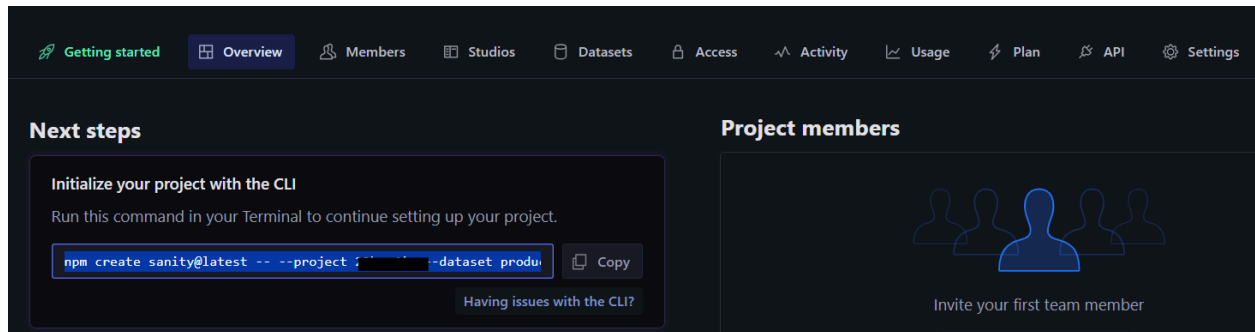
Select the organization the project should belong to

Mohammad Danish

Create project

## Step 2: Running the Initialization Command

Once the project was created, Sanity provided an initialization command. I copied the command and ran it in my terminal to set up the necessary configuration files locally.



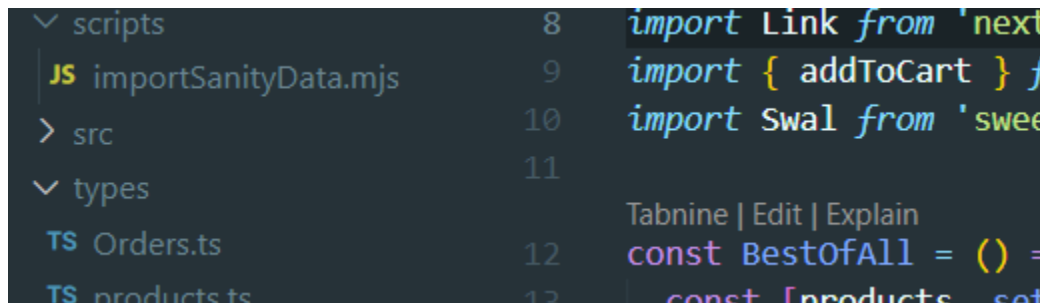
Copy cmd & paste into terminal

## Step 3: Organizing the File Structure

After initialization, I structured my Next.js project for efficient development. To handle the data migration scripts separately, I created a folder named scripts (outside the `src` folder) and added a new file:

📁 scripts/ImportSanityData.mjs

This file would contain the logic for data migration.



## Step 4: Adding Migration Code

Next, I pasted the migration script from [Sir Ameen Alam's](#) Day-3 documentation into `ImportSanityData.mjs`. This script handled the importing of product data into Sanity.

Additionally, I inserted the API URL in the script to establish a connection with Sanity's backend.

```
> .next
> node_modules
> public
  v scripts
    JS importSanityData.mjs
  > src
  > types
    TS Orders.ts
    TS products.ts
  $ .env.local
  @ .eslintrc.json
  @ .gitignore
  {} components.json
  TS next-env.d.ts
  TS next.config.ts
  {} package-lock.json
  {} package.json
  JS postcss.config.mjs
  TS sanity.cli.ts
  TS sanity.config.ts
  TS tailwind.config.ts
  tsconfig.json

1 import { createClient } from '@sanity/client';
2 import axios from 'axios';
3 import dotenv from 'dotenv';
4 import { fileURLToPath } from 'url';
5 import path from 'path';
6
7 // Load environment variables from .env.local
8 const __filename = fileURLToPath(import.meta.url);
9 const __dirname = path.dirname(__filename);
10 dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12 // Create Sanity client
13 const client = createClient({
14   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16   useCdn: false,
17   token: process.env.SANITY_API_TOKEN,
18   apiVersion: '2021-08-31'
19 });
20
21
22 Tabnine | Edit | Test | Explain | Document
23 async function uploadImageToSanity(imageUrl) {
24   try {
25     console.log(`Uploading image: ${imageUrl}`);
26     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
27     const buffer = Buffer.from(response.data);
```

## Step 5: Defining the Data Schema

To structure the product data correctly, I navigated to the Sanity project's schema folder and added a custom schema.

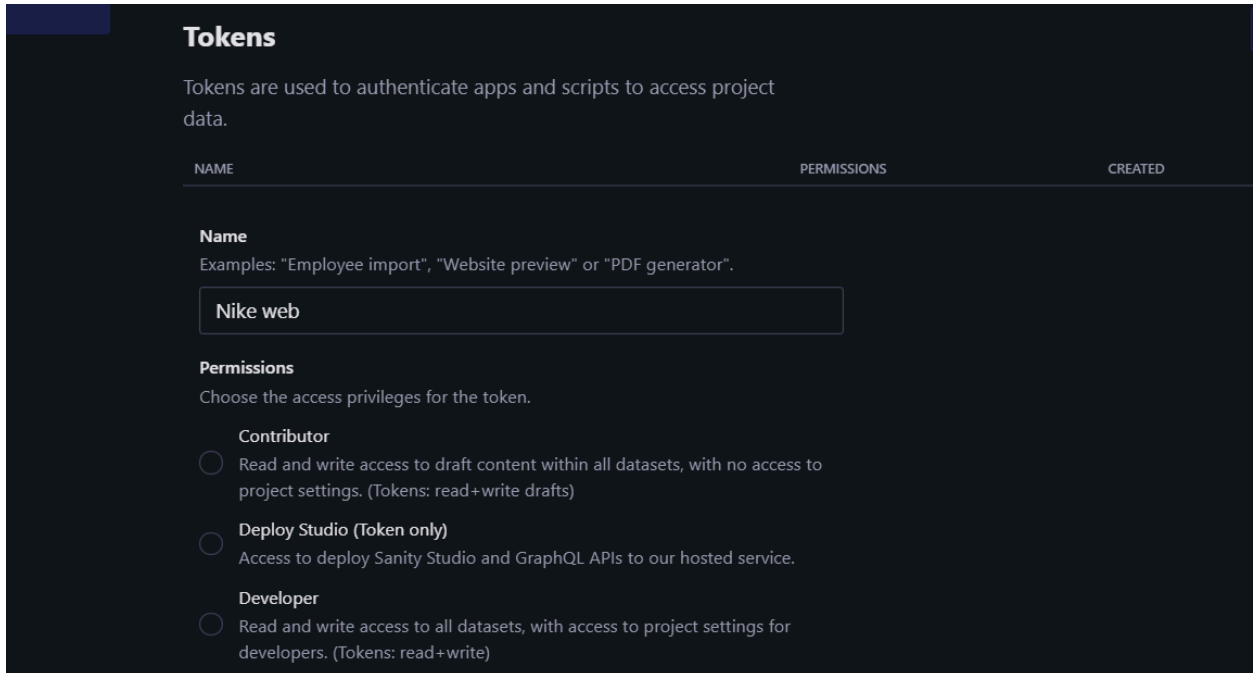
For example, a `products.js` schema was defined as follows:

```
src > sanity > schemaTypes > TS products.ts > [🔗] productSchema
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: 'slug',
13       title: 'Slug',
14       type: 'slug',
15       options: {
16         source: 'productName',
17         maxLength: 96,
18       },
19     },
20     {
21       name: 'category',
22       title: 'Category',
23       type: 'string',
24     },
25     {
26       name: 'price',
27       title: 'Price'
```

## Step 6: Generating an API Token

To enable secure interactions with Sanity, I generated an **authentication token** from the Sanity project settings.

After copying the token, I stored it securely inside the **.env.local** file in my **Next.js** project:



The screenshot shows the 'Tokens' page in the Sanity dashboard. It has a dark theme. At the top, it says 'Tokens' and explains that tokens are used to authenticate apps and scripts. Below this is a table with columns 'NAME', 'PERMISSIONS', and 'CREATED'. Under the 'NAME' column, there's a section titled 'Name' with examples: 'Employee import', 'Website preview', or 'PDF generator'. A text input field contains 'Nike web'. Below that is a section titled 'Permissions' with the instruction 'Choose the access privileges for the token.' There are three radio button options: 'Contributor' (Read and write access to draft content within all datasets, with no access to project settings. (Tokens: read+write drafts)), 'Deploy Studio (Token only)' (Access to deploy Sanity Studio and GraphQL APIs to our hosted service.), and 'Developer' (Read and write access to all datasets, with access to project settings for developers. (Tokens: read+write)).

## Step 7: Executing the Data Migration

With everything set up, I initiated the data migration by running the following command in the terminal:

```
npm run import-data
```

This command executed the script inside **ImportSanityData.mjs**, triggering the data import into Sanity. Within moments, the product data began populating in my Sanity dashboard.

**After running the command the process starts and data starts migrating from api to sanity:**

```
ool]] Image uploaded successfully: image-257e162393cc8f3486965fd649496e3ef9fe23be-348x348-png
Uploading image: https://template-03-api.vercel.app/products/4.png
Image uploaded successfully: image-611df72a5b65db2f78d3119889a742c134480265-348x348-png
Uploading image: https://template-03-api.vercel.app/products/5.png
Image uploaded successfully: image-46998b7aa7bbf7d305755b2964da5bf167434d40-348x348-png
Uploading image: https://template-03-api.vercel.app/products/6.png
Image uploaded successfully: image-0618b5440becd53aced91c6de2bd3c84757be8d1-348x348-png
Uploading image: https://template-03-api.vercel.app/products/7.png
Image uploaded successfully: image-b039b2d23e832f1d29375236a80f613bc011dbc3-348x348-png
Data migrated successfully!
```

## Step 8: Verifying Data in Sanity Studio

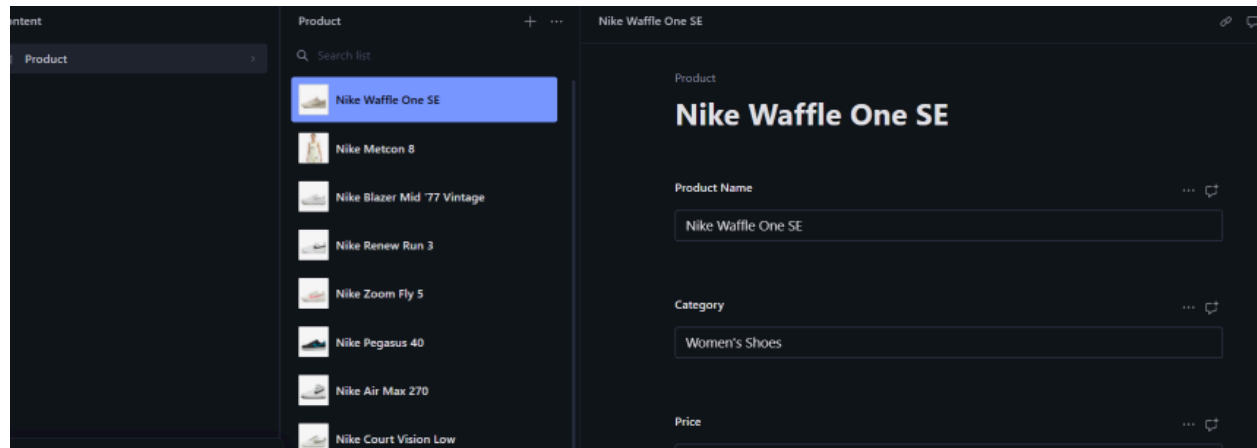
After the migration completed, I started the Next.js development server:

```
npm run dev
```

Then, I navigated to:

<https://localhost:3000/studio>

Here, I could see the successfully migrated product data inside Sanity Studio. 🎉



SAFDAR ALI	00410893	<div><div><div>July</div><div>17</div></div><div>Day: Sunday 2 PM - 5 PM</div></div>
------------	----------	--