

Day-3 API integration

[Nike-general ecommerce marketplace]

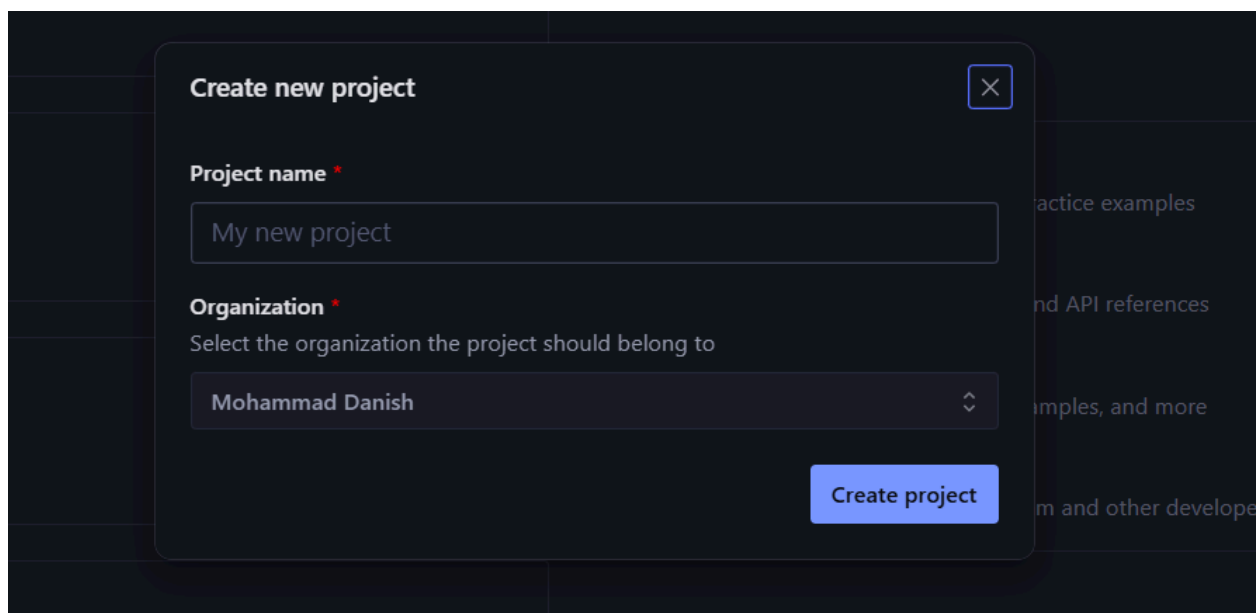
API INTEGRATION PROCESS:

A Step-by-Step Guide to API Integration and Data Migration in Next.js with Sanity

Integrating APIs and managing data migration is a crucial part of modern web development. Recently, I worked on a project involving Sanity, a popular headless CMS, to migrate product data seamlessly. Here's a detailed breakdown of the process I followed:

Step 1: Creating a Project in Sanity

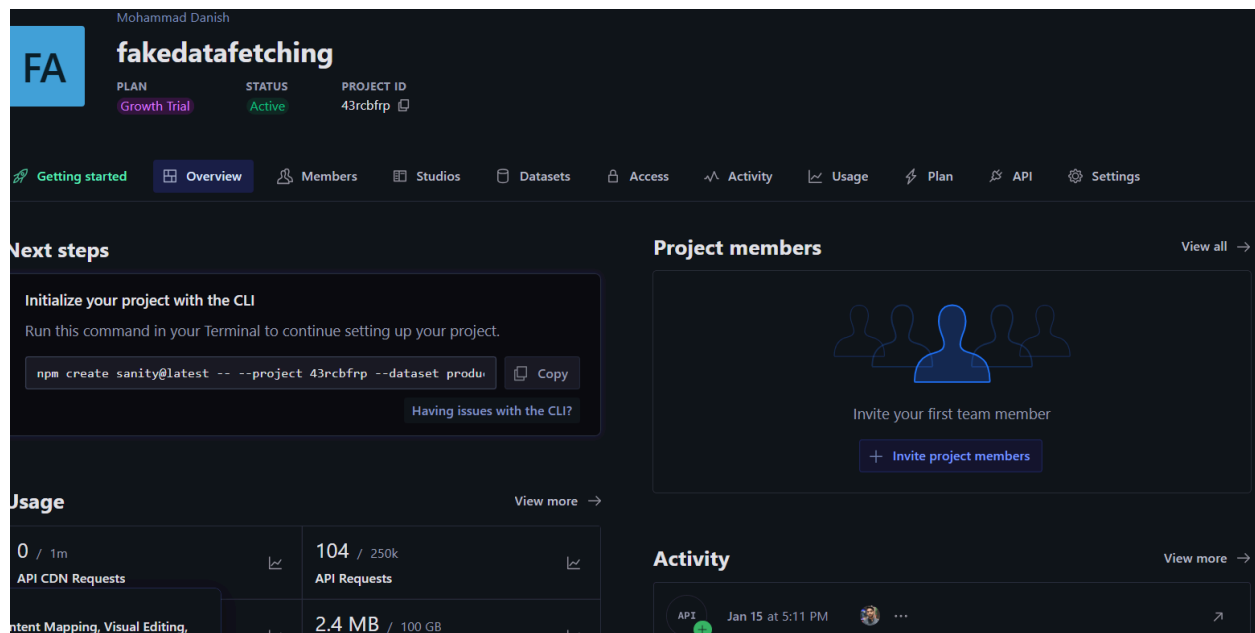
The journey begins by setting up a project in Sanity. I navigated to [Sanity.io](https://sanity.io) and created a new project by following the simple steps in their user-friendly interface.



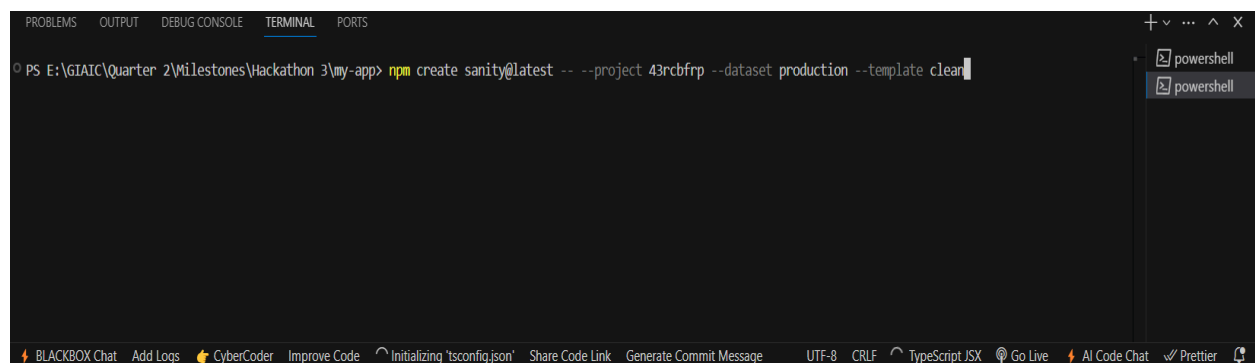
The screenshot shows the 'Create new project' modal in the Sanity.io web interface. The modal has a dark background with light text. It contains two main sections: 'Project name' and 'Organization'. The 'Project name' section has a text input field with the placeholder text 'My new project'. The 'Organization' section has a dropdown menu with the selected organization 'Mohammad Danish'. A blue 'Create project' button is located at the bottom right of the modal. A close button (X) is in the top right corner of the modal. The background of the page is dark with some faint text visible, including 'practice examples', 'nd API references', 'mples, and more', and 'm and other develope'.

Step 2: Running the Command

Once the project was created, Sanity provided a command to set up the required files. I copied this command and ran it in my terminal. This step initialized the Sanity project locally and generated the necessary configuration and schema files.

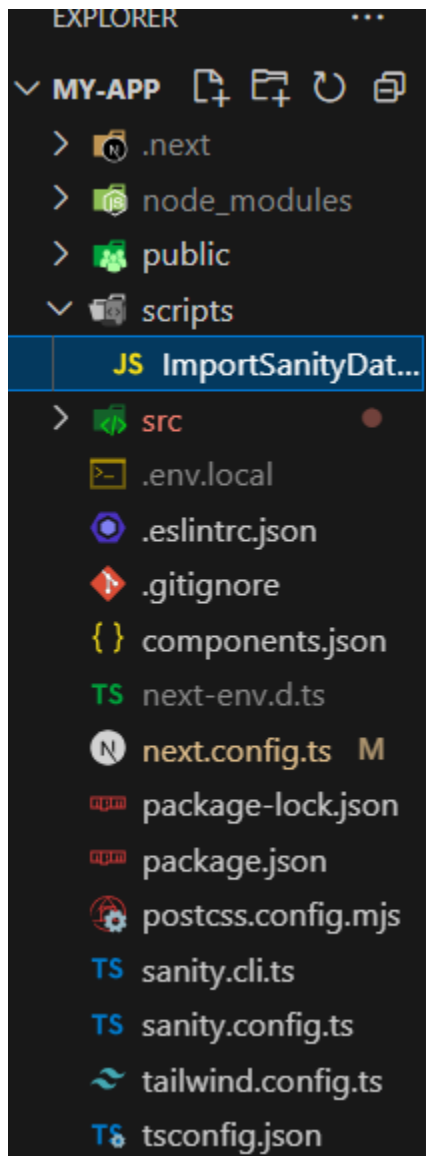


Copy that command and paste it into the terminal:



Step 3: File Structure Setup

After the files were created, I organized my project directory for efficient development. Specifically, I created a folder named `script` outside the `src` folder to handle scripts for data migration. Within this folder, I added a file named `ImportSanityData.mjs`.



Step 4: Adding Migration Code

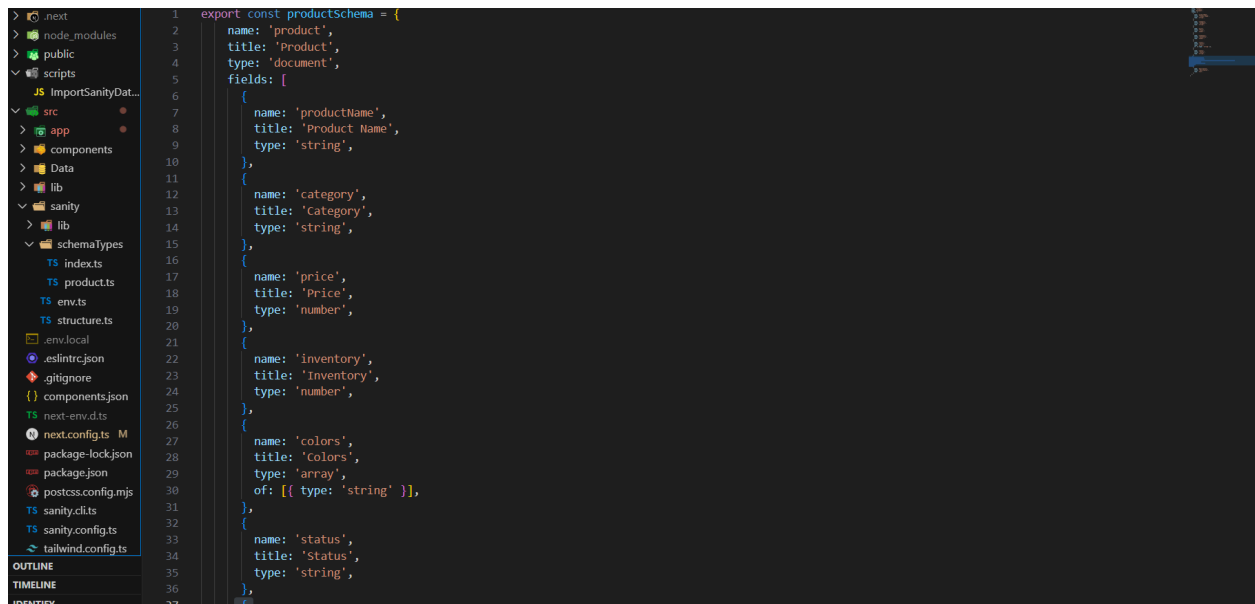
Next, I pasted the migration code provided by Sir Ameen Alam's day-3 documentation into the `ImportSanityData.mjs` file. This code serves as the backbone for importing data into the Sanity project. Additionally, I inserted the required API URL into the script to ensure the connection with Sanity's backend.

```
scripts > JS ImportSanityData.mjs > ...
18   const client = createClient({
19   });
20
21
22   async function uploadImageToSanity(imageUrl) {
23     try {
24       console.log('Uploading image: ${imageUrl}');
25       const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
26       const buffer = Buffer.from(response.data);
27       const asset = await client.assets.upload('image', buffer, {
28         filename: imageUrl.split('/').pop()
29       });
30       console.log('Image uploaded successfully: ${asset._id}');
31       return asset._id;
32     } catch (error) {
33       console.error('Failed to upload image:', imageUrl, error);
34       return null;
35     }
36   }
37
38   async function importData() {
39     try {
40       console.log('migrating data please wait...');
41
42       // API endpoint containing car data
43       const response = await axios.get('https://template-03-api.vercel.app/api/products');
44       const products = response.data.data;
45       console.log("products ==>> ", products);
46
47
48       for (const product of products) {
49         let imageRef = null;
50         if (product.image) {
51           imageRef = await uploadImageToSanity(product.image);
52         }
53
54         const sanityProduct = {
55           name: product.name,
56           price: product.price,
57           description: product.description,
58           imageRef: imageRef,
59         };
60       }
61     } catch (error) {
62       console.error('Error importing data:', error);
63     }
64   }
65
66   importData();
67 }
```

Step 5: Pasting the Schema

After adding the migration code, I navigated to the Sanity project's schema folder and pasted the custom schema for my data structure. This schema defined the

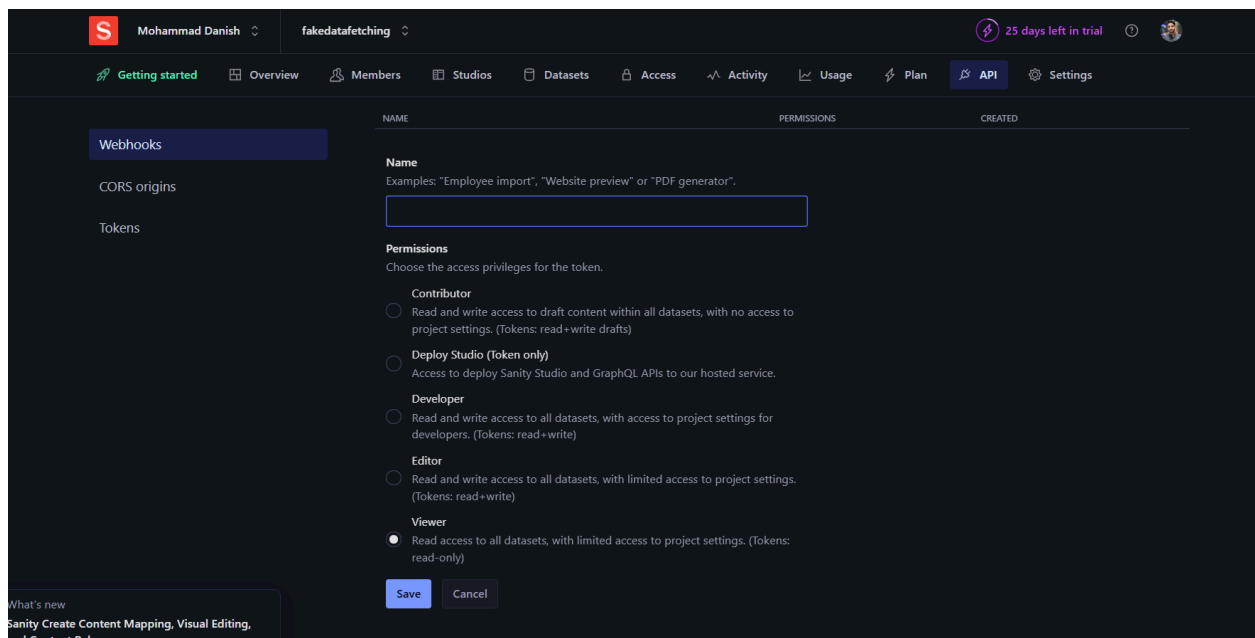
structure and fields of the product data being migrated, ensuring that the imported data aligned perfectly with the CMS.

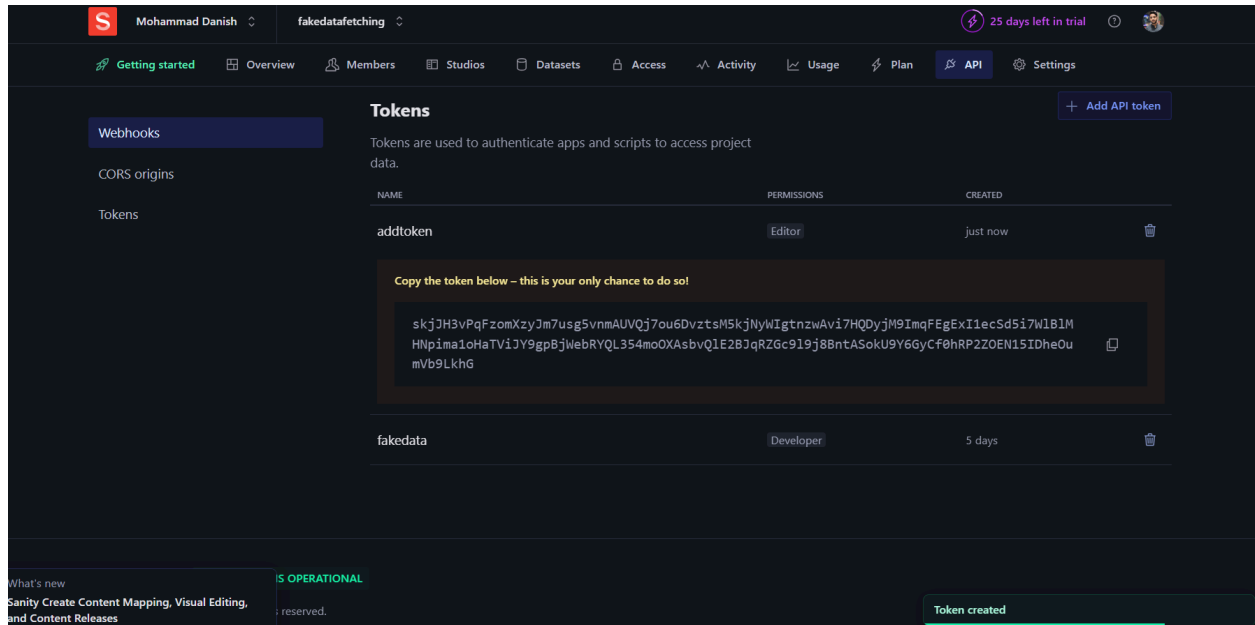


```
1 export const productsSchema = {
2   name: 'product',
3   title: 'Product',
4   type: 'document',
5   fields: [
6     {
7       name: 'productName',
8       title: 'Product Name',
9       type: 'string',
10    },
11    {
12      name: 'category',
13      title: 'Category',
14      type: 'string',
15    },
16    {
17      name: 'price',
18      title: 'Price',
19      type: 'number',
20    },
21    {
22      name: 'inventory',
23      title: 'Inventory',
24      type: 'number',
25    },
26    {
27      name: 'colors',
28      title: 'Colors',
29      type: 'array',
30      of: [{ type: 'string' }],
31    },
32    {
33      name: 'status',
34      title: 'Status',
35      type: 'string',
36    },
37  ],
38 }
```

Step 6: Generating and Adding a Token

To securely interact with the Sanity API, I opened the Sanity project settings and generated an authentication token.





After copying the token,

I added it to my `.env.local` file in the Next.js project. It looked something like this:



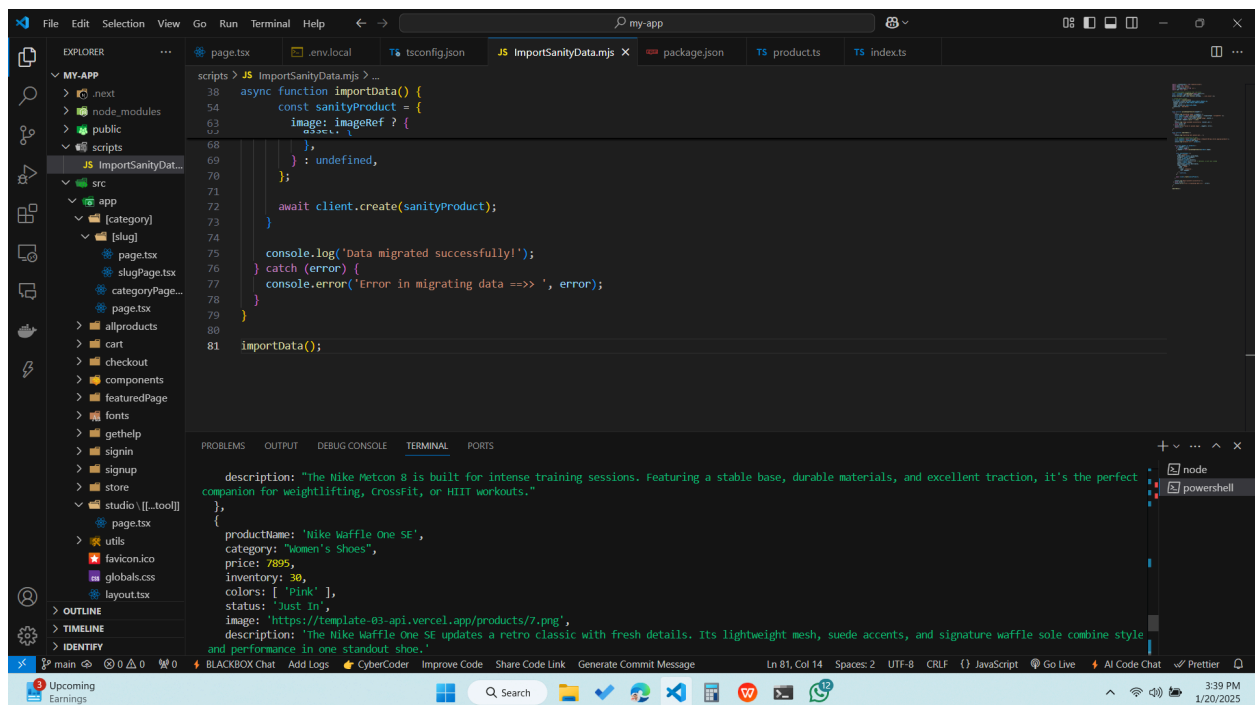
Step 7: Running the Migration Script

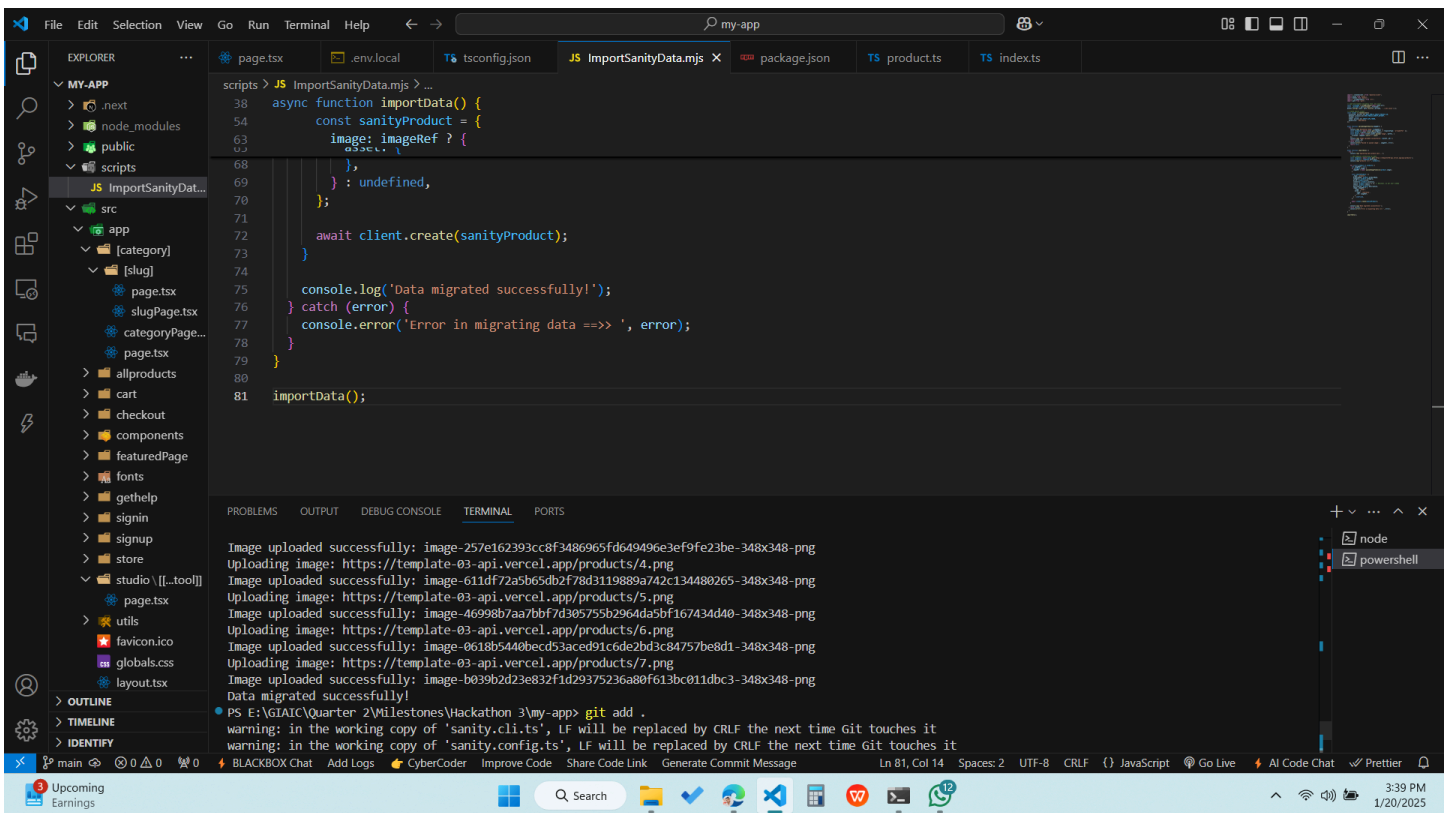
Finally, it was time to start the data migration. I ran the following command in the terminal:

```
npm run import-Data
```

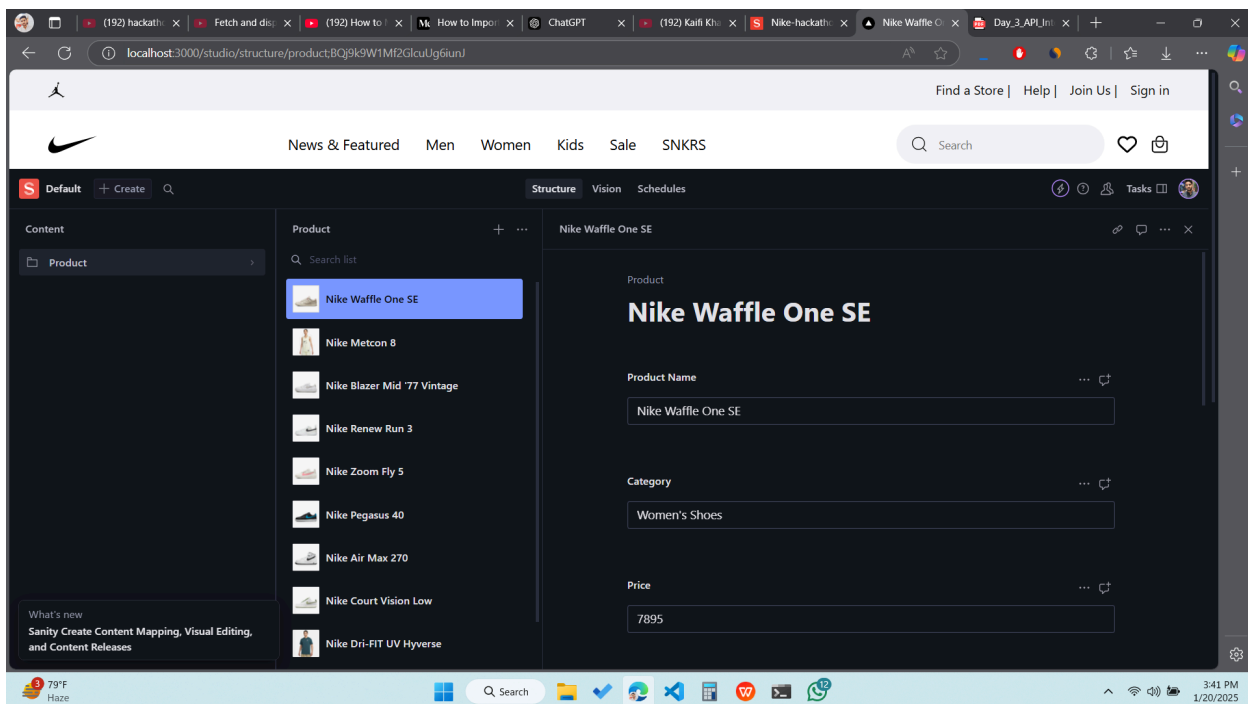
This command executed the script in `ImportSanityData.mjs`, initiating the migration process. The product data began importing into the Sanity project, and soon, the data was visible in my Sanity structure.

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





After Completion of run `npm run dev` and head towards the path [HTTPS://localhost:3000/studio/structure](https://localhost:3000/studio/structure) you will see the data in your sanity



**By: Mohammad Danish,
Day : Saturday 2 to 5,
Id: 00037323,**