

DAY-03

API INTEGRATION AND DATA MIGRATION

[E-Commerce (HELLO NIKE)]

Name: Alishba Shabbir

1. Setting Up Sanity Account:

- Visit [Sanity.io](https://sanity.io) and sign up for an account.
- Create a new project in the Sanity dashboard tailored to your application needs.

2. Generating an API Token:

- Navigate to the **API** section in your project's settings.
- Generate a new API token to enable secure access to your Sanity data.

3. Configuring the Sanity Client:

- Install the Sanity client library in your project.
- Configure the client with your **project ID** and **API token** for seamless communication with the Sanity backend.

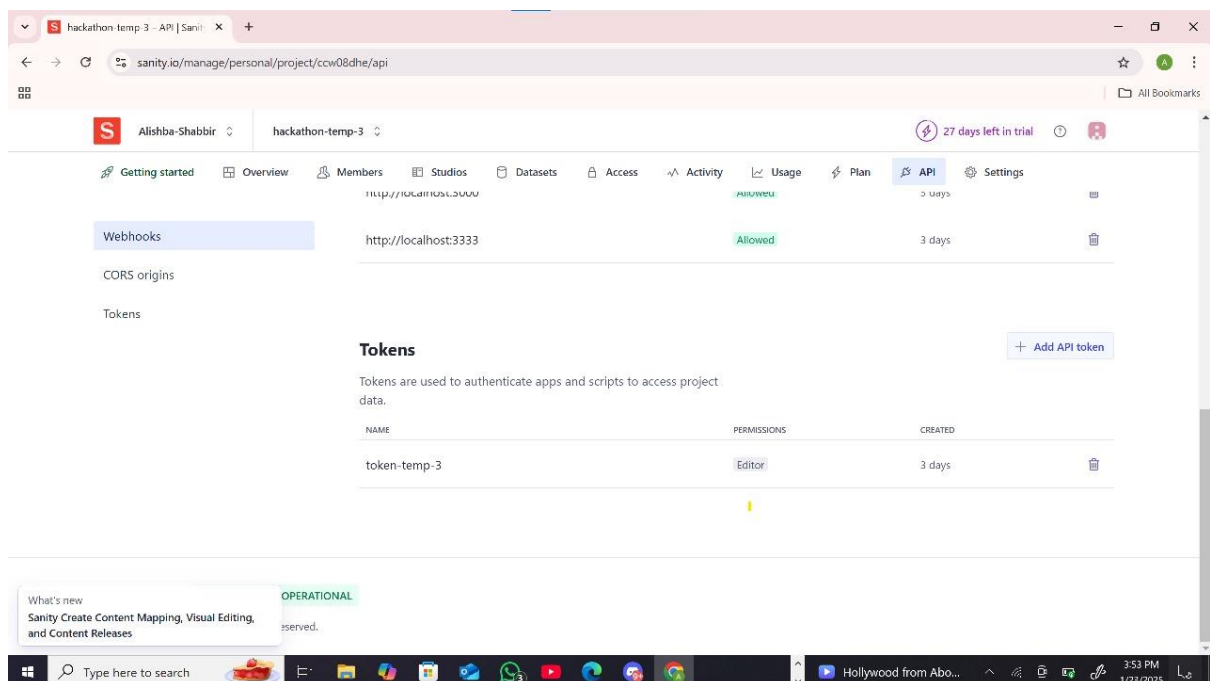
4. Fetching Data:

- Use the configured Sanity client to query and retrieve data like products, orders, or user details.

- Write GROQ queries to fetch specific fields and structure data as required.

5. Security Measures:

- Safeguard your API token by storing it in environment variables to prevent unauthorized access.
- Avoid exposing sensitive credentials in the frontend code



Sanity Integration in My Project

1. Sanity Setup:

- I registered for a Sanity account and generated an API token to facilitate secure communication with the Sanity CMS.

- The Sanity client was installed and configured in my project using the **project ID** and **API token** for seamless data interaction.

2. Using data-migration.mjs:

- I created the data-migration.mjs script to automate the import of product data into the Sanity CMS.
- This script is used to upload product details, such as product names, images, and prices, enabling bulk import and updating of product information at once.

3. Configuring product.ts:

- In product.ts, I defined the structure for product data, specifying key fields like name, description, price, and images.
- The file also handles fetching product data from Sanity by executing queries and ensures type safety through TypeScript.

4. Data Migration Process:

- I executed the data-migration.mjs script to migrate product data into Sanity, simplifying the management and updating of product information within the CMS.

```
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',
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       name: 'image',
39       title: 'Image',
40       type: 'image', // Using Sanity's image type for image field
41       options: {
42         hotspot: true,
43       },
44     },
45     {
46       name: 'description',
47       title: 'Description',
48       type: 'text',
49     },
50   ],
51 }
```

Uploading Images and Data to Sanity

1. Data Upload:

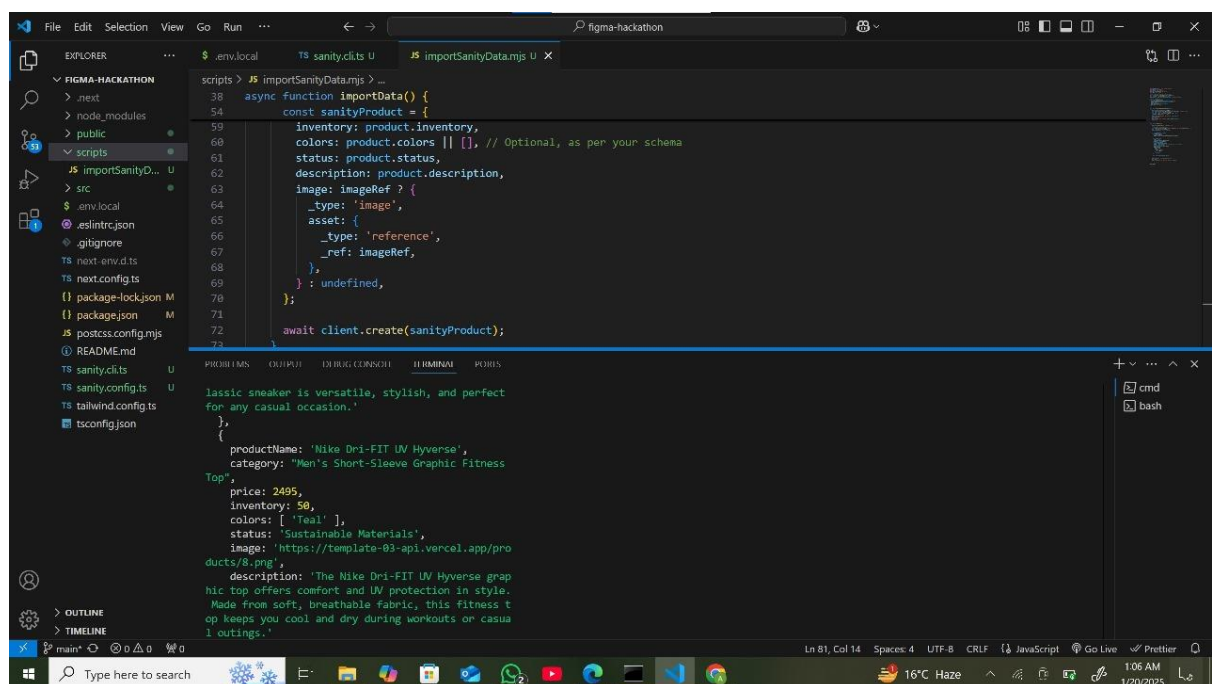
- I successfully imported product information, such as names, descriptions, and pricing, into Sanity via the CMS interface and the data-migration.mjs script.
- The data is organized within Sanity under relevant document types (e.g., product details).

2. Image Upload:

- I uploaded product images to Sanity's asset library, ensuring that each image is linked to its corresponding product.
- The image URLs are now stored and utilized in the product listings displayed on the frontend.

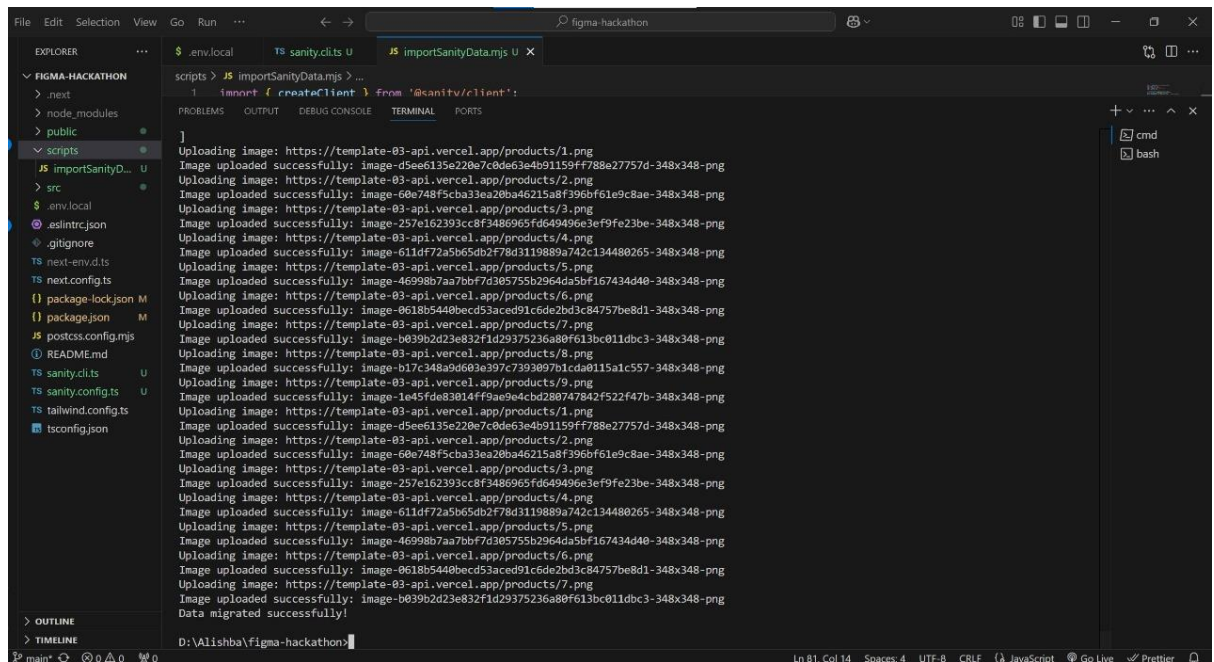
3. Managing Data and Images:

- All product-related data, including images, is now fully manageable and can be updated directly through the Sanity dashboard, streamlining content management.



```
scripts > $ npm run importSanityData.mjs > ...
38   async function importData() {
54     const sanityProduct = {
59       inventory: product.inventory,
60       colors: product.colors || [], // Optional, as per your schema
61       status: product.status,
62       description: product.description,
63       image: imageRef ? {
64         _type: 'image',
65         asset: {
66           _type: 'reference',
67           _ref: imageRef,
68         },
69       } : undefined,
70     };
71   };
72   await client.create(sanityProduct);
73 }
```

```
classic sneaker is versatile, stylish, and perfect
for any casual occasion.'
},
{
  productName: 'Nike Dri-FIT UV Hyverse',
  category: 'Men's Short-Sleeve Graphic Fitness
Top',
  price: 2495,
  inventory: 50,
  colors: [ 'Teal' ],
  status: 'Sustainable Materials',
  image: 'https://template-03-api.vercel.app/pro
ducts/8.png',
  description: 'The Nike Dri-FIT UV Hyverse grap
hic top offers comfort and UV protection in style.
Made from soft, breathable fabric, this fitness t
op keeps you cool and dry during workouts or casua
l outings.'
```



```
scripts > .\importSanityData.mjs > ...
1 import { createClient } from '@sanity/client';
[
  Uploading image: https://template-03-api.vercel.app/products/1.png
  Image uploaded successfully: image-d5ee6135e220e7c8de63e4b91159ff788e27757d-348x348-png
  Uploading image: https://template-03-api.vercel.app/products/2.png
  Image uploaded successfully: image-60e748f5cb33ea20ba46215a8f396bf61e9c8ae-348x348-png
  Uploading image: https://template-03-api.vercel.app/products/3.png
  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-4699807aa7bbf7d308755b2964da5bf167434d40-348x348-png
  Uploading image: https://template-02-api.vercel.app/products/6.png
  Image uploaded successfully: image-8618b5440becd53aced91c6de2bd3c84757be8d1-348x348-png
  Uploading image: https://template-03-api.vercel.app/products/7.png
  Image uploaded successfully: image-b039b2d23e832f1d29375236a80f613bc011dbc3-348x348-png
  Uploading image: https://template-03-api.vercel.app/products/8.png
  Image uploaded successfully: image-b17c348a9d603e397c7393097b1cda0115a1c557-348x348-png
  Uploading image: https://template-03-api.vercel.app/products/9.png
  Image uploaded successfully: image-1e45fde83014ff9ae9e4cbd280747842f522f47b-348x348-png
  Uploading image: https://template-03-api.vercel.app/products/1.png
  Image uploaded successfully: image-d5ee6135e220e7c8de63e4b91159ff788e27757d-348x348-png
  Uploading image: https://template-02-api.vercel.app/products/2.png
  Image uploaded successfully: image-60e748f5cb33ea20ba46215a8f396bf61e9c8ae-348x348-png
  Uploading image: https://template-03-api.vercel.app/products/3.png
  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-4699807aa7bbf7d308755b2964da5bf167434d40-348x348-png
  Uploading image: https://template-03-api.vercel.app/products/6.png
  Image uploaded successfully: image-8618b5440becd53aced91c6de2bd3c84757be8d1-348x348-png
  Uploading image: https://template-02-api.vercel.app/products/7.png
  Image uploaded successfully: image-b039b2d23e832f1d29375236a80f613bc011dbc3-348x348-png
  Data migrated successfully!
D:\Alishba\figma-hackathon>
```

Data Fetching from Sanity

Data was successfully fetched from Sanity CMS using GROQ (Graph-Relational Object Queries). The fetched data is dynamically displayed on the website, ensuring seamless integration between the backend and frontend.

