Amna Aftab

483865

Friday(9am- 12pm)

Hackathon-2025 (Day 3)

Template-5 Bandage

Explanation of Migrating Data from API to Sanity Using Next.js

In this project, I implemented a solution that allows us to migrate and import data from an external API into Sanity CMS. This process involves fetching data in a Next.js application, transforming it as needed, and then sending it to Sanity via its API. Below is a step-by-step explanation of how I achieved this.

1. Fetching Data from an External API

The first step in the process was to fetch data from an external API. In my Next.js application, I used the utility function to fetch the data from the API at runtime.

And checked data by using Thunderclient by **GET** method to fetch all responses and console the return to verify the result.

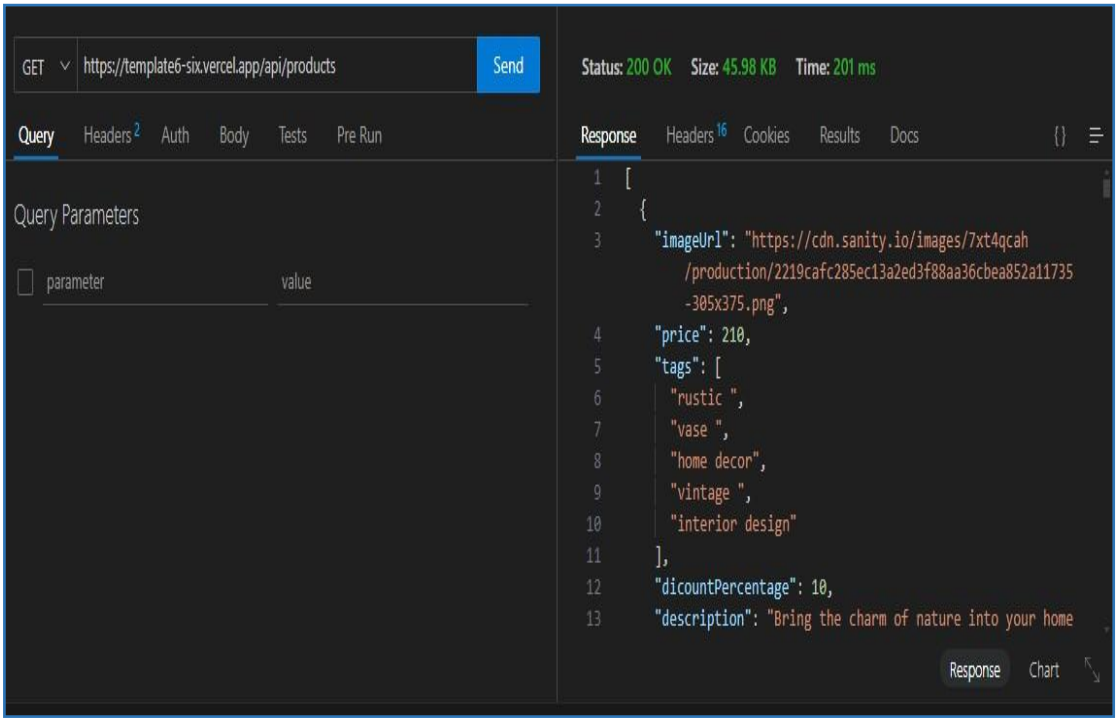
```
async function importProducts() {
  try {
    const response = await fetch('https://template6-six.vercel.app/api/products');

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error('Error fetching products:', error);
  }
}

importProducts();
```



2. Comparing the API Data with Sanity Schema

Once the data was fetched, the next task was to compare the structure of the API data with the **Sanity CMS schema**. The Sanity schema

defines the structure of the content that will be stored in the CMS. In this case, a product schema was defined to handle the product data, which included fields such as:

Name (String)
Price (Number)
Description (Text)
Image (Image)
Tags(Array)
Discount(Number)

The comparison was performed to ensure that the data fetched from the API adhered to the structure defined in the Sanity schema. This step ensured that each field in the API data matched a corresponding field in the Sanity schema.

3. Writing the Migration Script

After confirming the schema and the API data structure, the next step was to write a migration script that would automate the process of importing the fetched data into Sanity. A script was created that would:

1. Fetch data from the API.
2. Format the data according to the Sanity schema.
3. Use the **Sanity Client** to push the data into Sanity.

The migration script **(migration.mjs)** was created as follows:

```
async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      filename: imageUrl.split('/').pop(),
    });

    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

async function uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: 'product',
        title: product.title,
```

```
        price: product.price,
        productImage: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        tags: product.tags,
        dicountPercentage: product.dicountPercentage, // Typo in field name: dicountPerc
        description: product.description,
        isNew: product.isNew,
      };

      const createdProduct = await client.create(document);
      console.log(`Product ${product.title} uploaded successfully:`, createdProduct);
    } else {
      console.log(`Product ${product.title} skipped due to image upload failure.`);
    }
  } catch (error) {
    console.error('Error uploading product:', error);
  }
}
```

4. Setting Up the Environment

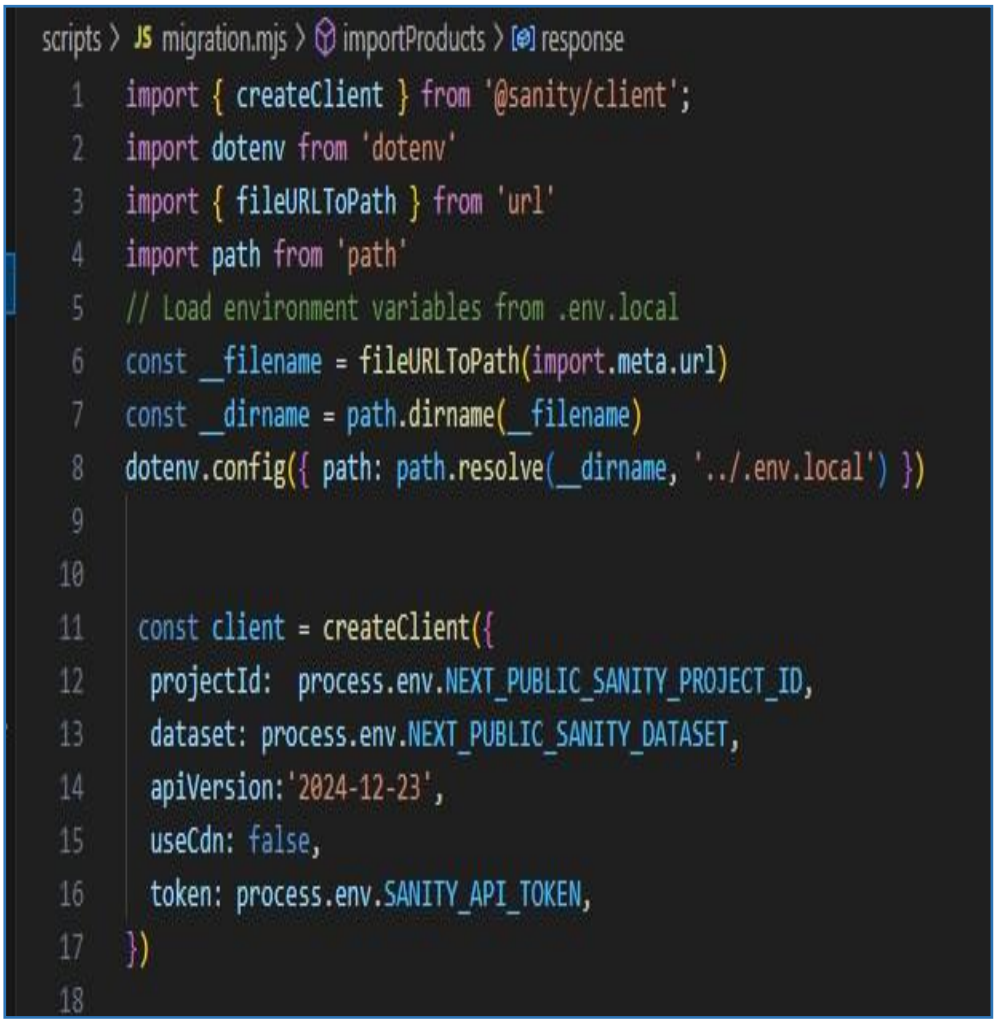
Once the script was created, the environment required for running the migration script was set up. This involved configuring the **Sanity Client** by providing necessary credentials, such as the **Sanity Project ID**, **Dataset**, and **Auth Token** (if necessary).

Installed dotenv:

Command:

npm install dotenv

To set path env.local file to migration.mjs file.



5. Modifying package.json

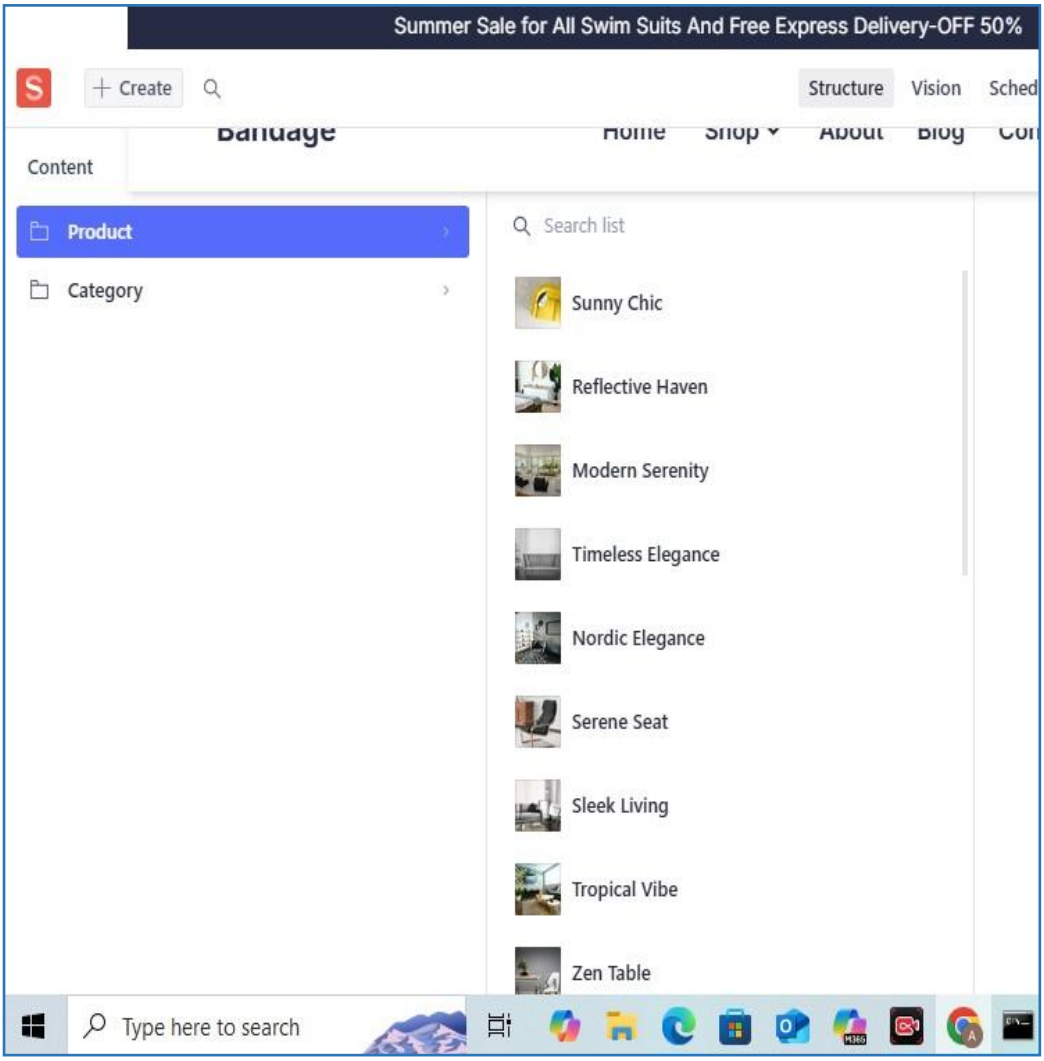
To run the migration script, the necessary npm package was added to the package.json file under the scripts section. This made it easier to run the script through the command line.

```
"migrate": "ts-node scripts/migration.mjs"
```

6. Importing Data into Sanity

With everything set up, I executed the migration script by running the following command:

npm run import-data



Conclusion

This process successfully automated the migration of data from an external API to Sanity CMS. The key steps included:

1. **Fetching the data** from the API using Next.js.
2. **Mapping and transforming** the API data to match the Sanity schema.
3. **Writing a migration script** to import the data into Sanity.
4. **Configuring the environment** and running the script via package.json.

Day-3 Checklist

API understanding	√
Schema Validation	√
Data Migration	√
API Integration in Next.Js	√
Submission Preparation	√