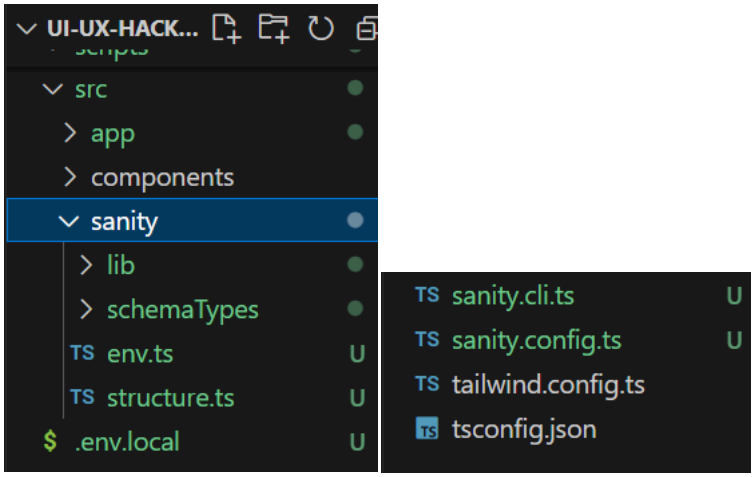
HACKATHON 3 (Day 3) - API Integration

This documentation outlines the process of integrating API data into Sanity CMS for Template 6, a furniture e-commerce platform. Sanity CMS was set up through sanity.io and connected to the local project, where the required schema was created. Data from the provided API was imported, validated, and successfully integrated into Sanity CMS.

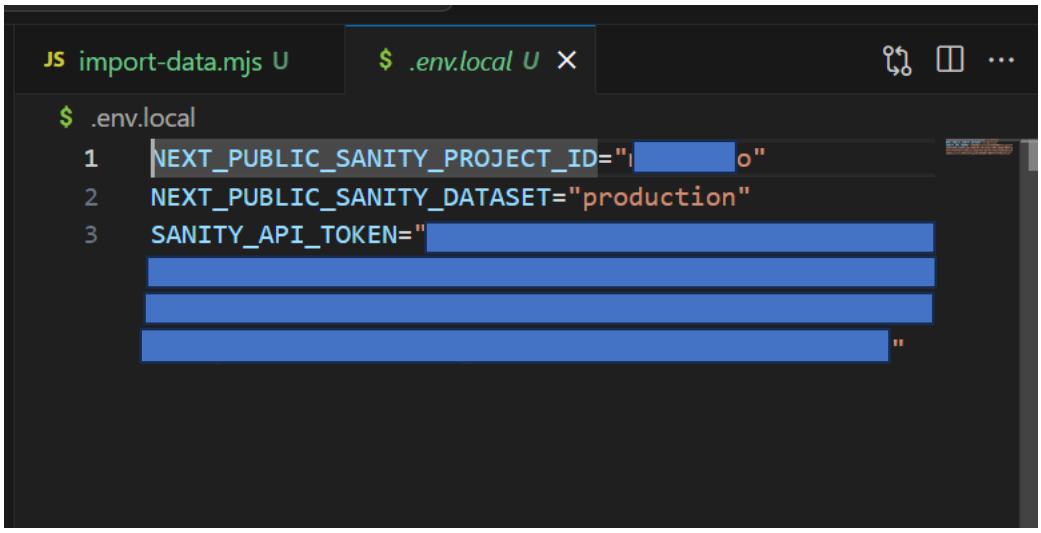
1. Creating and Connecting Sanity CMS

I created a Sanity CMS project on sanity.io and connected it to my local Next.js project. During the setup, I initialized the project locally using the Sanity CLI, which created a sanity folder with all the necessary configuration files.



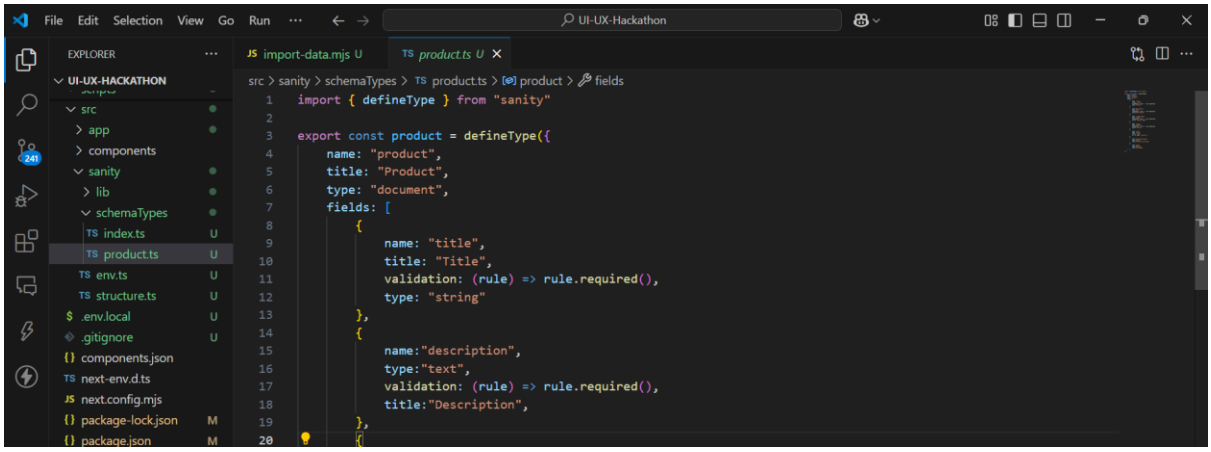
2. Adding the API Token

To securely access the Sanity CMS project, generated an API token on the Sanity dashboard. Then added the **Project ID**, **Dataset Name**, and **API Token** to the .env.local file for secure access:



3. Creating the Schema

Added the provided product.ts schema for Template 6 and added it to the sanity/schemaTypes folder.



4. Data Migration Script

To automate the migration of data from the API to Sanity, created a script named **import-data.mjs** in the scripts folder. The script fetched the data from the provided API and imported it into Sanity CMS.

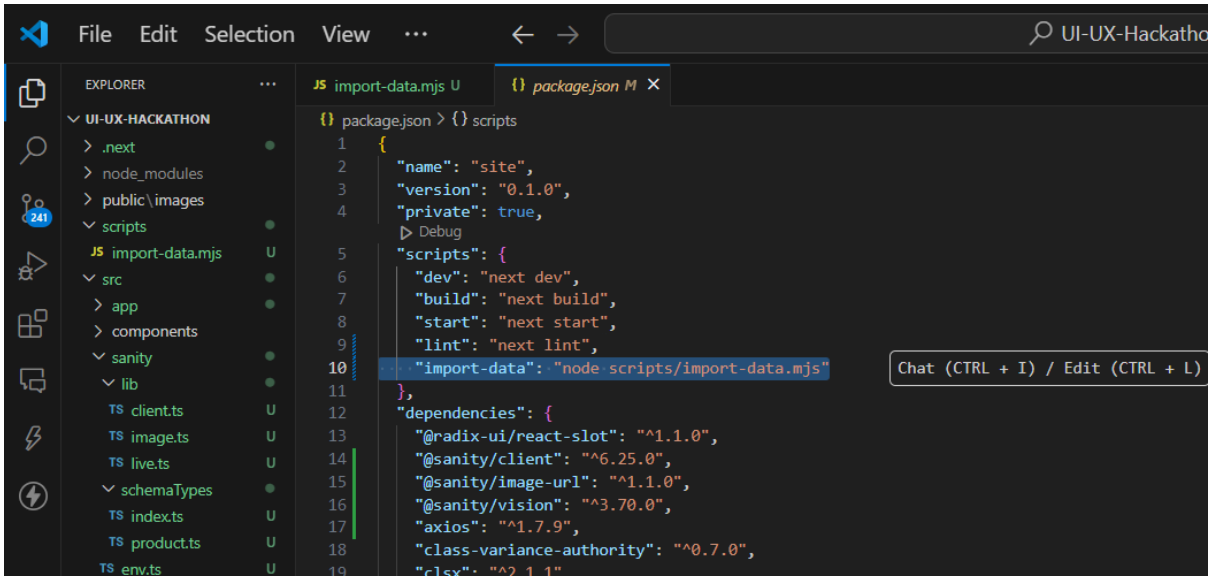
```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31',
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
```

```
    const asset = await client.assets.upload('image', buffer, {
      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 {
    console.log(`Uploading product: ${product.title}`);

    const imageId = product.imageUrl ? await uploadImageToSanity(product.imageUrl) : null;

    const document = {
      _type: 'product',
      title: product.title,
      price: product.price,
      productImage: imageId
        ? {
            _type: 'image',
            asset: {
              _type: 'reference',
              _ref: imageId,
            },
          }
        : undefined,
      tags: product.tags || [],
      discountPercentage: product.discountPercentage, // Fixed typo
      description: product.description,
      isNew: product.isNew,
    };

    const createdProduct = await client.create(document);
    console.log(`Product "${product.title}" uploaded successfully:`, createdProduct);
  } catch (error) {
    console.error(`Error uploading product "${product.title}":`, error);
  }
}

async function importProducts() {
  try {
    console.log('Fetching products...');
    const response = await axios.get('https://template6-six.vercel.app/api/products');
    if (response.status !== 200) {
```

```
      throw new Error(`Failed to fetch products: HTTP ${response.status}`);
    }
    const products = response.data;

    for (const product of products) {
      await uploadProduct(product);
    }

    console.log('All products uploaded successfully!');
  } catch (error) {
    console.error('Error during product import:', error);
  }
}

importProducts();
```

5. Running the Migration Script

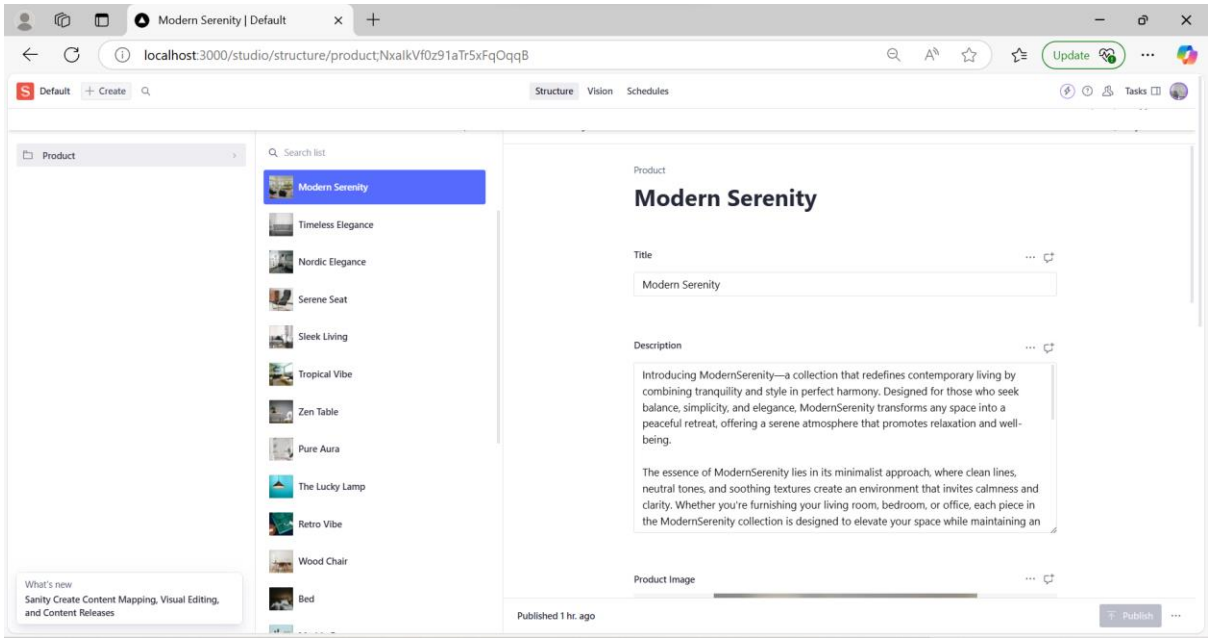
Added a new script to package.json to simplify the execution of the migration:



Then ran the script using: **npm run import-data**

Results

1. Populated Sanity CMS



2. Console Logs