

Day 3 - API Integration and Data Migration for Hekto Ecommerce

Template 4

NAME: Zakia Bashir

Roll No: 141704

I created my own API and migrated data to Sanity using Next.js

. In this e-commerce website, I implemented a solution that migrates and imports data from my Mocki API into Sanity CMS using Next.js. This process involves fetching data in a Next.js application, transforming it as needed, and sending it to Sanity through its API.

| Below is a step-by-step explanation of how I achieved this.

1. Fetching Data from My Mocki API

The first step in the process was fetching data from my Mocki API. In my Next.js application, I used the GET method to fetch all the responses and checked the results in the console.

javascript

```
async function importData() {  
  try {  
    console.log('Fetching products from API...');  
    const response = await axios.get('https://mocki.io/v1/8df2c1');  
    const products = response.data;  
    console.log(`Fetched ${products.length} products`);  
  } catch (error) {
```

```
    console.error('Error fetching data:', error);  
  }  
}
```

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:

json

```
[  
  {  
    "id": 1,  
    "productName": "Ergonomic Concrete Chair",  
    "productDescription": "The Ergonomic Concrete Chair is designed for  
comfort and durability.",  
    "price": "199.99",  
    "prevPrice": "249.99",  
    "stock": 45,  
    "productImage":  
    "https://res.cloudinary.com/dwd9h8qgy/image/upload/v1736416576/ima  
ge_32_1_dy4j9j.png",  
    "tag": "Featured Products",  
    "shipmentArray": [  
      {
```

```
"trackingId": "123e4567-e89b-12d3-a456-426614174000",
"deliveryStatus": "Shipped",
"estimatedDeliveryDate": "2023-11-15"
}
]
}
]
javascript
export default {
  name: 'products',
  title: 'Products',
  type: 'document',
  fields: [
    { name: 'id', title: 'ID', type: 'number' },
    { name: 'productName', title: 'Product Name', type: 'string' },
    { name: 'productDescription', title: 'Product Description', type: 'text' },
    { name: 'price', title: 'Price', type: 'string' },
    { name: 'prevPrice', title: 'Previous Price', type: 'string' },
    { name: 'stock', title: 'Stock', type: 'number' },
    { name: 'productImage', title: 'Product Image', type: 'image', options: {
      hotspot: true } },
    {
      name: 'tag',
```

```

    title: 'Tag',
    type: 'string',
    options: {
      list: [
        { title: 'Featured Products', value: 'Featured Products' },
        { title: 'Trending Products', value: 'Trending Products' },
        { title: 'Latest Products', value: 'Latest Products' },
        { title: 'Top Categories', value: 'Top Categories' }
      ]
    }
  },
  { name: 'shipmentArray', title: 'Shipment Array', type: 'array', of: [{
    type: 'shipment' }] }
]
};

```

I compared the data structure between the API and Sanity schema to ensure alignment. This verification step confirmed that each API data field had a matching field in the Sanity schema structure.

3. Creating and Implementing a Comprehensive Data Migration Script

After confirming the schema and 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:

- Fetch data from the API
- Format the data according to the Sanity schema

- Use the Sanity Client to push the data into Sanity

The migration script was created as follows:

```
javascript
```

```
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 importData() {  
  try {  
    console.log('Fetching products from API...');  
    const response = await axios.get('https://mocki.io/v1/8df2c1');  
    const products = response.data;  
    console.log(`Fetched ${products.length} products`);  
  }  
}
```

```
for (const product of products) {  
  console.log(`Processing product: ${product.productName}`);  
  let imageRef = null;  
  if (product.productImage) {  
    imageRef = await uploadImageToSanity(product.productImage);  
  }  
  const sanityProduct = {  
    _type: 'products',  
    id: product.id,  
    productName: product.productName,  
    productDescription: product.productDescription,  
    price: product.price,  
    prevPrice: product.prevPrice,  
    stock: product.stock,  
    productImage: imageRef ? { _type: 'image', asset: { _type:  
'reference', _ref: imageRef } } : undefined,  
    tag: product.tag,  
    shipmentArray: product.shipmentArray.map(shipment => ({  
      _type: 'shipment',  
      trackingId: shipment.trackingId,  
      deliveryStatus: shipment.deliveryStatus,  
      estimatedDeliveryDate: shipment.estimatedDeliveryDate,  
    })),  
  }  
}
```

```

    };
    console.log('Uploading product to Sanity:', sanityProduct);
    const result = await client.create(sanityProduct);
    console.log(`Product uploaded successfully: ${result._id}`);
  }
  console.log('Data import completed successfully!');
} catch (error) {
  console.error('Error importing data:', error);
}
}

```

```
importData();
```

4. Setting Up the Environment Variables in .env File

These environment variables are very important because they store sensitive information securely and keep it separate from the codebase. It enhances security.

```
plaintext
```

```
NEXT_PUBLIC_SANITY_PROJECT_ID=""
```

```
NEXT_PUBLIC_SANITY_DATASET=""
```

```
SANITY_API_TOKEN=""
```

5. Install dotenv and Set Path

Command:

```
bash
```


npm install dotenv

Installing dotenv is essential when you need to set and manage environment variables in a project. It enables you to load these variables from a .env file into your application's process.env.

javascript

```
import { createClient } from '@sanity/client';
```

```
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: '2025-01-16',  
});
```

6. Modifying package.json

json

```
"import-data": "node scripts/importSanityData.mjs"
```

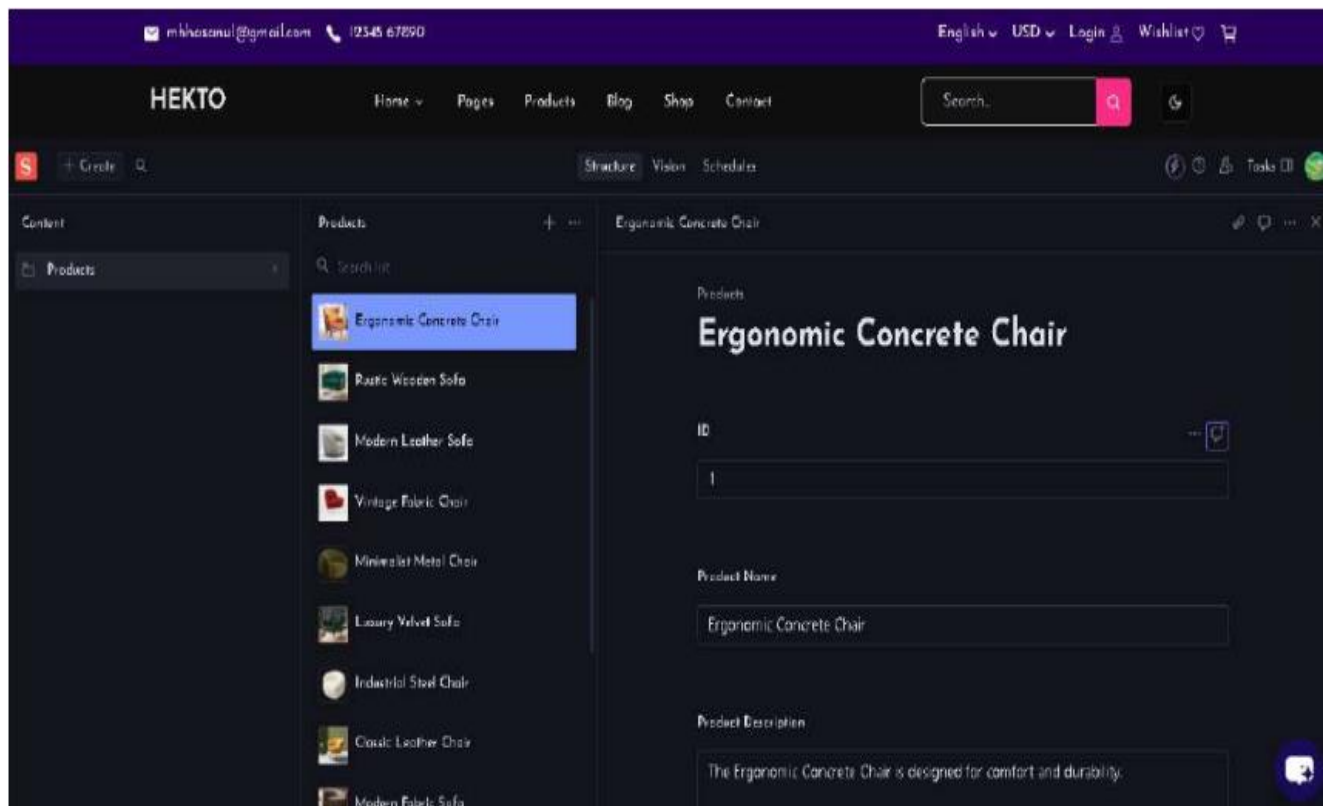
7. Run Command to Import Data to Sanity

With everything properly set up, I successfully executed the migration script using the following command:

```
bash
```

```
npm run import-data
```

As a result, all the data was seamlessly imported into Sanity, one entry at a time.



Conclusion

In this process, we successfully automated the seamless migration of data from an external API into Sanity CMS. The key steps involved:

1. Fetching data from the API
2. Mapping and transforming the data to align with the Sanity schema
3. Writing a migration script to efficiently import the data into Sanity
4. Configuring the necessary environment and executing the script via `package.json`
5. Running the command to import data to Sanity

This automation made everything easier and kept the data safe and consistent, so the move to Sanity was quick and simple.