

## Day 3 - API Integration Report – Mens Apparel

### Introduction

This report outlines the process of integrating APIs, making necessary adjustments to the schemas, and migrating data to the Sanity CMS for **Mens Apparel**. The goal was to enable seamless API interaction, integrate external product data, and ensure the CMS is populated and functional.

### API Integration Process

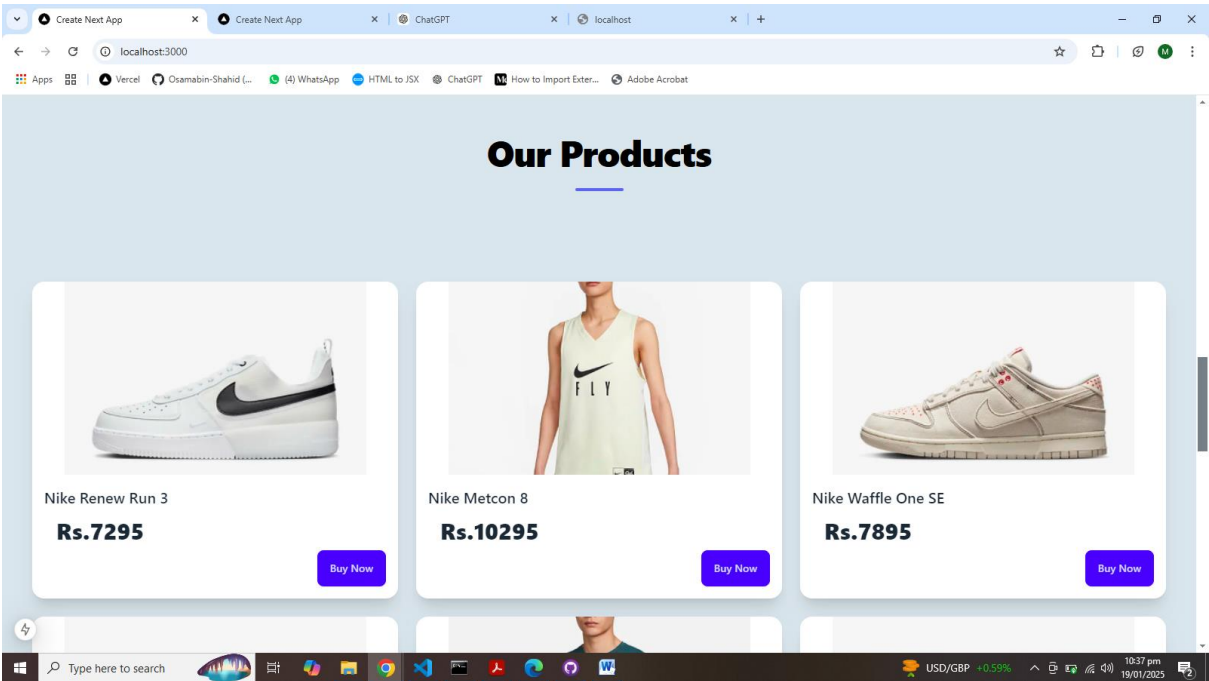
#### 1. API Integration

The API integration involved fetching product data from an external source and populating it in Sanity CMS. The following steps were followed:

1. Identifying the API Endpoints:  
The external API that provides product data was identified, including endpoints for product information,
2. Creating API Fetch Functions:  
We created a fetch function using `fetch()` or an alternative like `axios` to retrieve product data from the external API.
3. Integrating API Calls in the Project:  
API calls were made during the data fetching process in Next.js. The API was connected to the frontend and used in pages like Product Detail.

#### Frontend Display of API Data:

- Once the data is fetched, it is displayed in the frontend. For example, on the Product Detail page, we display the product details like title, price, description, and inventory status fetched from the API.



### Handling Data:

- Upon successful API calls, product data was parsed and displayed in the frontend. The product details were rendered on the Product Detail page.

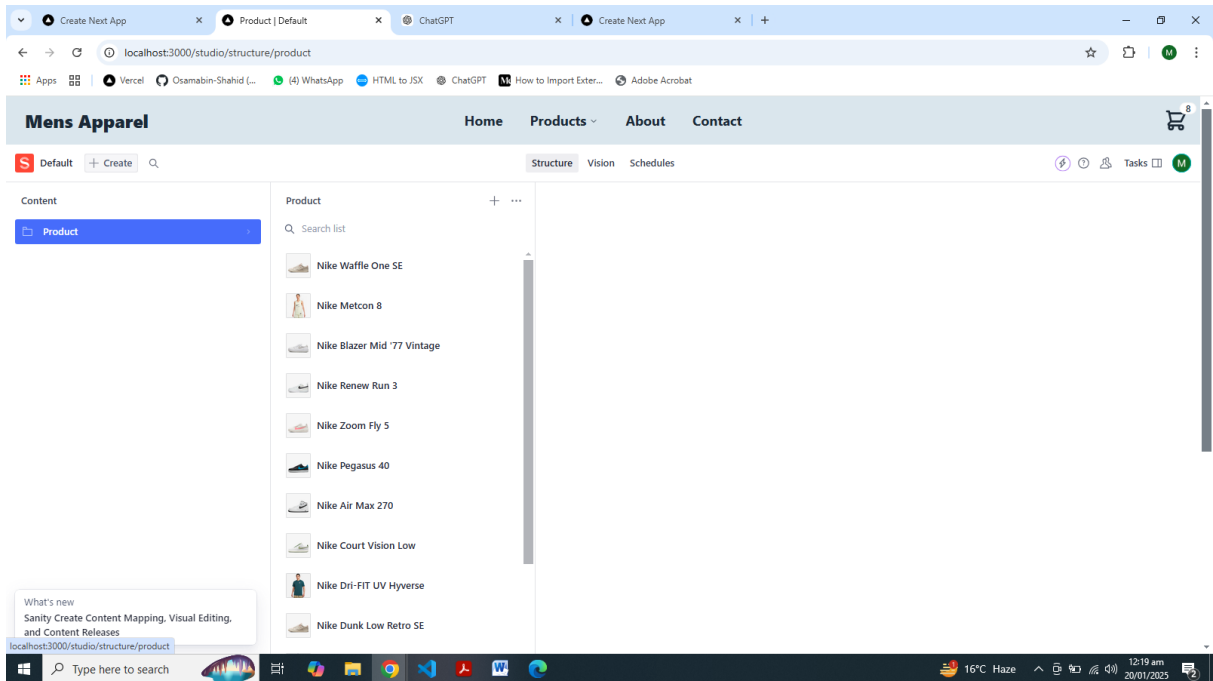
### Error Handling:

- A fallback mechanism was implemented to handle any errors during the API fetch process (e.g., when the product isn't found).

### Product Schema:



```
src > sanity > schemaTypes > TS products.ts > [0] productSchema
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',
```



## Migration Steps and Tools Used

### Data Migration

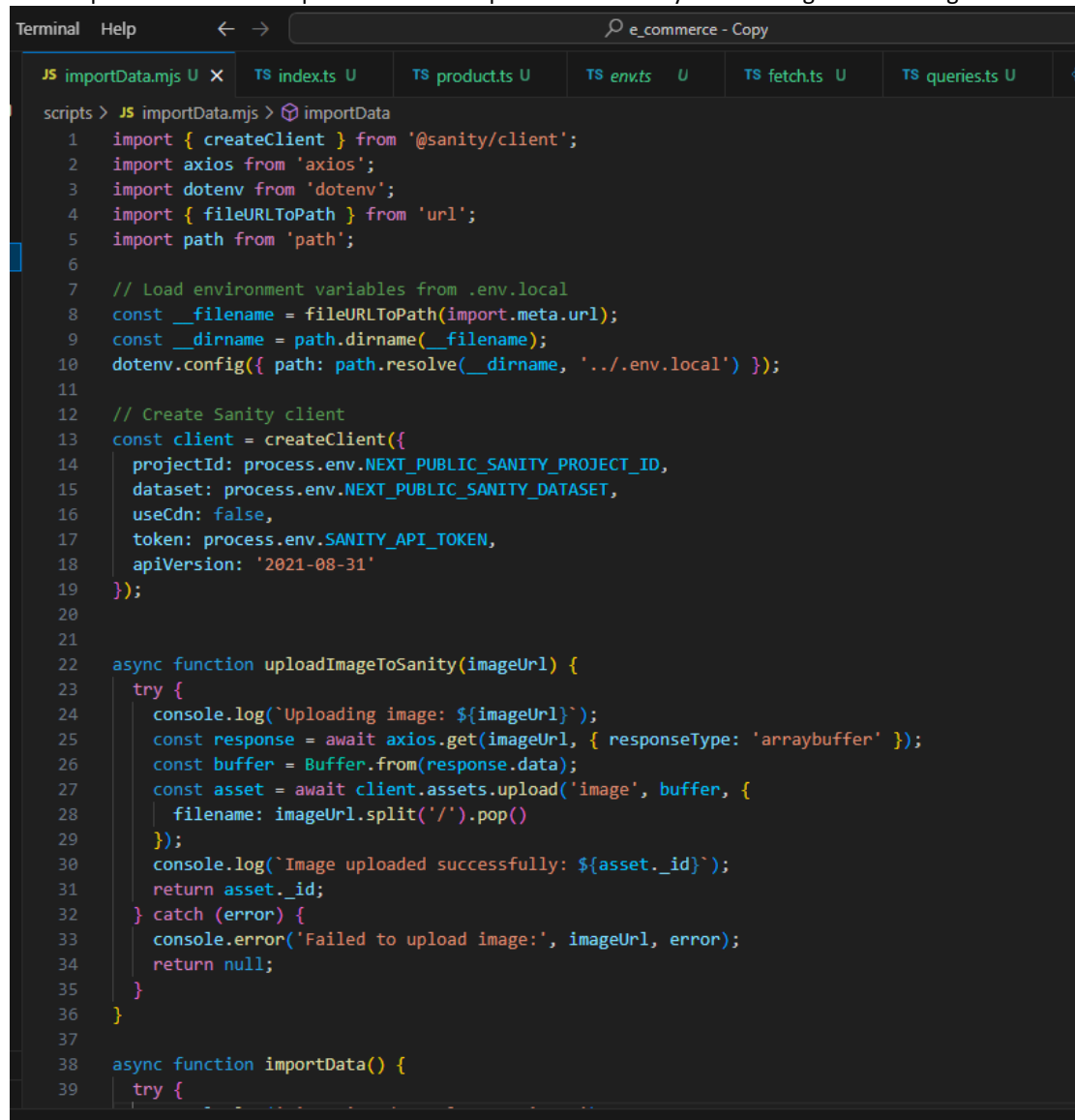
The product data was migrated from the external API into the Sanity CMS using the following steps:

#### Preparing the Migration Script:

A Node.js script was written to automate the migration process. This script used Sanity's client to create documents in the CMS with data from the external API.

#### Script to Migrate Product Data:

The script iterated over the product data and pushed it to Sanity's CMS using the following code:



```
Terminal  Help  ← →  e_commerce - Copy

JS importData.mjs U X  TS index.ts U  TS product.ts U  TS env.ts U  TS fetch.ts U  TS queries.ts U

scripts > JS importData.mjs > importData
1  import { createClient } from '@sanity/client';
2  import axios from 'axios';
3  import dotenv from 'dotenv';
4  import { fileURLToPath } from 'url';
5  import path from 'path';
6
7  // Load environment variables from .env.local
8  const __filename = fileURLToPath(import.meta.url);
9  const __dirname = path.dirname(__filename);
10  dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12  // Create Sanity client
13  const client = createClient({
14    projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15    dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16    useCdn: false,
17    token: process.env.SANITY_API_TOKEN,
18    apiVersion: '2021-08-31'
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 {
```

```
importData.mjs  X  15 indexes  0  15 products  0  15 envs  0  15 fetches  0  15 queries
scripts > JS importData.mjs > importData
38  async function importData() {
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              _type: 'product',
56              productName: product.productName,
57              category: product.category,
58              price: product.price,
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      }
74
75      console.log('Data migrated successfully!');
76  } catch (error) {
77      console.error('Error in migrating data ==>> ', error);
78  }
79  }
80
81  importData();
```

## Sanity Studio

Sanity Studio was used to verify that the data was correctly migrated and populated. We could check the product documents and their fields to ensure that all relevant information was stored.

### Verification:

After migration, the data was verified in the Sanity Studio interface, ensuring that all product data, including pricing, description, and categories, were correctly added.

## Conclusion

The API integration process was successfully completed, with the product data being fetched, stored, and displayed on the frontend. Necessary adjustments were made to the Sanity CMS schema to accommodate the new product information. Data migration was automated using a custom script, ensuring a smooth transition from the external API to the CMS.