By Ubaid ur rahman
0063553

API Integration Report - Nike Store

Name: Ubaid ur rahman

Roll No: 0063553

Project: API Integration and Data Migration

Overview

The focus of Day 3 was to integrate APIs, migrate data into Sanity CMS, and ensure the frontend dynamically

displays product information. This report outlines the steps taken to achieve these objectives, including API

integration, schema updates, and data migration.

1. Reviewed API Documentation

• Objective: Understand the structure and and functionality of the provided API.

• Steps Taken:

o Reviewed the API documentation for the assigned template.

o Identified key endpoints (e.g., /products) and the *structure of the returned data.*

o Noted field names and data types to ensure compatibility with the Sanity schema.

2. Set Up API Calls

• Objective: Fetch data from the API and prepare it for migration.

• Steps Taken:

o Used Thunder Client to test the API endpoint and verify the data structure.

o Created utility functions in the Next.js project to fetch data using fetch or Axios.

o Logged API responses in the console to validate the data structure.

4. Data Migration to Sanity CMS

• Objective: Migrate product data (including images) from the API to Sanity CMS
• Steps Taken:

o Created a migration script (importData.mjs) to fetch data from the API and transform it into the *required format.*

o Used the Sanity client library to upload data and images to the CMS.

o Verified the imported data in the Sanity dashboard to ensure all fields were correctly populated.

5. API Integration in Next.js

• Objective: Display product data dynamically on the frontend.

• Steps Taken:

o Created utility functions to fetch data from Sanity using GROQ queries.

o Rendered product data in components (e.g., ProductCard).

o Tested the integration using tool Postman and browser developer tools.

6. Self-Validation Checklist

Task Status

API Understanding ✓

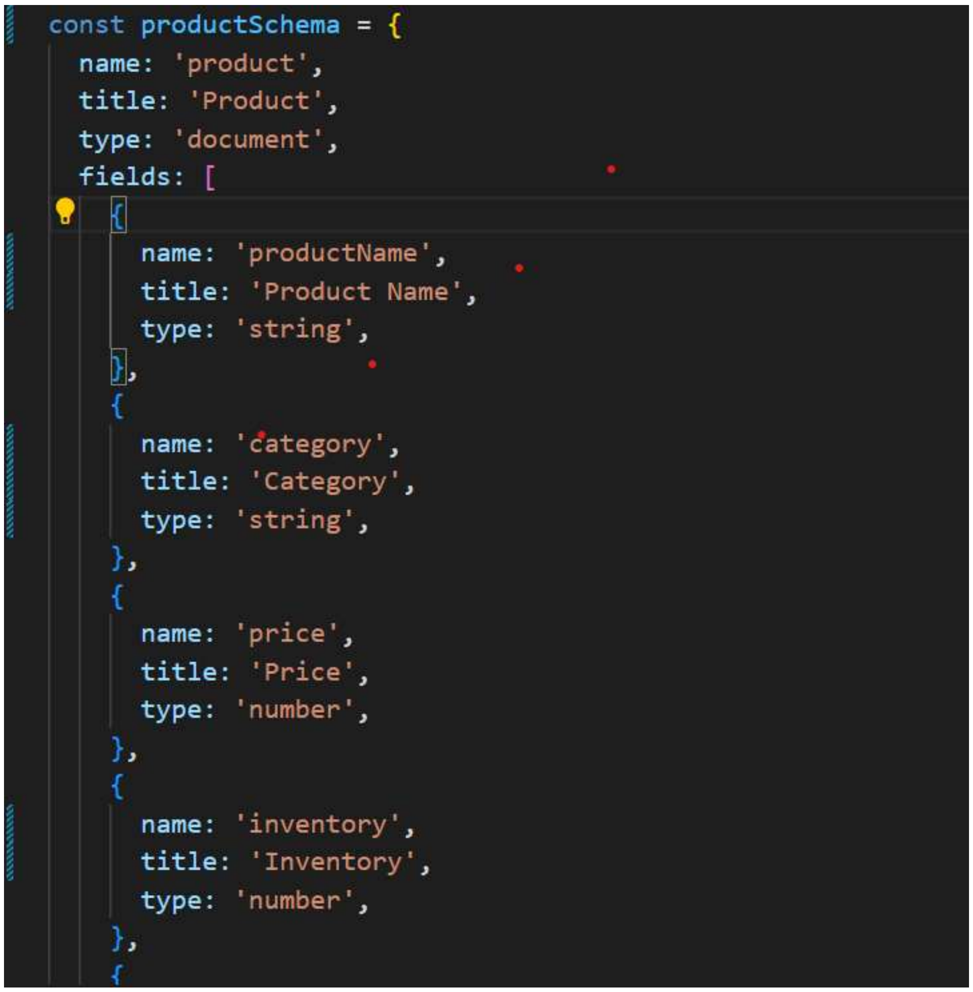
Schema Validation ✓

Data Migration ✓

API Integration in Next.js ✓

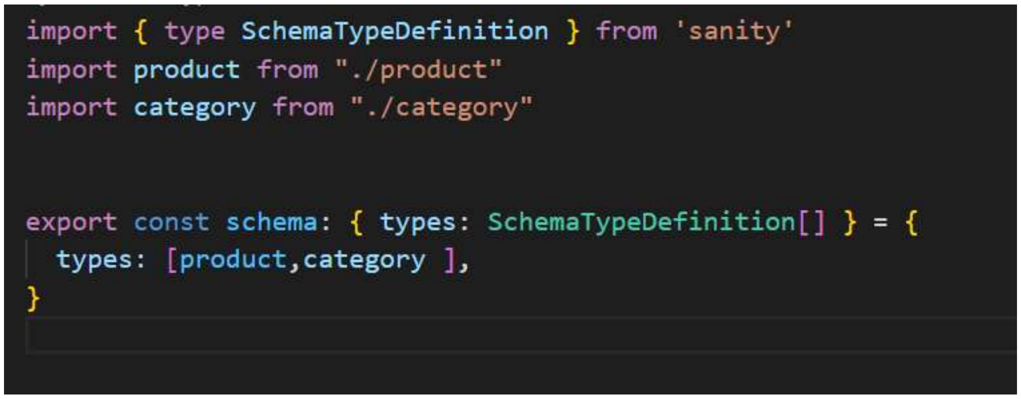
Submission Preparation ✓

7. Here's a snippet of the updated schema:



```
const productSchema = {
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    {
      name: 'productName',
      title: 'Product Name',
      type: 'string',
    },
    {
      name: 'category',
      title: 'Category',
      type: 'string',
    },
    {
      name: 'price',
      title: 'Price',
      type: 'number',
    },
    {
      name: 'inventory',
      title: 'Inventory',
      type: 'number',
    },
    {
```

import categoryType and productSchema in index.ts



```
import { type SchemaTypeDefinition } from 'sanity'
import product from "./product"
import category from "./category"


export const schema: { types: SchemaTypeDefinition[] } = {
  types: [product,category ],
}
```

8. Outcomes of Day 3

• Successfully integrated Sanity CMS with the application.

• Implemented dynamic GROQ queries to fetch and display product data.

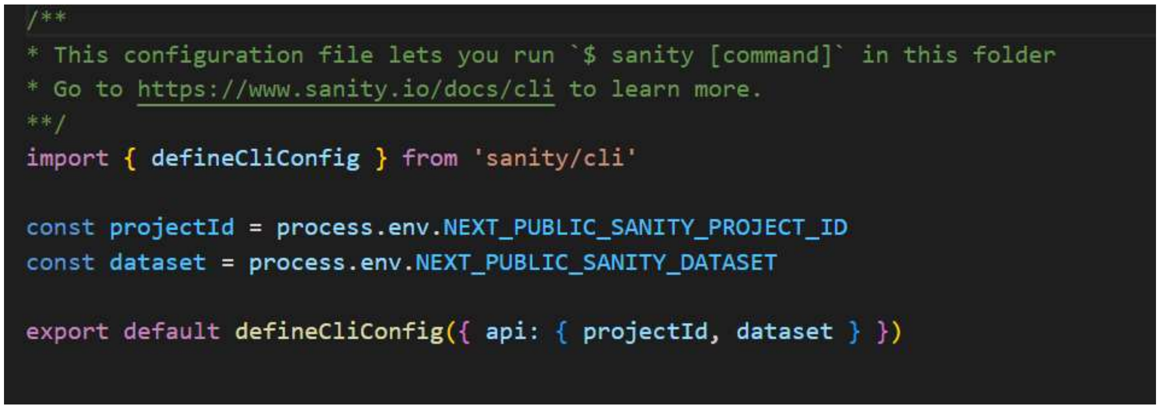
• Enhanced the Sanity schema to support extended product attributes.

• Migrated data and images from an external API to Sanity CMS.

• Validated and tested the integration to ensure seamless backend-to-frontend data flow.

9. Code Snippets

Sanity Client Setup

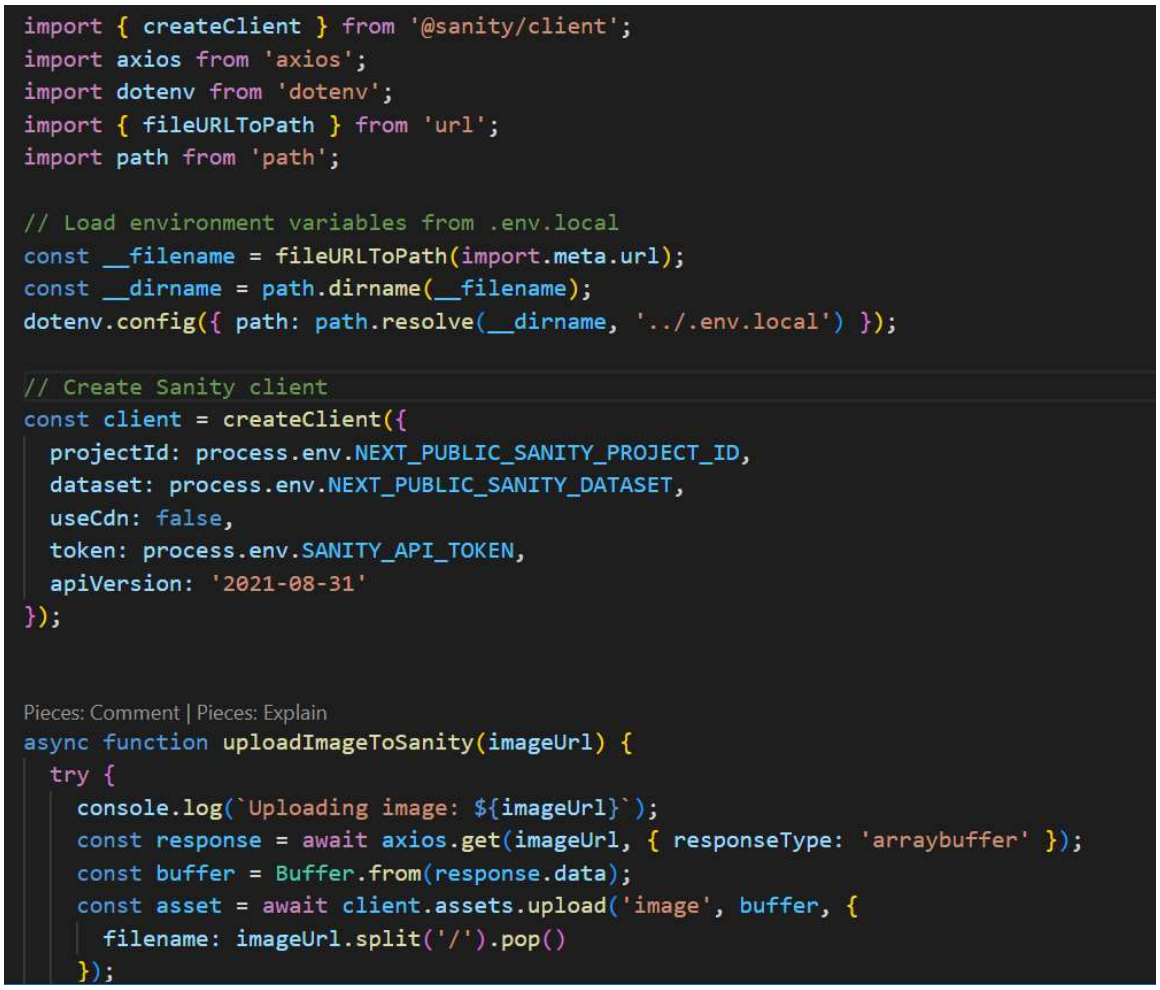


```
/**
* This configuration file lets you run `$ sanity [command]` in this folder
* Go to https://www.sanity.io/docs/cli to learn more.
**/
import { defineCliConfig } from 'sanity/cli'

const projectId = process.env.NEXT_PUBLIC_SANITY_PROJECT_ID
const dataset = process.env.NEXT_PUBLIC_SANITY_DATASET

export default defineCliConfig({ api: { projectId, dataset } })
```

Migration Script (importData.mjs)

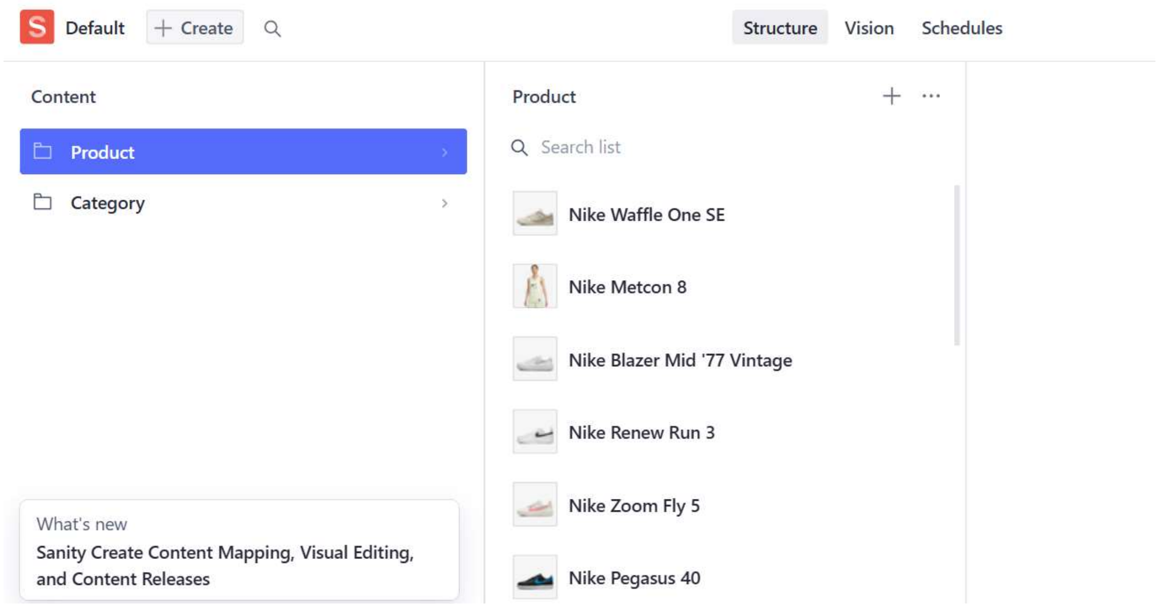


```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// 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'
});


Pieces: Comment | Pieces: Explain
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()
    });
```



Nike Waffle One SE

Nike Metcon 8

Nike Blazer Mid '77 Vintage

Nike Renew Run 3

Nike Zoom Fly 5

Nike Pegasus 40

10. Conclusion

This concludes my Day 3 work for the Nike Store project. I successfully integrated APIs, migrated data to

Sanity CMS, and ensured the frontend dynamically displays product information. I look forward to continuing

to refine and enhance this e-commerce platform.

Check Preview here [https://marketplace-hakathone-yuea.vercel.app/]