

API Integration Report - Nike Store

Name: Dua Fatima

Roll No: 00300923

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 functionality of the provided API.
- Steps Taken:
 - Reviewed the API documentation for the assigned template.
 - Identified key endpoints (e.g., /products) and the structure of the returned data.
 - 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:
 - Used Thunder Client to test the API endpoint and verify the data structure.
 - Created utility functions in the Next.js project to fetch data using fetch or Axios.
 - 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:
 - Created a migration script (importData.mjs) to fetch data from the API and transform it into the required format.
 - Used the Sanity client library to upload data and images to the CMS.
 - 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:
 - Created utility functions to fetch data from Sanity using GROQ queries.
 - Rendered product data in components (e.g., ProductCard).
 - 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',
    },
    {
      name: 'colors',
      title: 'Colors',
      type: 'array',
      of: [{ type: 'string' }],
    },
    {
      name: 'status',
      title: 'Status',
      type: 'string',
    },
    {
      name: 'image',
      title: 'Image',
      type: 'image', // Using Sanitary's image type for image field
      options: {
        hotspot: true,
      },
    },
    {
      name: 'description',
      title: 'Description',
      type: 'text',
    },
  ],
}

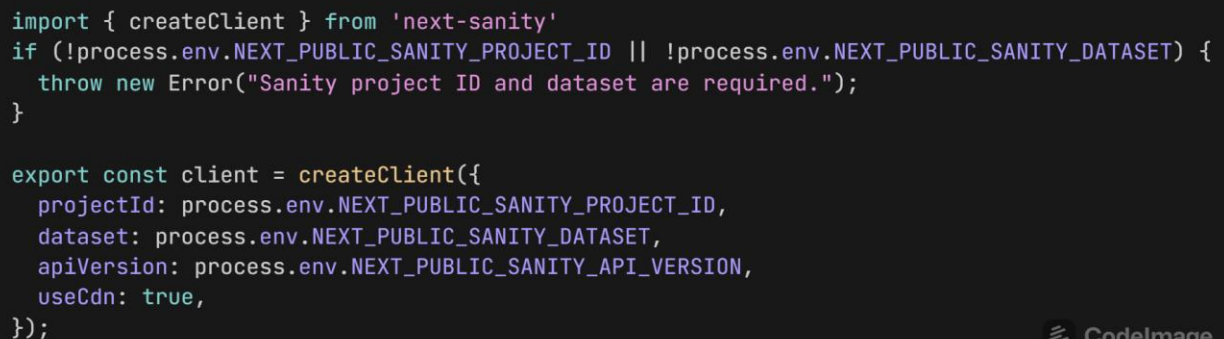
export default productSchema
```

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



```
import { createClient } from 'next-sanity'
if (!process.env.NEXT_PUBLIC_SANITY_PROJECT_ID || !process.env.NEXT_PUBLIC_SANITY_DATASET) {
  throw new Error("Sanity project ID and dataset are required.");
}

export const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  apiVersion: process.env.NEXT_PUBLIC_SANITY_API_VERSION,
  useCdn: true,
});
```

CodeImage

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: '2025-01-17'
});

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('migrating data please wait...');

    // API endpoint containing car data
    const response = await axios.get('https://template-03-api.vercel.app/api/products');
    const products = response.data.data;
    console.log("products ==> ", products);

    for (const product of products) {
      let imageRef = null;
      if (product.image) {
        imageRef = await uploadImageToSanity(product.image);
      }

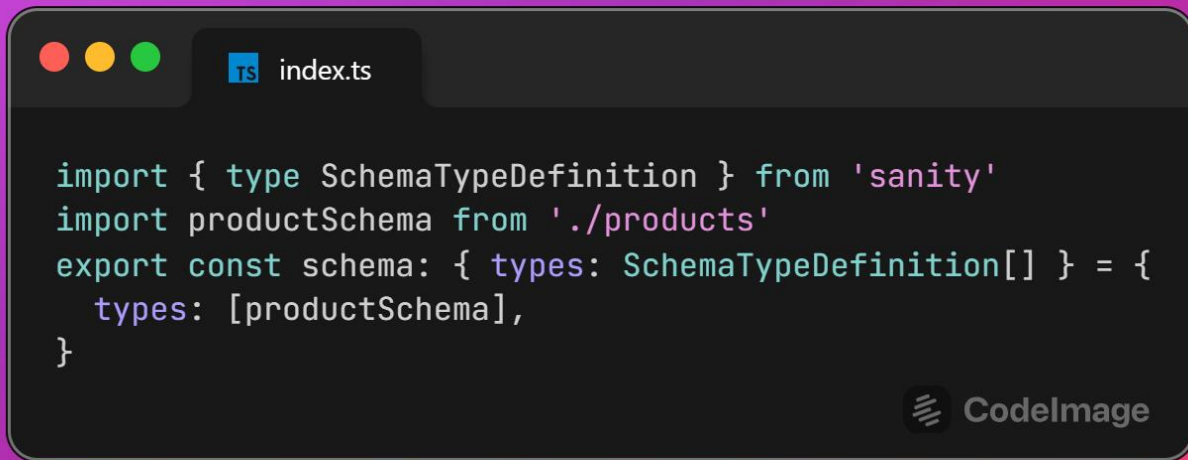
      const sanityProduct = {
        _type: 'product',
        productName: product.productName,
        category: product.category,
        price: product.price,
        inventory: product.inventory,
        colors: product.colors || [], // Optional, as per your schema
        status: product.status,
        description: product.description,
        image: imageRef ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        } : undefined,
      };

      await client.create(sanityProduct);
    }

    console.log('Data migrated successfully!');
  } catch (error) {
    console.error('Error in migrating data ==> ', error);
  }
}

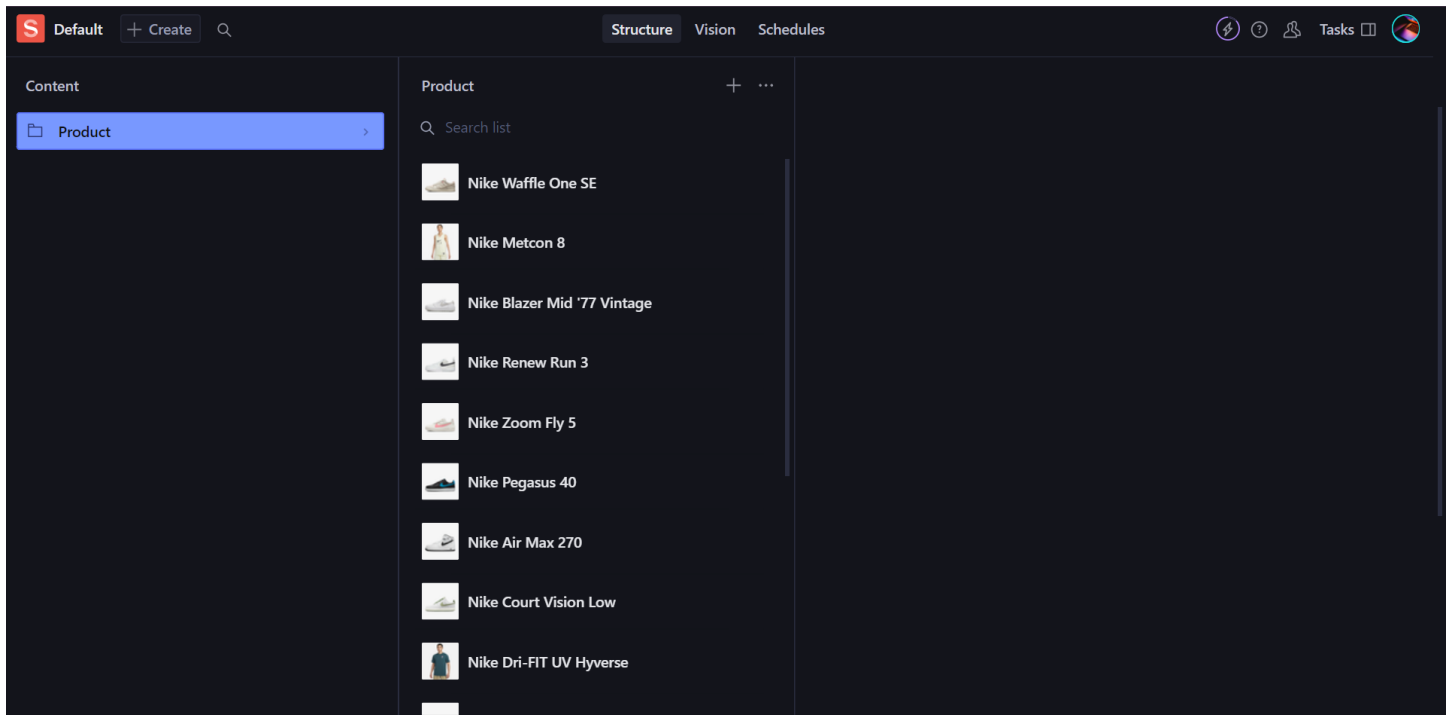
importData();
```

 CodeImage



```
import { type SchemaTypeDefinition } from 'sanity'
import productSchema from './products'
export const schema: { types: SchemaTypeDefinition[] } = {
  types: [productSchema],
}
```

CodelImage



Product List page

New (500)

Shoes

Sports Bras

Tops & T-Shirts

Hoodies & Sweatshirts

Jackets

Trousers & Tights

Shorts

Tracksuits

Jumpsuits & Rompers

Skirts & Dresses

Socks

Accessories & Equipment

Gender

Men

Women

Unisex

5

Hide Filters

Sort By

In Stock

Nike Air Force 1 Mid '07

Men's Shoes

White

MRP : 10795

In Stock

Nike Court Vision Low Next Nature

Men's Shoes

Black

MRP : 4995

In Stock

Air Jordan 1 Elevate Low

Women's Shoes

White

MRP : 11895

Product Details Page

Nike Air Force 1 Mid '07

Men's Shoes

MRP : ₹ 10795

incl. of taxes

(Also includes all applicable duties)

Select Size

Size Guide

UK 6 (EU 40)

UK 7 (EU 41)

UK 8 (EU 42)

UK 9 (EU 43)

UK 10 (EU 44)

UK 11 (EU 45)

UK 12 (EU 46)

Add to Bag

Shipping

You'll see our shipping options at checkout.

The Nike Air Force 1 Mid '07 delivers timeless style with premium leather and mid-cut design. Perfect for everyday

7 of 8

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://nike-app-df.vercel.app/>]