

# **Day 3 - API Integration Report**

## **API Integration Process**

1. **Overview:** The integration connects an external API providing **Nike** data to a Sanity CMS project.
2. **Steps Taken:**
  - **Environment Setup:**
    - Used **dotenv** to load environment variables from **.env.local**.
    - Key variables included:
      - NEXT\_PUBLIC\_SANITY\_PROJECT\_ID
      - NEXT\_PUBLIC\_SANITY\_DATASET
      - SANITY\_TOKEN
  - **Sanity Client Creation:**
    - Used **@sanity/client** to establish a connection to the Sanity project.
    - Configured the client with the project ID, dataset, API version, and authentication token.
    - **Data Fetching:**
      - Made concurrent API calls using **axios** to fetch food and chef data.
      - Endpoints accessed:
        - <https://sanity-nextjs-rouge.vercel.app/api/Nike>
  - **Data Processing:**
    - Iterated through the fetched data.
    - Uploaded images to Sanity's asset library using the **client.assets.upload()** method.
  - **Sanity Document Creation:**
    - Transformed fetched data into Sanity-compatible document structures.
    - Uploaded each document using **client.create()**.

### 3. Error Handling:

- Implemented try-catch blocks for API calls and Sanity operations.
- Logged errors for debugging.

## Schema : for Nike E-Commerce Website.

### 1. Product Schema:

- **Title:** (String) Name of the product (e.g., Nike Air Max 270)
- **Category:** (String) Shoes, Clothing, Accessories, etc.
- **Price:** (Number) Current price
- **Original Price:** (Number, Optional) Discounted price
- **Tags:** (Array) Categories like "Men," "Running," "Limited Edition"
- **Stock Status (Available):** (Boolean) Whether the product is in stock or not
- **Size Options:** (Array) Different sizes available (e.g., 7, 8, 9, 10)
- **Color Variants:** (Array) Available color options
- **Image Handling:**
  - **Enabled Hotspot:** Allows flexible cropping for product images
  - **Multiple Image Support:** Supports multiple images for different angles

### 2. User Schema:

- **Username:** (String) Name of the user
- **Email:** (String) Email address
- **Password:** (String, Hashed) For security
- **Role:** (String) User role (e.g., "Admin", "Customer")
- **Order History:** (Array) Details of previous orders
- **Wishlist:** (Array) List of favorite products

### 3. Order Schema:

- **User ID:** (Reference) The user who placed the order
- **Products:** (Array) List of ordered products
- **Total Amount:** (Number) Total order cost
- **Payment Status:** (String) "Paid" or "Pending"
- **Order Status:** (String) "Processing," "Shipped," "Delivered"
- **Address:** (String) Delivery address

## Migration Steps and Tools Used:

### 1. Data Preparation:

- Analyzed the API data structure and mapped it to **Sanity schemas**
- Created **Sanity schemas** for products and users

### 2. Data Import Script:

- Built **import-data.mjs** script to automate **fetching, processing, and uploading**
- Used **Axios** for API interaction
- Used **@sanity/client** for CMS operations

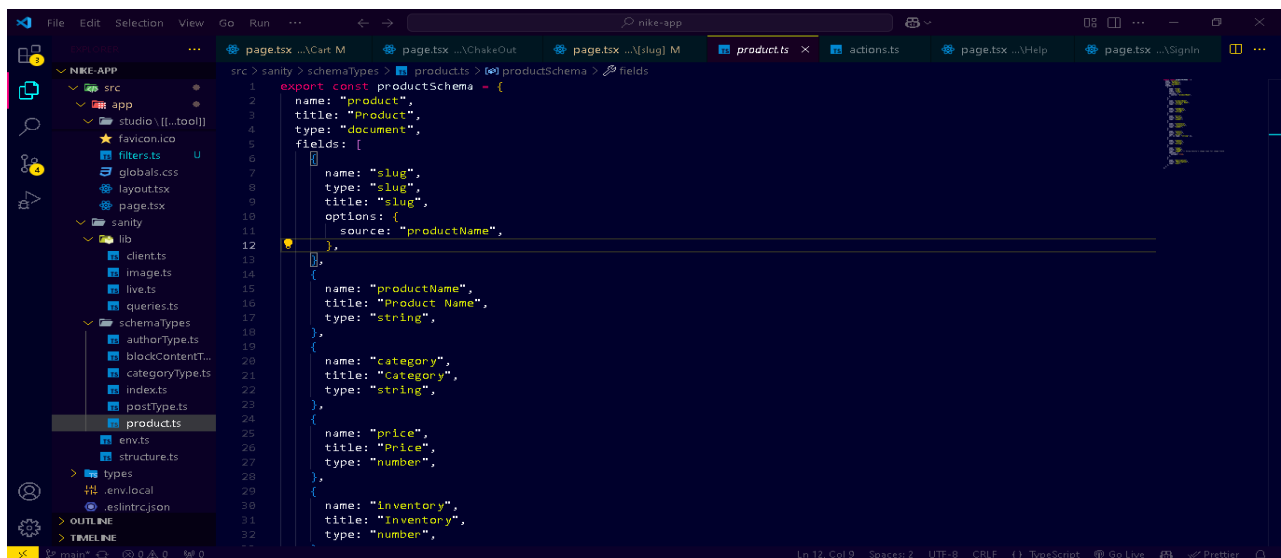
### 3. Image Handling:

- Downloaded images using **Axios** with `arraybuffer` responseType
- Uploaded images to **Sanity**, storing **asset references** in document fields

### 4. Document Creation:

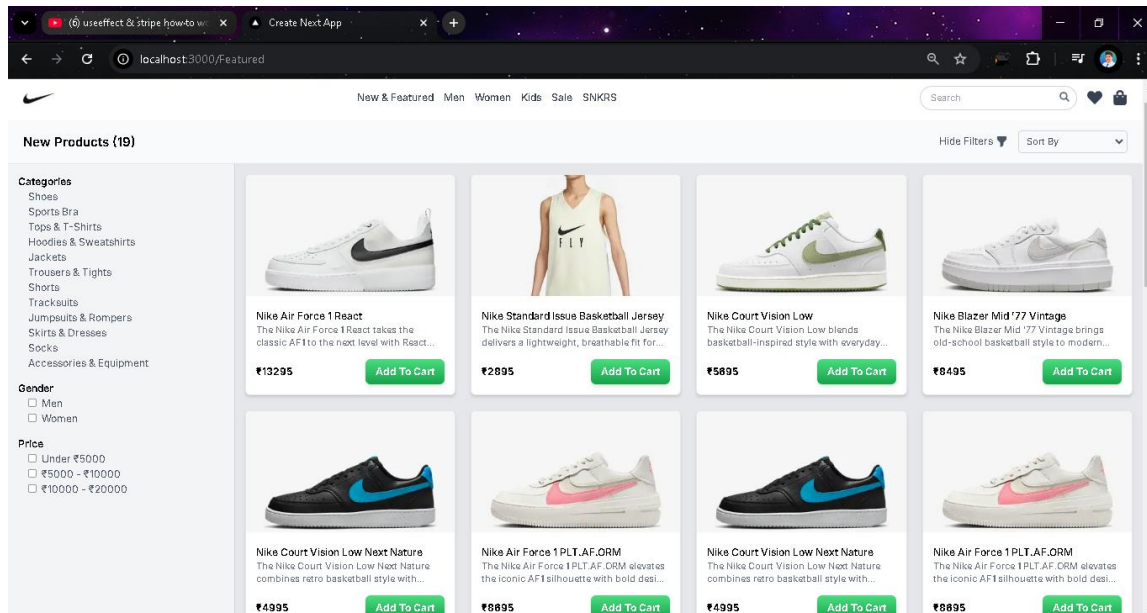
- Mapped API data fields to **Sanity schema fields**
- Set **fallback values** for optional or missing fields

## Screenshot of:



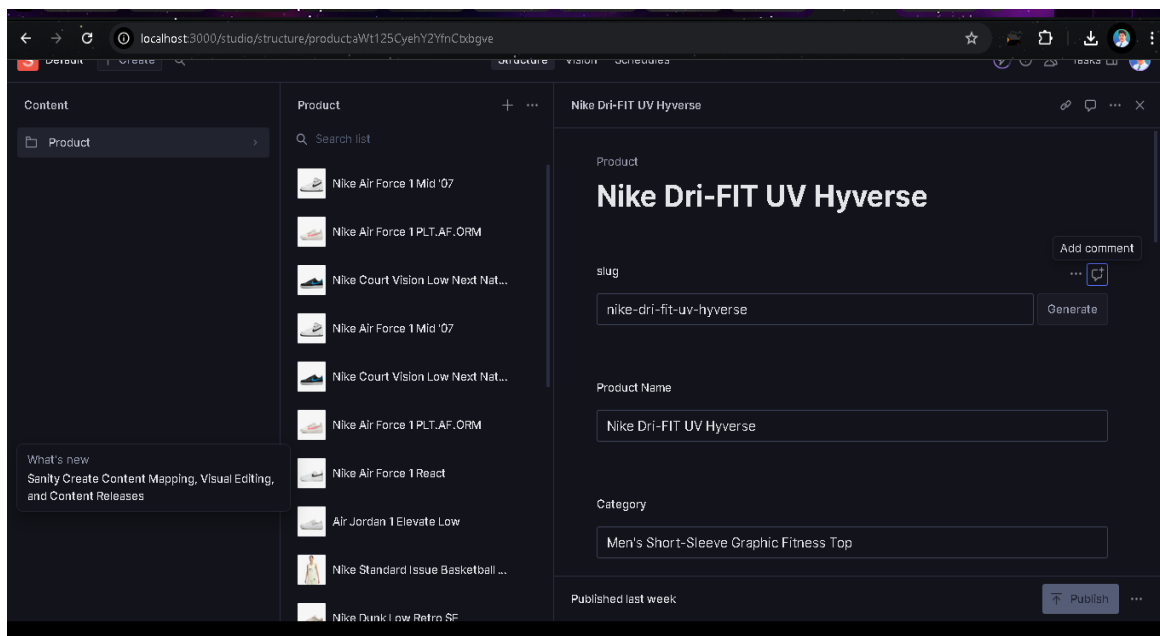
## Data successfully displayed in the frontend:

### All Product:

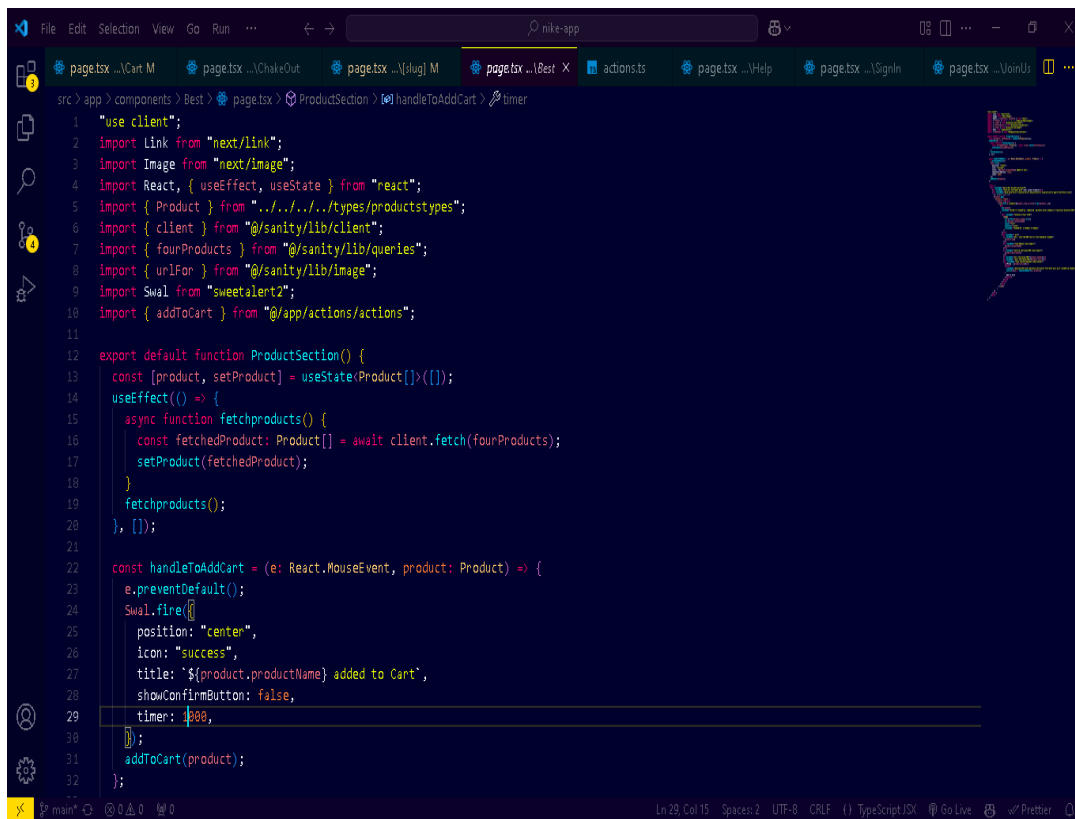


## Populated sanity CMS fields:

### Product :



## ***Code Snippets for API integration and migration scripts:***



```
1  "use client";
2  import Link from "next/link";
3  import Image from "next/image";
4  import React, { useEffect, useState } from "react";
5  import { Product } from "../../types/producttypes";
6  import { client } from "@sanity/lib/client";
7  import { fourProducts } from "@sanity/lib/queries";
8  import { urlFor } from "@sanity/lib/image";
9  import Swal from "sweetalert2";
10 import { addToCart } from "@app/actions/actions";
11
12 export default function ProductSection() {
13   const [product, setProduct] = useState<Product[]>([]);
14   useEffect(() => {
15     async function fetchproducts() {
16       const fetchedProduct: Product[] = await client.fetch(fourProducts);
17       setProduct(fetchedProduct);
18     }
19     fetchproducts();
20   }, []);
21
22   const handleToAddCart = (e: React.MouseEvent, product: Product) => {
23     e.preventDefault();
24     Swal.fire({
25       position: "center",
26       icon: "success",
27       title: `${product.productName} added to Cart`,
28       showConfirmButton: false,
29       timer: 300,
30     });
31     addToCart(product);
32   };
33 }
```

## ***Final check:***

API Understanding	Schema Validation	Data Migration	API Integration in Next.js	Submission Preparation
✓	✓	✓	✓	✓

By NOMAN