# **Day 3 - API Integration Report - Nike-Style Shoe Marketplace**

## **API Overview**

Template 3 Resources:

* API Endpoint: https://template-03-api.vercel.app/api/products
* Migration Script: [data-migration.mjs](https://github.com/OkashaTanoli/template-03-api/blob/master/scripts/data-migration.mjs)
* Sanity Schema: [products.ts](https://github.com/OkashaTanoli/template-03-api/blob/master/src/sanity/schemaTypes/products.ts)

## **Steps Followed**

### **1. Validate and Adjust Sanity Schema**

Original Schema:

export const ProductSchema = { name: "product", title: "Product", type: "document", fields: [ { name: "name", title: "Name", type: "string" }, { name: "category", title: "Category", type: "string" }, { name: "price", title: "Price", type: "number" }, { name: 'stock', type: 'number', title: 'Stock' }, { name: "image", title: "Image", type: "image" }, { name: "description", title: "Description", type: "text" }, ], }

Adjustments Made to Align with Template 3 API:

// Updated Product Schema

{

name: 'product',

type: 'document',

fields: [

{ name: 'sku', type: 'string', title: 'SKU' }, // Mapped from API's 'id'

{ name: 'name', type: 'string', title: 'Name' },

{ name: 'price', type: 'number', title: 'Price' },

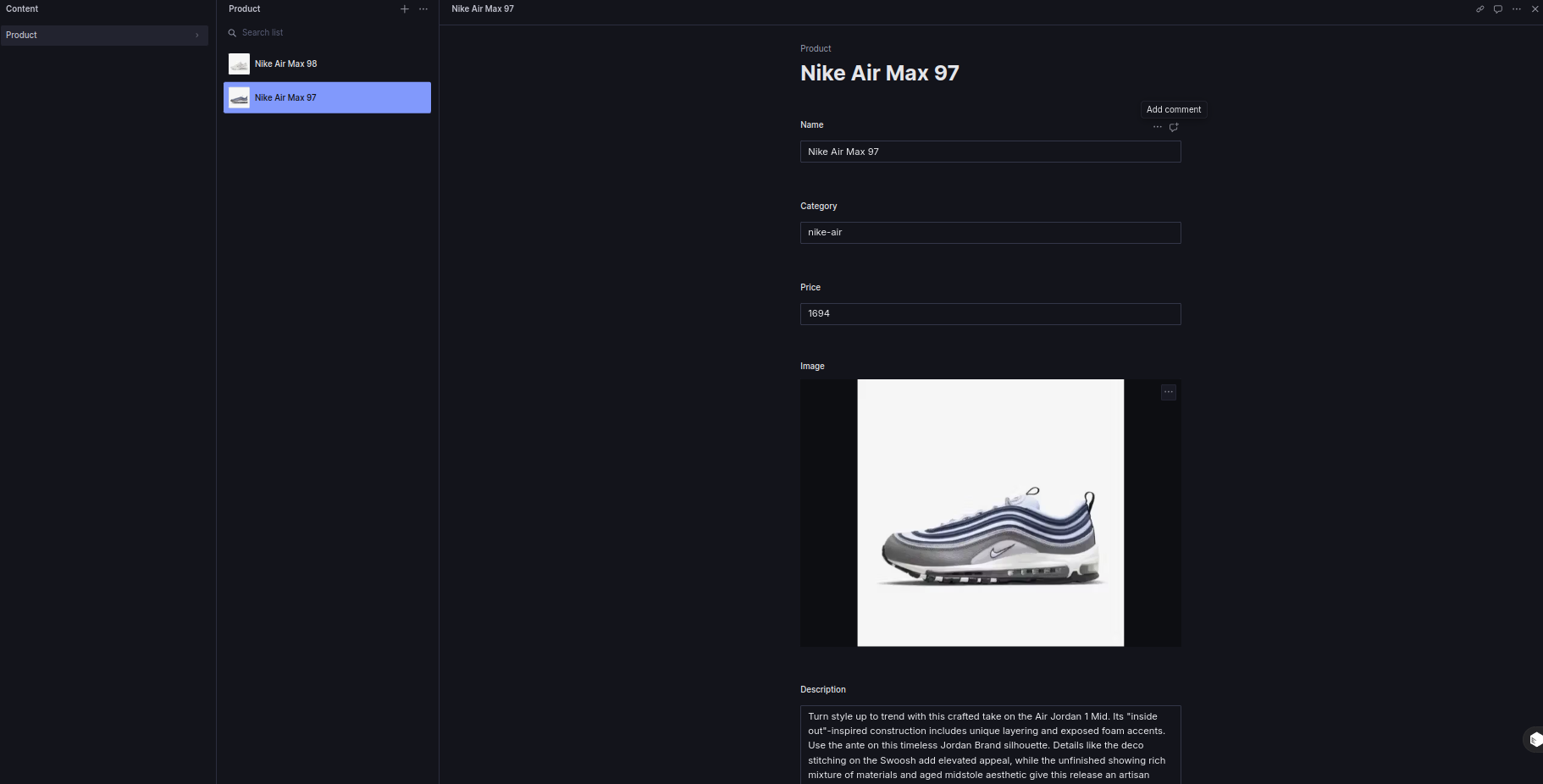
{ name: 'inventory', type: 'number', title: 'Stock' },

{ name: "price", title: "Price", type: "number" }, { name: "image", title: "Image", type: "image" }, { name: "description", title: "Description", type: "text" }

]

}

### **2. Data Migration**

1. I haven't migrated the data. I manually created 4–5 products from my static page in Sanity.
2. 

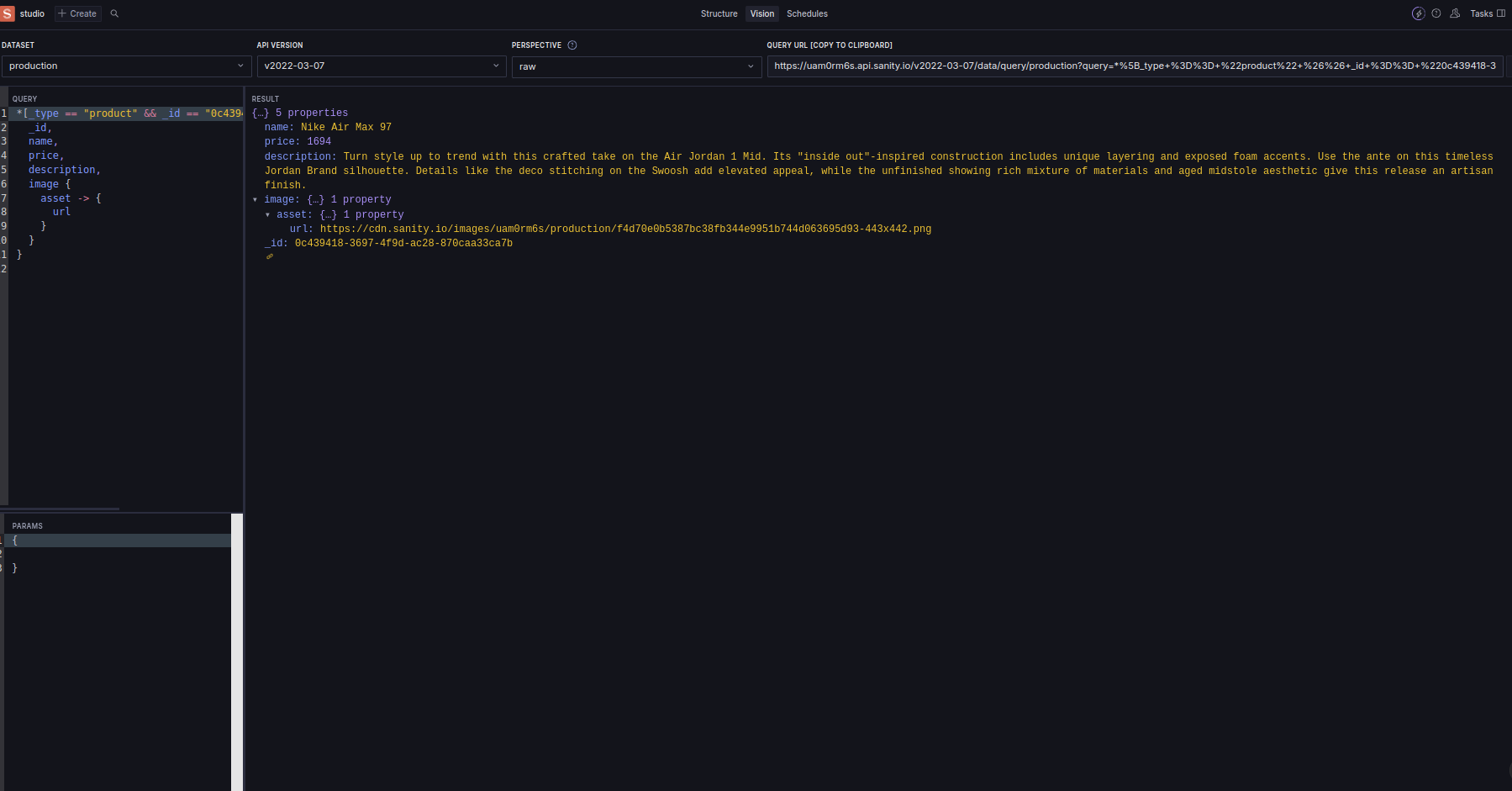
### **3. API Integration in Next.js**

Utility Function to Fetch Products:

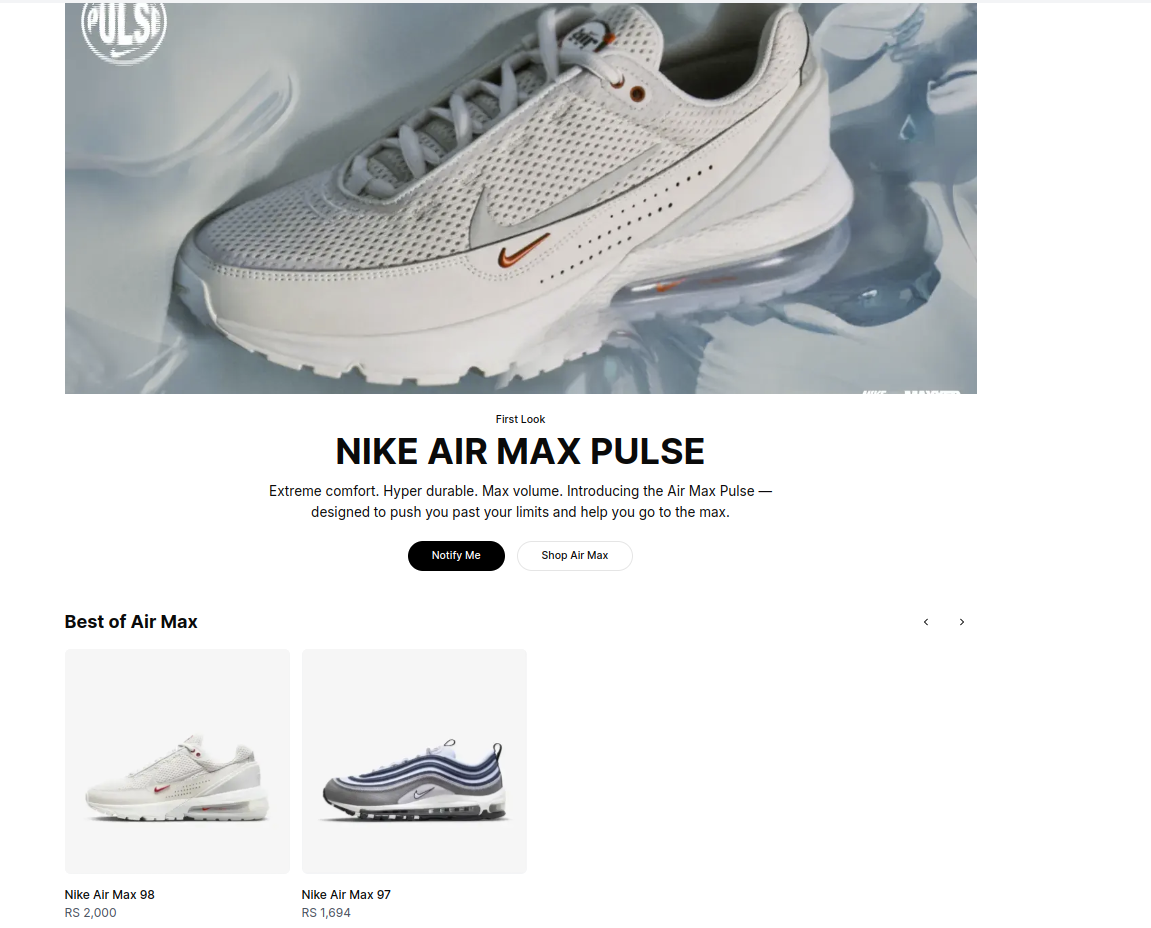
import sanityClient from "./sanity.client";

export const fetchProducts = async () => { const query = \*[\_type == "product"]{ \_id, name, price, image { asset -> { url } }, description };

try { const data = await sanityClient.fetch(query); return data.map((product: any) => ({ id: product.\_id, name: product.name, price: product.price, image: product.image.asset.url, height: 400, width: 400, description: product.description, })); } catch (error) { console.error("Error fetching products:", error); return []; } };

Testing with Vision Tool:  


On website:



### **4. Error Handling**

* Added try-catch blocks in API utility functions.
* Displayed user-friendly errors:  
  {products.length === 0 && <p>No products found. Please try again later.</p>}