Table of Contents

Day 3 - API Integration Report – General E-Commerce

Introduction

This report documents the process of integrating the Sanity CMS API into a Next.js application for managing and displaying dynamic data. The integration ensures seamless communication between the backend and frontend, enabling efficient data retrieval and visualization.

The report outlines the following key areas:

- Step-by-step API integration, including configuration and usage in the application.
- Adjustments made to Sanity schemas to accommodate project-specific requirements.
- Migration of existing data into Sanity CMS, detailing the tools and steps used.

Additionally, the report includes screenshots of API calls, data displayed on the frontend, and populated Sanity CMS fields. Relevant code snippets for the integration and migration scripts are provided for a comprehensive understanding of the process.

The objective of this report is to serve as a guide for developers, showcasing the implementation of a CMS-powered application while ensuring scalability and maintainability.

API Integration Process

The API integration process involved fetching product data from an external API and utilizing it in the Next.js application. The following steps detail the approach used for seamless integration:

1. Fetching Data from the External API

- A custom asynchronous function named importData was created to fetch product data.
- The axios library was used to make HTTP requests to the external API.
- The API endpoint used to fetch the data was: `https://template1-neon-nu.vercel.app/api/products.`

2. Configuring Axios

- The axios library was installed as a dependency using the following command:

```
npm install axios
```

3. Data to be fetched

- Products Data

Changes made to Schema

Initially, the schema I created used a specific naming convention where all field names were prefixed with "product" for clarity (e.g., productName, productPrice). However, the provided schema followed a simplified naming convention without such prefixes (e.g., name, price).

Fields removed from Schema:

- Stock
- Tags

Fields added in Schema:

- Sizes
- Colors
- disccountPercent

Updated Schema

```
import { defineType } from "sanity"

export const productSchema = defineType({
  name: 'products',
  title: 'Products',
  type: 'document',
  fields: [
    {
    name: 'name',
    title: 'Name',
    type: 'string',
    },
    {
    name: 'price',
    title: 'Price',
    type: 'number',
    },
    {
    name: 'description',
    title: 'Description',
    type: 'text',
    },
    {
    name: 'image',
    title: 'Image',
    type: 'image',
    },
      {
        name:"category",
        title:"Category",
        type: 'string',
        options:{
          list:[
            {title: 'T-Shirt', value: 'tshirt'},
            {title: 'Short', value: 'short'},
            {title: 'Jeans', value: 'jeans'} ,
            {title: 'Hoddie', value: 'hoodie'} ,
            {title: 'Shirt', value: 'shirt'} ,
          ]
        }
      },
      {
        name:"discountPercent",
        title:"Discount Percent",
        type: 'number',
      },
      {
        name:"isNew",
        type: 'boolean',
        title:"New",
      },
      {
        name:"colors",
        title:"Colors",
        type: 'array',
        of:[
          {type: 'string'}
        ]
      },
      {
        name:"sizes",
        title:"Sizes",
        type: 'array',
        of:[
          {type: 'string'}
        ]
      }
    ],
})
```

Migration Steps and Tools Used

The migration process involved transferring data from an external API to Sanity CMS. Below are the detailed steps and tools used for this process:

1. Fetching Data from External API

- **Tool/Technology Used:** `Axios` (for HTTP requests)
- Created an asynchronous function named `importData` to fetch data from the external API.
- Utilized `axios.get()` to make a GET request to the API endpoint.
- Parsed and logged the fetched data to ensure accuracy.

2. Sanity Client Setup

- **Tool/Technology Used:** `@sanity/client`
- Configured the Sanity client using credentials from environment variables.
- Included the `SANITY_API_TOKEN` to authenticate API calls.

3. Uploading Images to Sanity

- **Tool/Technology Used:** Sanity Asset Upload API
- Implemented an image upload helper function using `client.assets.upload`.
- Converted image data to a buffer format and uploaded images to Sanity CMS.

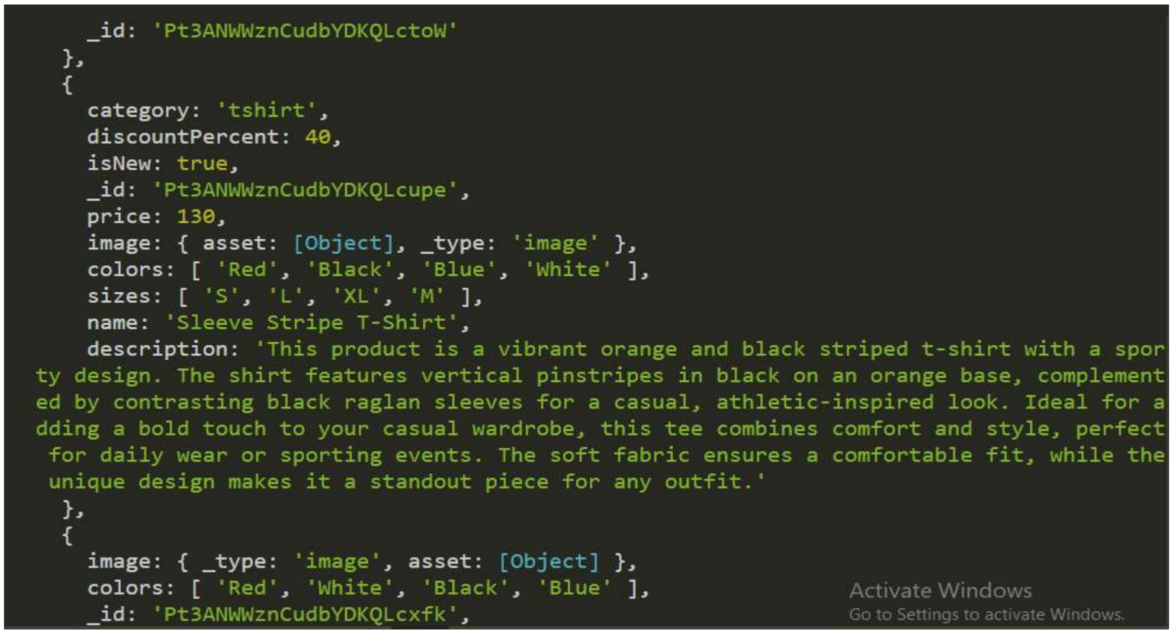
4. Migrating Product Data to Sanity

- For each product fetched from the API, created a corresponding document in Sanity CMS.
- Handled image references separately using the uploaded image asset IDs.

Tools Used

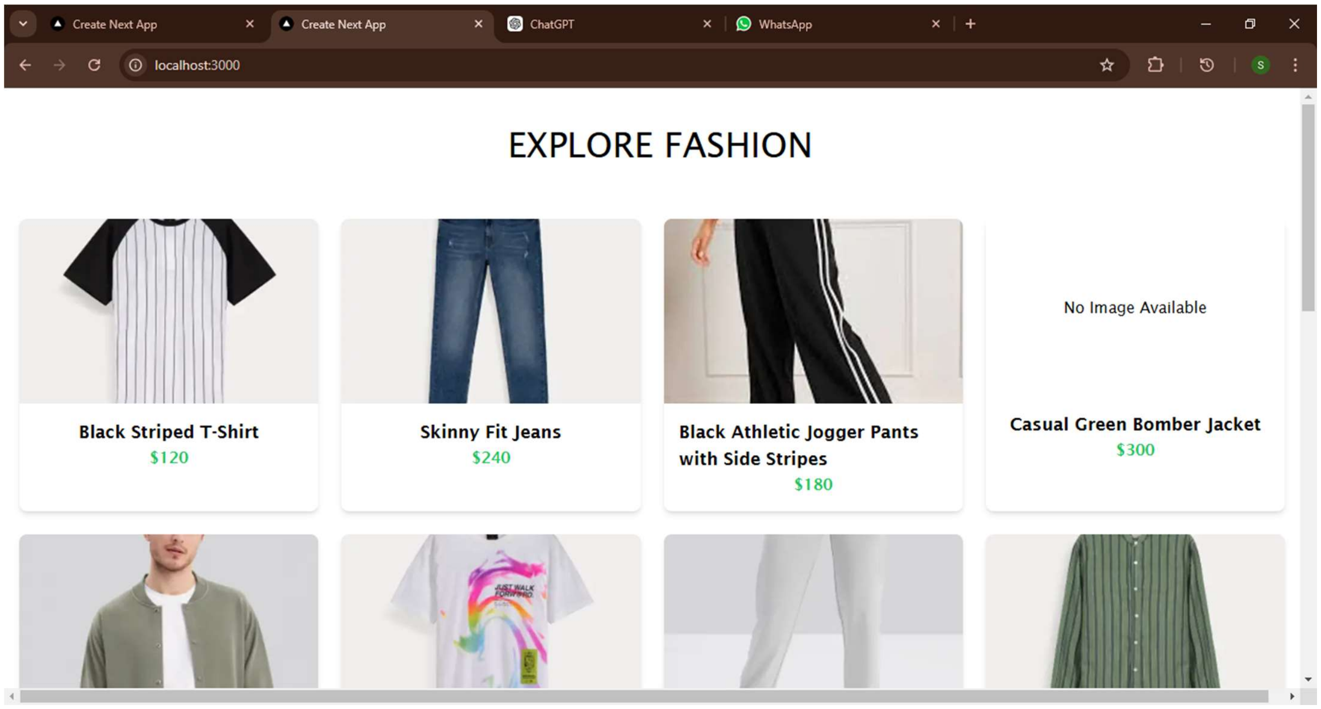
1. **Axios:** For fetching data from the external API.
2. **Sanity Client:** For interacting with Sanity CMS.
3. **Node.js Buffer:** For handling binary data during image upload.
4. **Environment Variables:** Used `.env.local` for managing sensitive credentials securely.

Screenshot of API Calls



```
    _id: 'Pt3ANWWznCudbYDKQLctoW'
  },
  {
    category: 'tshirt',
    discountPercent: 40,
    isNew: true,
    _id: 'Pt3ANWWznCudbYDKQLcupe',
    price: 130,
    image: { asset: [Object], _type: 'image' },
    colors: [ 'Red', 'Black', 'Blue', 'White' ],
    sizes: [ 'S', 'L', 'XL', 'M' ],
    name: 'Sleeve Stripe T-Shirt',
    description: 'This product is a vibrant orange and black striped t-shirt with a spor
ty design. The shirt features vertical pinstripes in black on an orange base, complement
ed by contrasting black raglan sleeves for a casual, athletic-inspired look. Ideal for a
dding a bold touch to your casual wardrobe, this tee combines comfort and style, perfect
 for daily wear or sporting events. The soft fabric ensures a comfortable fit, while the
 unique design makes it a standout piece for any outfit.'
  },
  {
    image: { _type: 'image', asset: [Object] },
    colors: [ 'Red', 'White', 'Black', 'Blue' ],
    _id: 'Pt3ANWWznCudbYDKQLcxfk',
```

Data successfully displayed in the frontend



EXPLORE FASHION

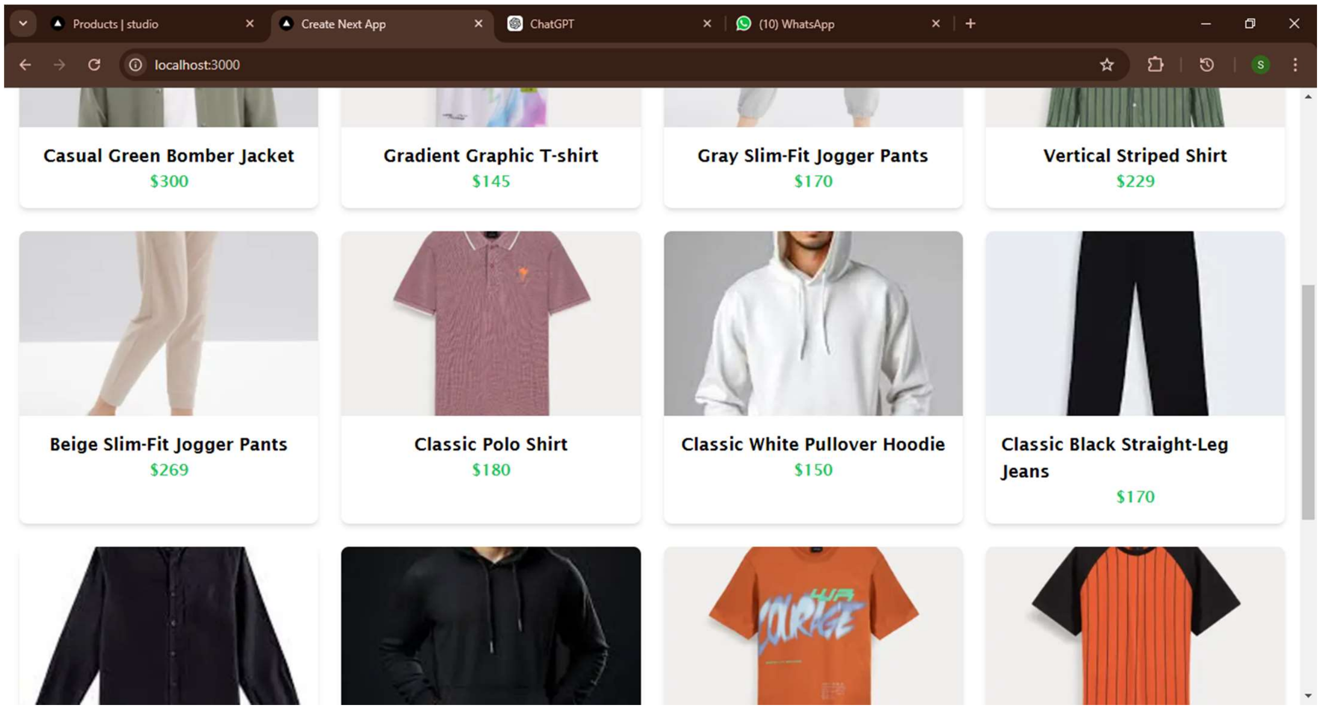
Black Striped T-Shirt
$120

Skinny Fit Jeans
$240

Black Athletic Jogger Pants with Side Stripes
$180

No Image Available

Casual Green Bomber Jacket
$300



Casual Green Bomber Jacket
$300

Gradient Graphic T-shirt
$145

Gray Slim-Fit Jogger Pants
$170

Vertical Striped Shirt
$229

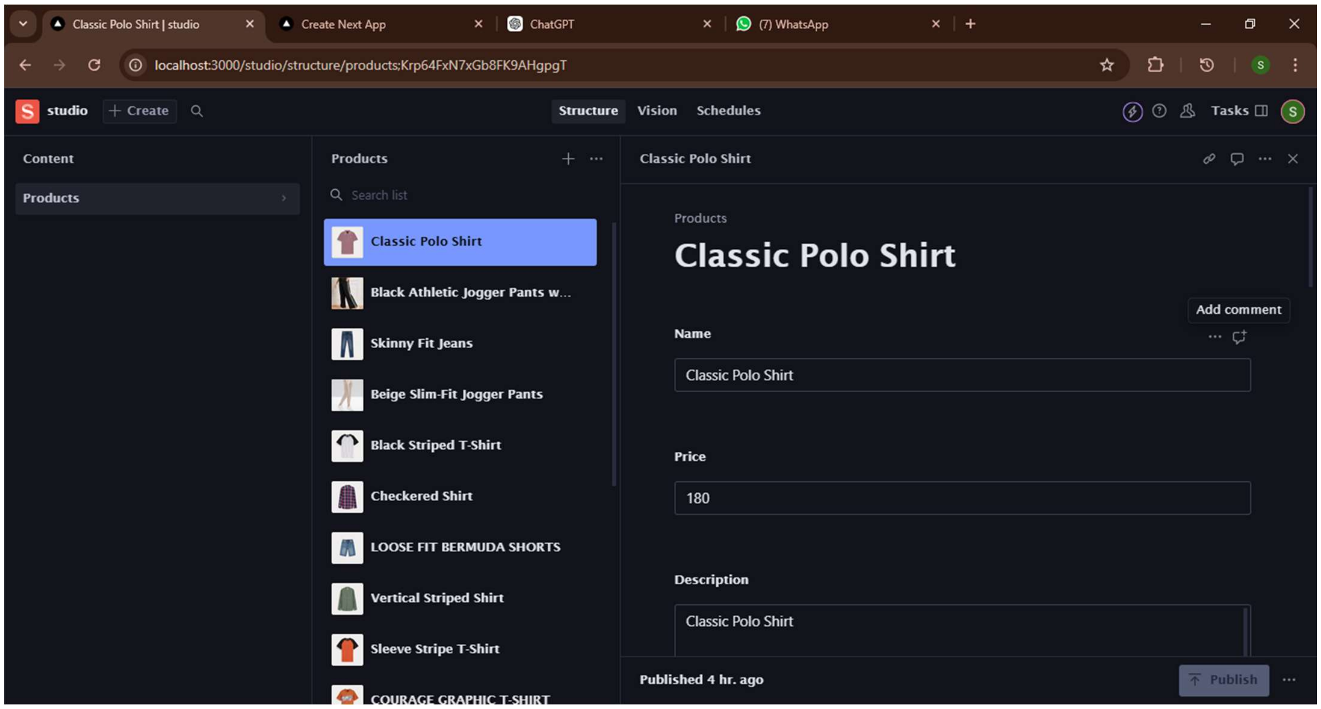
Beige Slim-Fit Jogger Pants
$269

Classic Polo Shirt
$180

Classic White Pullover Hoodie
$150

Classic Black Straight-Leg Jeans
$170

Populated Sanity CMS fields



Products

Classic Polo Shirt

Name

Classic Polo Shirt

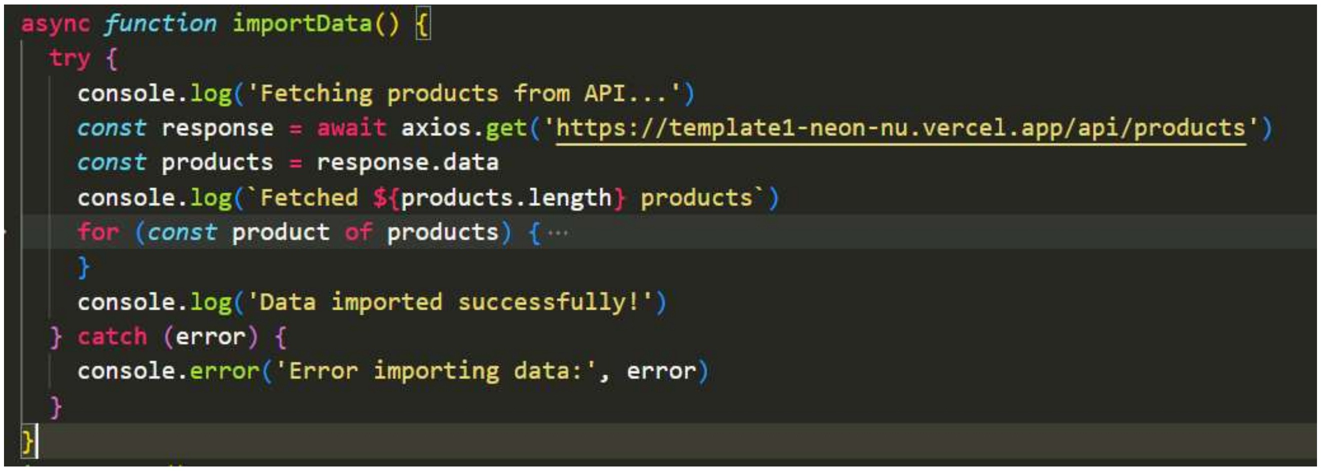
Price

180

Description

Classic Polo Shirt

Code snippet for API integration



```
async function importData() {
  try {
    console.log('Fetching products from API...')
    const response = await axios.get('https://template1-neon-nu.vercel.app/api/products')
    const products = response.data
    console.log(`Fetched ${products.length} products`)
    for (const product of products) {⋯
    }
    console.log('Data imported successfully!')
  } catch (error) {
    console.error('Error importing data:', error)
  }
}
```

Code snippets for API migration scripts

```
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: '2025-01-18'
})
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('Fetching products from API...')
    const response = await axios.get('https://template1-neon-nu.vercel.app/api/products')
    const products = response.data
    console.log(`Fetched ${products.length} products`)
    for (const product of products) {
      console.log(`Processing product: ${product.name}`)
      let imageRef = null
      if (product.imageUrl) {
        imageRef = await uploadImageToSanity(product.imageUrl)
      }
      const sanityProduct = {
        _type: 'products',
        name: product.name,
        description: product.description,
        price: product.price,
        image: imageRef? {
          _type: 'image',
          asset: {
            _ref: imageRef,
          },
        }:undefined,
        category: product.category,
        discountPercent: product.discountPercent,
        isNew: product.isNew,
        colors: product.colors,
        sizes: product.sizes
      }
      console.log('Uploading product to Sanity:', sanityProduct.name)
      const result = await client.create(sanityProduct)
      console.log(`Product uploaded successfully: ${result._id}`)
    }
    console.log('Data imported successfully!')
  } catch (error) {
    console.error('Error importing data:', error)
  }
}
importData()
```