

API Integration Report- Aromas

Asaad Hussain

00414030

GIAIC, Quarter 2

18th January 2025

Table of Content

API integration process	3
Adjustment to Schema	3
Migration Steps and tools	4
Screenshots	4

API integration process


Creating own API

As no proper API was available online, I created my own API for perfumes,

A data file is created where all the data is put, then this GET and returned as a json, the API is hosted on vercel



```
EXPLORER  ...  {} perfume.json X
src > data > {} perfume.json > {} 13 > # stock_quantity
1  [
2    {
3      "product_name": "ACQUA DI",
4      "price": 1800,
5      "description": "A fresh aquatic fragrance perfect for summer days",
6      "size": 50,
7      "stock_quantity": 25,
8      "category": "perfume",
9      "tags": [
10       "uni",
11       "perfume"
12     ],
13     "image": "https://api-perfume.vercel.app/product/aqua%20di%20gio.png",
14     "priceAdiscount": 1600,
15     "product_id": "1"
16   },
17   {
18     "product_name": "COCO CHANEL",
19     "price": 3500,
20     "description": "A luxurious ambr fragrance for special occasions."
21   }
22 ]
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS A:\Work\GAIC\API\api_perfume>
```



```
EXPLORER  ...  TS route.ts X
src > app > api > product > TS route.ts > GET
1  import { NextResponse } from 'next/server';
2  import data from '@data/perfume.json';
3
4  export async function GET() {
5    return NextResponse.json({success:true, data:data});
6  }
TS route.ts
```

Integration of API

A file is made in the src directory of the next app with the name of getProducts.ts in that 2 functions are written, both getting products from the sanity studio a GROQ query is written in the fetch function which is a function of client

In the main page,tsx of the app directory where some products are returned an interface is created to handle type of the data fetched the function is called and the data returned is mapped to show card component

Adjustments made to schemas

The adjustment made to schema was according to data api, the scema created in sanity and the names of the data in the migration should be same.

Migration steps and tools used

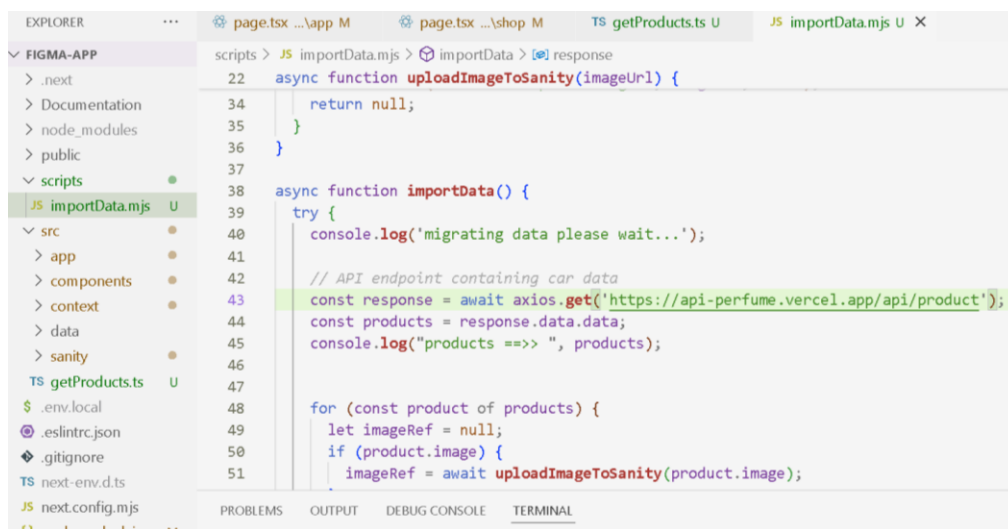
- For migration of the api data to sanity, a scripts folder is created in that a file importData.mjs,
- Secondly in the .env file the project id of the sanity and the data set type with the API token is stored.
- A client is needed to set with the above set credentials
- There are 2 functions one uploads the Images and the other imports the data, It is important to correctly set the schema

Tools used

- **Axios** a popular, promise-based HTTP client for JavaScript that simplifies making HTTP requests to REST endpoints.
- **.env** a file where all secret keys and API's are stored and it is put in .gitignore
- **sanity** Cloud-based, open source unified content platform
- **API token** , token created from sanity

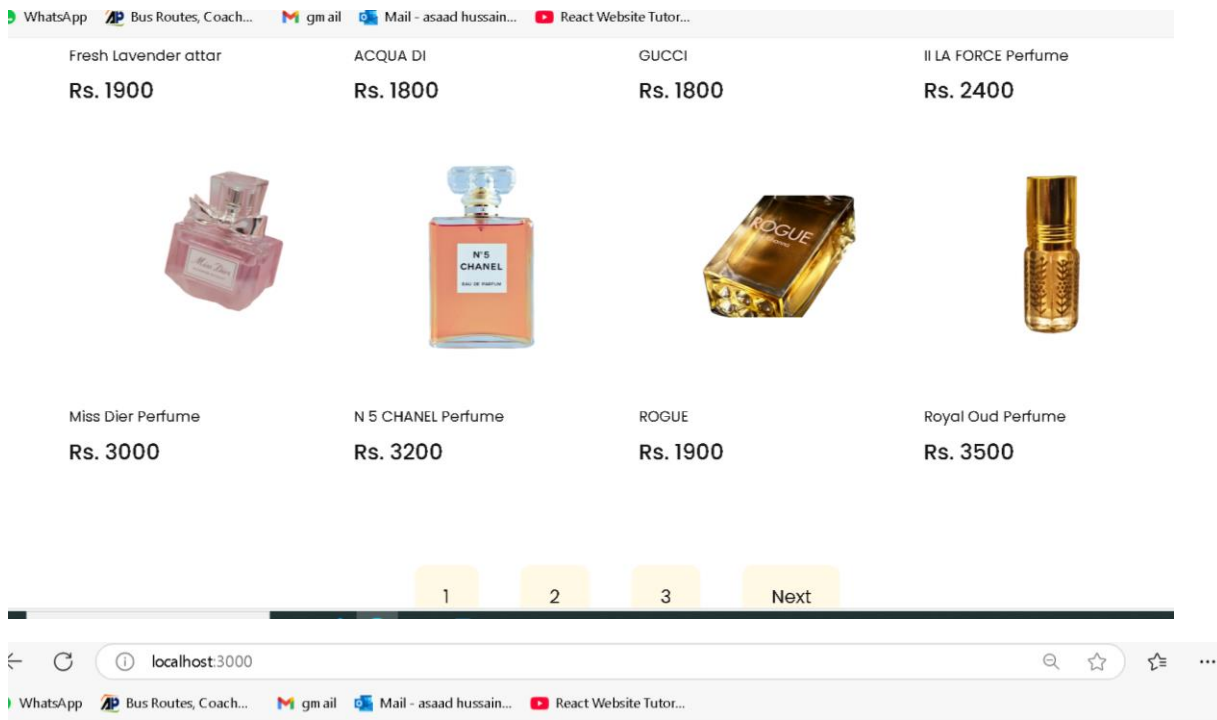
Screenshots

API calls



```
scripts > JS importData.mjs > importData > [response]
22  async function uploadImageToSanity(imageUrl) {
34      return null;
35  }
36  }
37
38  async function importData() {
39      try {
40          console.log('migrating data please wait...');
41
42          // API endpoint containing car data
43          const response = await axios.get('https://api-perfume.vercel.app/api/product');
44          const products = response.data.data;
45          console.log("products ==>> ", products);
46
47          for (const product of products) {
48              let imageRef = null;
49              if (product.image) {
50                  imageRef = await uploadImageToSanity(product.image);
51              }
52          }
53      } catch (error) {
54          console.error('Error migrating data:', error);
55      }
56  }
```

Data successfully displayed in the frontend

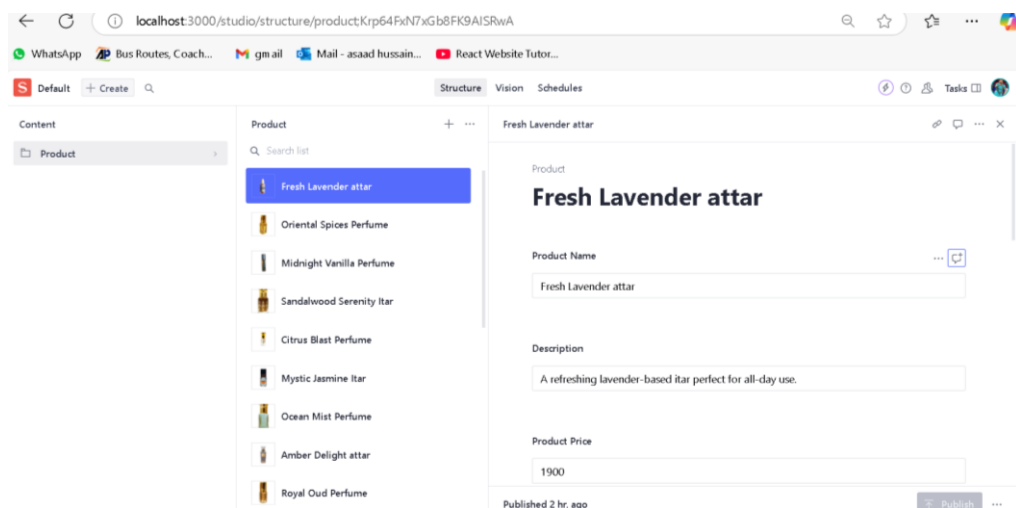


Top Picks For You

Find a bright ideal to suit your taste with our great selection of suspension, floor and table lights.



Populated sanity CMS fields



Code snippets for API integration and migration scripts

```
EXPLORER    ...    page.tsx ...\app M    page.tsx ...\shop M    TS getProducts.ts U    JS importData.mjs U X

FIGMA-APP
> .next
> Documentation
> node_modules
> public
> scripts
  JS importData.mjs U
  > src
    > app
    > components
    > context
    > data
    > sanity
  TS getProducts.ts U
$ .env.local
.eslintrc.json
.gitignore
TS next-env.d.ts

scripts > JS importData.mjs > importData > response
1  import { createClient } from '@sanity/client';
2  import axios from 'axios';
3  import dotenv from 'dotenv';
4  import { fileURLToPath } from 'url';
5  import path from 'path';
6
7  // Load environment variables from .env.local
8  const __filename = fileURLToPath(import.meta.url);
9  const __dirname = path.dirname(__filename);
10  dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12  // Create Sanity client
13  const client = createClient({
14    projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15    dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16    useCdn: false,
17    token: process.env.SANITY_API_TOKEN,
18    apiVersion: '2021-08-31'
19  });
```

```
EXPLORER    ...    page.tsx ...\app M    page.tsx ...\shop M    TS getProducts.ts U    JS importData.mjs U X

FIGMA-APP
> .next
> Documentation
> node_modules
> public
> scripts
  JS importData.mjs U
  > src
    > app
    > components
    > context
    > data
    > sanity
  TS getProducts.ts U
$ .env.local
.eslintrc.json
.gitignore
TS next-env.d.ts

scripts > JS importData.mjs > importData > response
21
22  async function uploadImageToSanity(imageUrl) {
23    try {
24      console.log('Uploading image: ${imageUrl}');
25      const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
26      const buffer = Buffer.from(response.data);
27      const asset = await client.assets.upload('image', buffer, {
28        filename: imageUrl.split('/').pop()
29      });
30      console.log('Image uploaded successfully: ${asset._id}');
31      return asset._id;
32    } catch (error) {
33      console.error('Failed to upload image:', imageUrl, error);
34      return null;
35    }
36  }
37
38  async function importData() {
39    try {
```

```
> .next
> Documentation
> node_modules
> public
> scripts
  JS importData.mjs U
  > src
    > app
    > components
    > context
    > data
    > sanity
  TS getProducts.ts U
$ .env.local
.eslintrc.json
.gitignore
TS next-env.d.ts

38  async function importData() {
39    try {
40      console.log('migrating data please wait...');
41
42      // API endpoint containing car data
43      const response = await axios.get('https://api-perfume.vercel.app/api/product');
44      const products = response.data.data;
45      console.log("products ==> ", products);
46
47      for (const product of products) {
48        let imageRef = null;
49        if (product.image) {
50          imageRef = await uploadImageToSanity(product.image);
51        }
52
53        const sanityProduct = {
54          _type: 'product',
55          product_name: product.product_name,
56
```

```
FIGMA-APP
> .next
> Documentation
> node_modules
> public
> scripts
  JS importData.mjs U
  > src
    > app
    > components
    > context
    > data
    > sanity
  TS getProducts.ts U
$ .env.local
.eslintrc.json
.gitignore
TS next-env.d.ts

scripts > JS importData.mjs > importData > response
38  async function importData() {
39
40    const sanityProduct = {
41      _type: 'product',
42      product_name: product.product_name,
43      description: product.description,
44      price: product.price,
45      priceAdiscount: product.priceAdiscount,
46      stock_quantity: product.stock_quantity,
47      category: product.category,
48      tags: product.tags,
49      size: product.size,
50      image: imageRef ? {
51        _type: 'image',
52        asset: {
53          _type: 'reference',
54          _ref: imageRef,
55        },
56      } : undefined,
57    };
58  }
```

VS Code Explorer shows the project structure: .next, Documentation, node_modules, public, scripts, src (app, components, context, data, sanity), TS getProducts.ts, .env.local, .eslintrc.json, .gitignore, next-env.d.ts, next.config.mjs, package-lock.json.

```

src > TS getProducts.ts > [0] getProducts > [0] products
1  import { client } from "../sanity/lib/client"
2
3
4  export const getProductsFour = async () => {
5      const products = await client.fetch(
6          `*[_type=="product"]{0..3}{
7              _id,
8              product_name,
9              price,
10             description,
11             size,
12             category,
13             tags,
14             stock_quantity,
15             "image_url":image.asset->url
16         }`
17     )
18     return products
19 }

```

```

4  export const getProductsFour = async () => {
19 }
20 export const getProducts = async () => {
21     const products = await client.fetch(
22         `*[_type=="product"]{0..15}{
23             _id,
24             product_name,
25             price,
26             description,
27             size,
28             category,
29             tags,
30             stock_quantity,
31             "image_url":image.asset->url
32         }`
33     )
34     return products
35 }

```

Terminal Output:

```

GET /studio/vision 200 in 25466ms
  o Compiling /blog ...
  ✓ Compiled /blog in 28.5s (7219 modules)
GET /shop 200 in 19502ms

```

```

1  import ProductCard from "@components/Producrcard/page";
2  import Link from "next/link";
3  import Navbar from "../components/Navbar";
4  import React from "react"
5  import { ProductData } from "@context/productData/context";
6  import { getProducts, getProductsFour } from "@getProducts";
7
8  interface Product {
9      _id: string;
10     product_name: string;
11     price: number;
12     description: string;
13     size: number;
14     category: string;
15     tags: string[];
16     stock_quantity: number;
17     image_url: string;
18 }
19

```

```

19
20 export default async function Home() {
21
22     const Products:Product[] = await getProductsFour()
23     //const products = useContext(ProductData)
24

```

```

20 export default async function Home() {
87   </div>
88   <div className="poppins font-[500] text-[16px] pt-4 text-center" style={{ color
89     <p>
90       Find a bright ideal to suit your taste with our great selection of sus
91     </p>
92   </div>
93   <div className="flex items-center justify-center pt-24 pb-3 gap-2">
94     {
95       Products.map((product) => (
96         <ProductCard key={product._id} product={product} />
97       ))
98     }
99   </div>
100
101   <div className="poppins font-[500] text-[24px] pt-6 ">
102

```