API Understanding for the Workflow

The workflow involves two main APIs: External API (to fetch data) and Sanity API (to store and manage data). Here's how each API is utilized in this process:

1. External API (Source of Product Data)

- **Purpose: The external API provides raw product data like:**
  - **Product**
  - **Price**
  - **Description**
  - **Image**
  - **Other Relevant Fields**
- **How It Works:**

  Axios sends a GET request to the external API endpoint, for example, https://fakestoreapi.com/products.

  The API returns a JSON response with the product data.

Workflow Description for Data Flow from API to Sanity and Displaying Data Using Context API

1. Transfering Data from API to Sanity (importSanityData.mjs)

- In this file named (importSanityData.mjs), I would manage the whole process of how data is to be transferred from an external API to Sanity, which is our CMS.

- In this file, I first started by setting up the Sanity client by using required project credentials: project Id, dataset, and API token.

- Then I fetched product data from the external API, which may include, for example: product name, description, price, image URL, etc., using Axios.

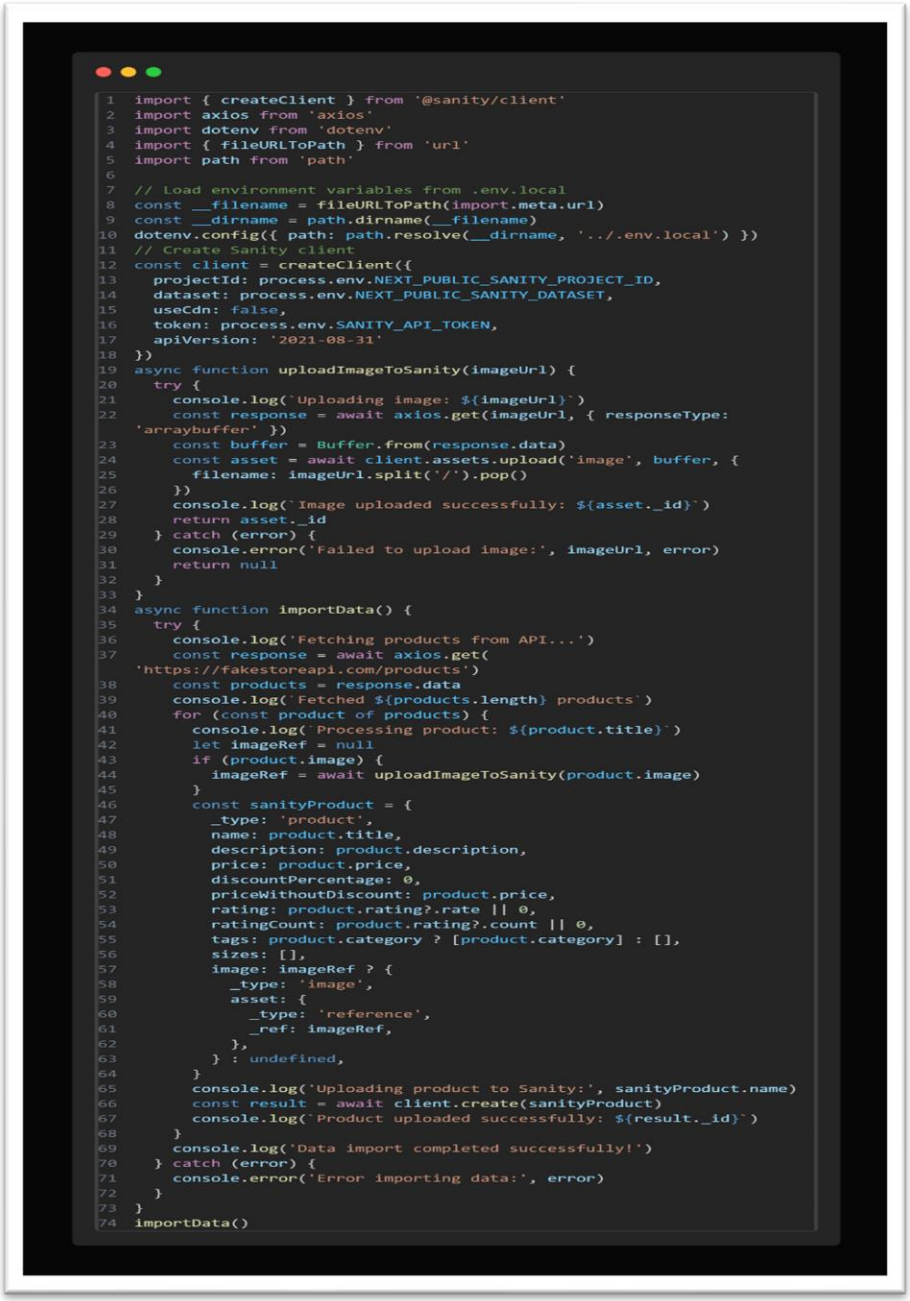
- I would upload the image for each product to Sanity via the asset upload feature. It returned a reference that was assigned to the data of the product.

- I had created each of the product objects based on the schema defined within Sanity and saved it to the Sanity dataset with the create method.

- Once I finished uploading all of the products, I logged messages to indicate successful completion of the process.

Transfering Data API to Sanity (CMS)

Input :

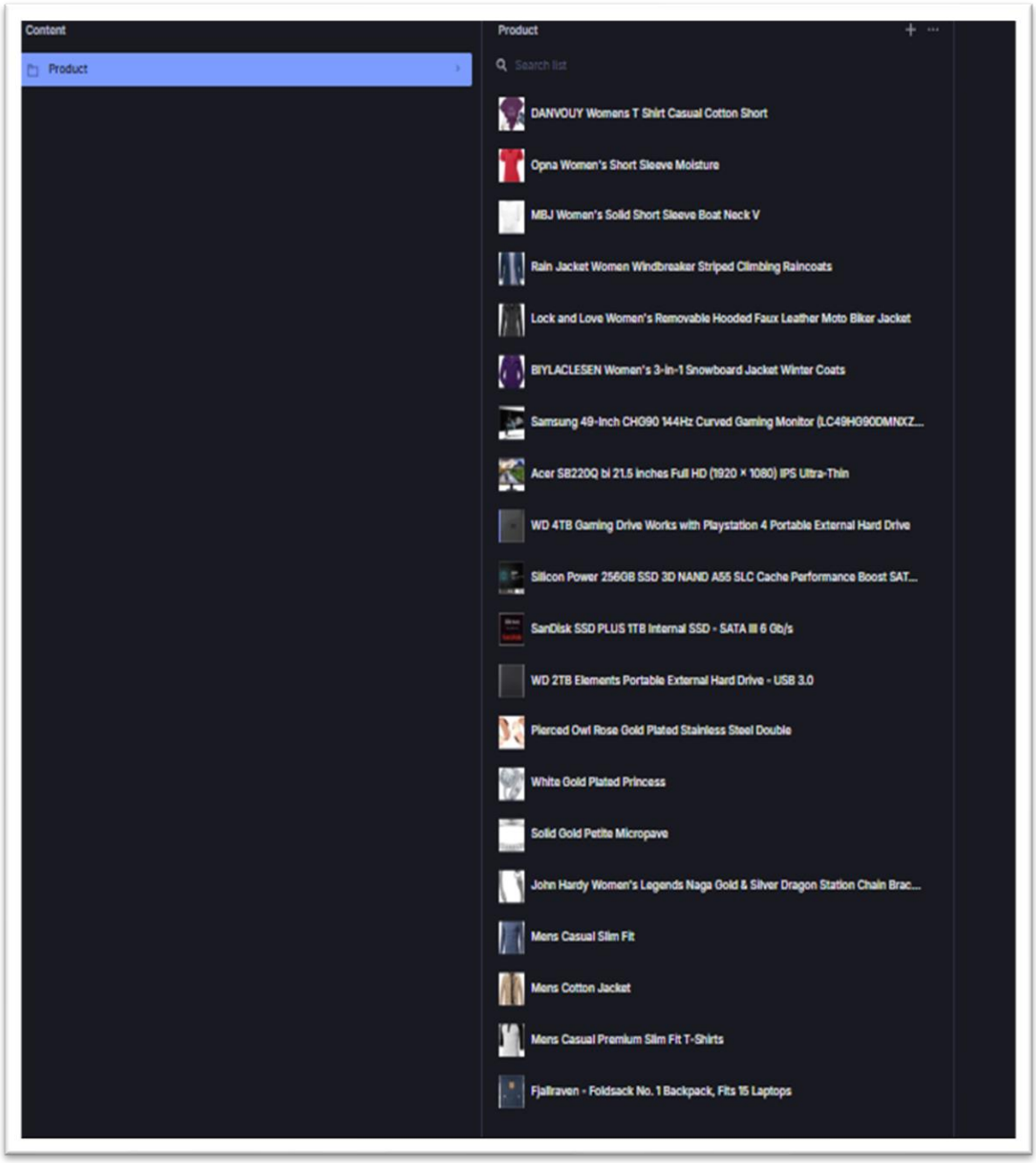


```
import { createClient } from '@sanity/client'
import axios from 'axios'
import dotenv from 'dotenv'
import { fileURLToPath } from 'url'
import path from 'path'

// 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: '2021-08-31'
})
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://fakestoreapi.com/products')
    const products = response.data
    console.log(`Fetched ${products.length} products`)
    for (const product of products) {
      console.log(`Processing product: ${product.title}`)
      let imageRef = null
      if (product.image) {
        imageRef = await uploadImageToSanity(product.image)
      }
      const sanityProduct = {
        _type: 'product',
        name: product.title,
        description: product.description,
        price: product.price,
        discountPercentage: 0,
        priceWithoutDiscount: product.price,
        rating: product.rating?.rate || 0,
        ratingCount: product.rating?.count || 0,
        tags: product.category ? [product.category] : [],
        sizes: [],
        image: imageRef ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        } : undefined,
      }
      console.log('Uploading product to Sanity:', sanityProduct.name)
      const result = await client.create(sanityProduct)
      console.log(`Product uploaded successfully: ${result._id}`)
    }
    console.log('Data import completed successfully!')
  } catch (error) {
    console.error('Error importing data:', error)
  }
}
importData()
```

Outputs :

Sanity User Interface



Product Data


```
[
  {
    discountPercentage: 0,
    rating: 4.7,
    name: 'Mens Cotton Jacket',
    description: 'great outerwear jackets for Spring/Autumn/Winter, suitable for many occasions, such as working, hiking, camp
ing, mountain/rock climbing, cycling, traveling or other outdoors. Good gift choice for you or your family member. A warm hear
ted love to Father, husband or son in this thanksgiving or Christmas Day.',
    image: 'https://cdn.sanity.io/images/6icj86y1/production/5519a5fa0fee4d3f36f7c3e3b9e4c81e501c8c61-679x755.jpg',
    price: 55.99
  },
  {
    name: 'Solid Gold Petite Micropave ',
    description: 'Satisfaction Guaranteed. Return or exchange any order within 30 days.Designed and sold by Hafeez Center in t
he United States. Satisfaction Guaranteed. Return or exchange any order within 30 days.',
    image: 'https://cdn.sanity.io/images/6icj86y1/production/74256ec87298d5e8303a2e92358c893738977af7-640x333.jpg',
    price: 168,
    discountPercentage: 0,
    rating: 3.9
  },
  {
    description: "Classic Created Wedding Engagement Solitaire Diamond Promise Ring for Her. Gifts to spoil your love more for
 Engagement, Wedding, Anniversary, Valentine's Day...",
    image: 'https://cdn.sanity.io/images/6icj86y1/production/d47de256189cf2cf23d2093674a881889b36a704-540x640.jpg',
    price: 9.99,
    discountPercentage: 0,
    rating: 3,
    name: 'White Gold Plated Princess'
  },
  {
    name: 'WD 2TB Elements Portable External Hard Drive - USB 3.0 ',
    description: 'USB 3.0 and USB 2.0 Compatibility Fast data transfers Improve PC Performance High Capacity; Compatibility Fo
rmatted NTFS for Windows 10, Windows 8.1, Windows 7; Reformatting may be required for other operating systems; Compatibility m
ay vary depending on user’s hardware configuration and operating system',
    image: 'https://cdn.sanity.io/images/6icj86y1/production/44c629d778597d3fa5fbbb61eee516a0cf1b1624-653x879.jpg',
    price: 64,
    discountPercentage: 0,
```

2. **Calling data from Sanity using Context API (Product data folder)**

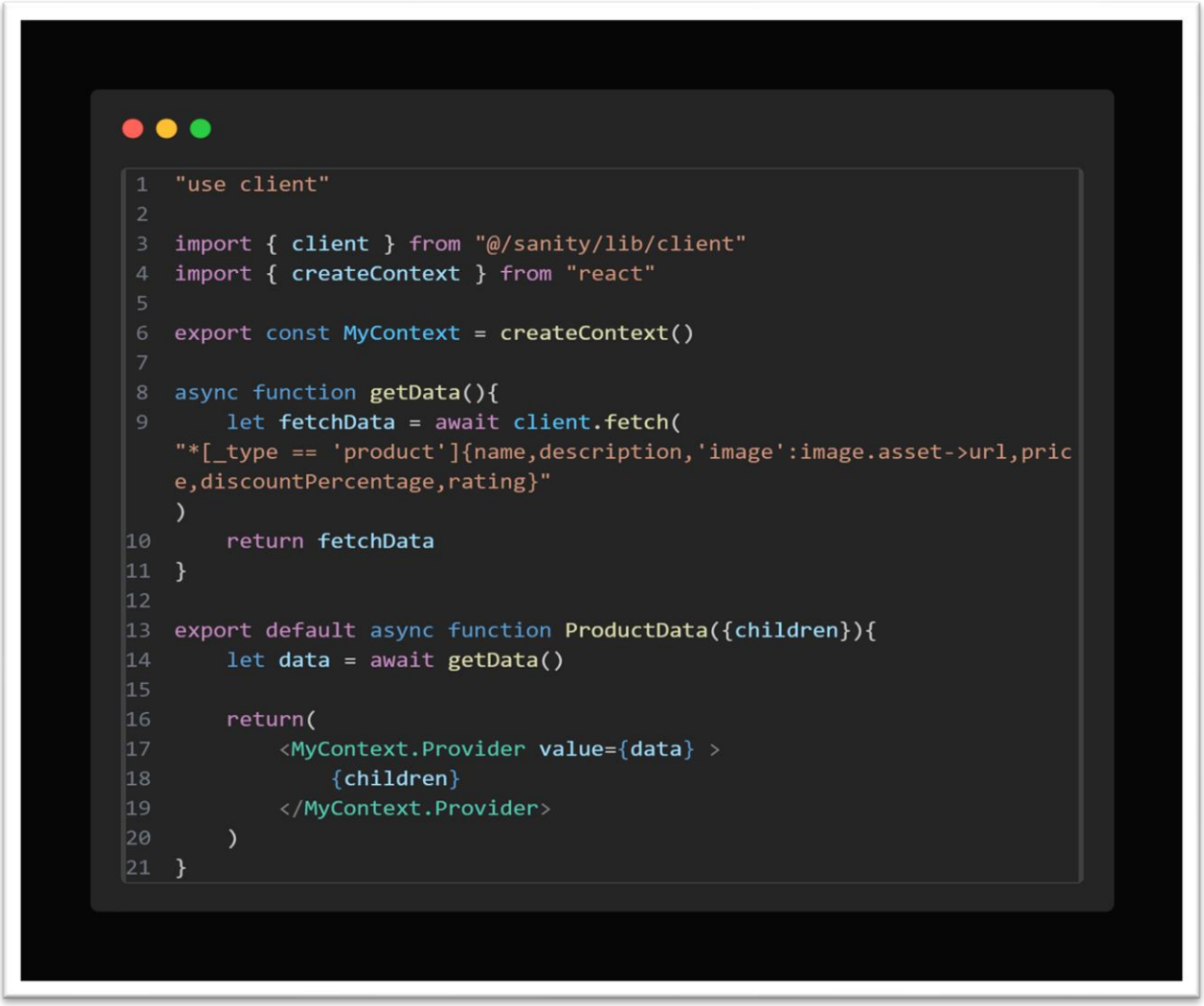
- I was making use of this function called getData() in the folder Product Data, which makes use of GROQ to fetch details for all products inside Sanity.

- For the products, the name, description, price , rating, and discount, not to mention images, were captured.

- This is an asynchronous function getData(). The retrieved data will be passed to the Context API

.

- I created a context using createContext() and stored the fetched product data in the context provider. This allowed me to share the product data across different components in my app without prop drilling.



```
"use client"

import { client } from "@/sanity/lib/client"
import { createContext } from "react"

export const MyContext = createContext()

async function getData(){
    let fetchData = await client.fetch(
"*[_type == 'product']{name,description,'image':image.asset->url,pric
e,discountPercentage,rating}"
)
    return fetchData
}

export default async function ProductData({children}){
    let data = await getData()

    return(
        <MyContext.Provider value={data} >
            {children}
        </MyContext.Provider>
    )
}
```

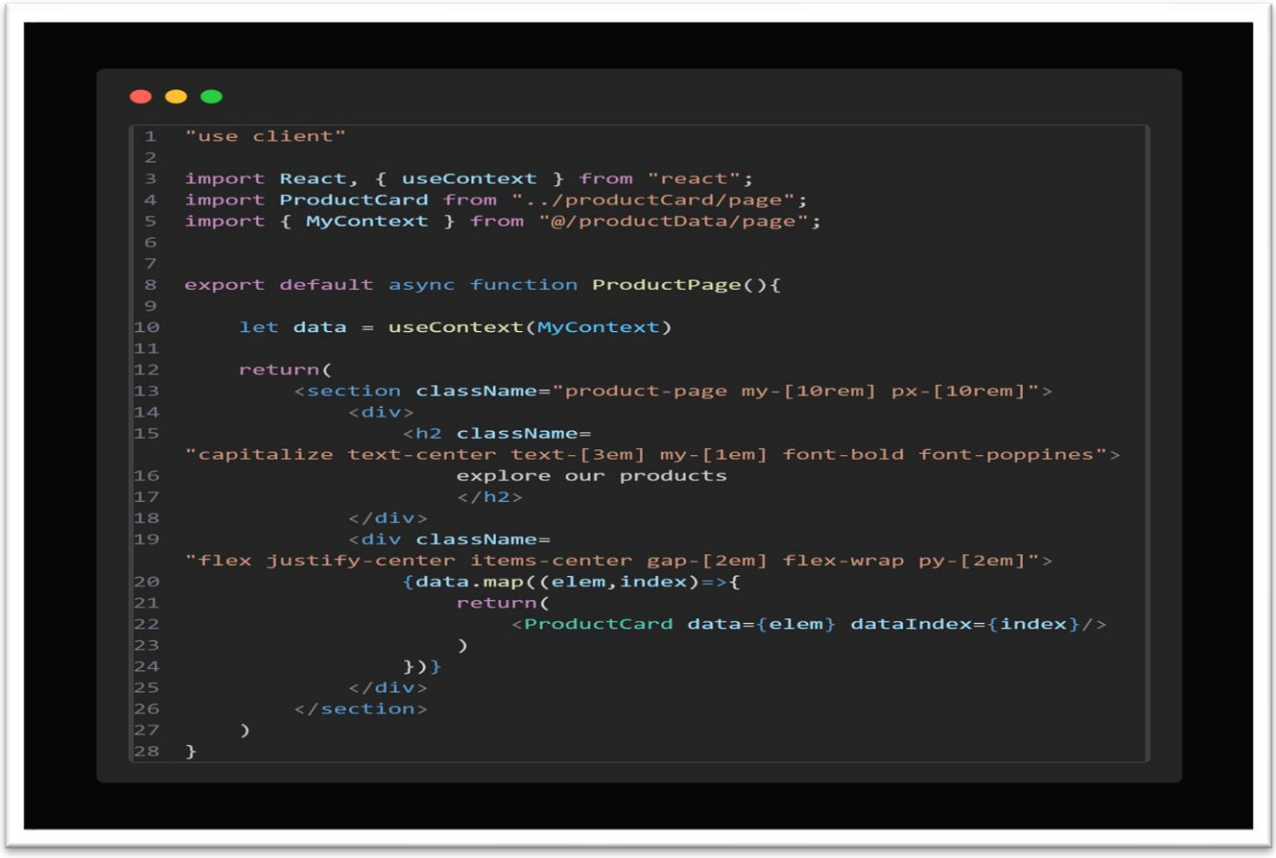
3. Display data using context in the product page folder.

- I accessed the fetched product data from Sanity in a folder called Product Page through using the Context API.

- In the Product Page component, I have consumed the context to get all the products. I applied product data mapping to display individual product cards. Each product card is rendered through the Product Card component.

- That Product Card component accepted as props the name, image, and price related to the details of the products, then depicted them in an attractive manner.

- This structure ensured the separation of concerns because data fetching and UI rendering were handled in different parts of the application.



```
"use client"

import React, { useContext } from "react";
import ProductCard from "../productCard/page";
import { MyContext } from "@/productData/page";


export default async function ProductPage(){

    let data = useContext(MyContext)

    return(
        <section className="product-page my-[10rem] px-[10rem]">
            <div>
                <h2 className=
"capitalize text-center text-[3em] my-[1em] font-bold font-poppines">
                    explore our products
                    </h2>
            </div>
            <div className=
"flex justify-center items-center gap-[2em] flex-wrap py-[2em]">
                {data.map((elem,index)=>{
                    return(
                        <ProductCard data={elem} dataIndex={index}/>
                    )
                })}
            </div>
        </section>
    )
}
```

Data show on UI User Interface