Hackathon3 _ Day 3

API Integration and Data Migration

Prepared by: Nida Tariq

Introduction:

Day 3 focuses on API Integration and Data Migration into Sanity CMS, a headless content management system. The goal of this process is to seamlessly fetch and display product data from an external API while dynamically populating the Sanity CMS database. Once integrated, the data is presented in a visually appealing format on a Next.js frontend.

The primary objective is to bridge the gap between an external API and Sanity CMS by successfully importing product data, such as images, descriptions, and prices. This integration ensures that the data is readily available for use in frontend applications.

Key Learning Outcomes

1. API Integration:

Learn how to retrieve data from external APIs and efficiently use it within a Next.js application.

2. Schema Modifications:

Understand how to modify and design schemas in Sanity CMS to ensure compatibility with the imported product data.

3. Data Migration:

Learn how to migrate product information into Sanity CMS using API calls. This includes importing assets such as images directly from external URLs into the CMS

4. Frontend Display:

Master the skill of displaying migrated data on a Next.js frontend in an organized and visually appealing format.

API Integration Process

1. API Integration:

The initial step involves integrating an external API into the Next.js application to fetch product data. The API endpoint used for this project is:

API Endpoint:

https://template-03-api.vercel.app/api/products

This API provides a list of products, where each product contains essential details such as:

Name: The title or name of the product.

Price: The cost of the product.

Description: A brief overview of the product features.

Image URL: A link to the product's image hosted online.

We used axios to fetch the data and then set it in the React state for rendering.

API Request Setup:

We use axios to make API requests to fetch product data from an endpoint.

```
import axios from 'axios';

async function fetchData() {

  try {

    const response = await axios.get('https://template-03-api.vercel.app/api/products');

    const products = response.data.data;

    console.log("Products fetched:", products);

  } catch (error) {

    console.error('Error fetching product data:', error);
```

```
 }
}
fetchData();
```

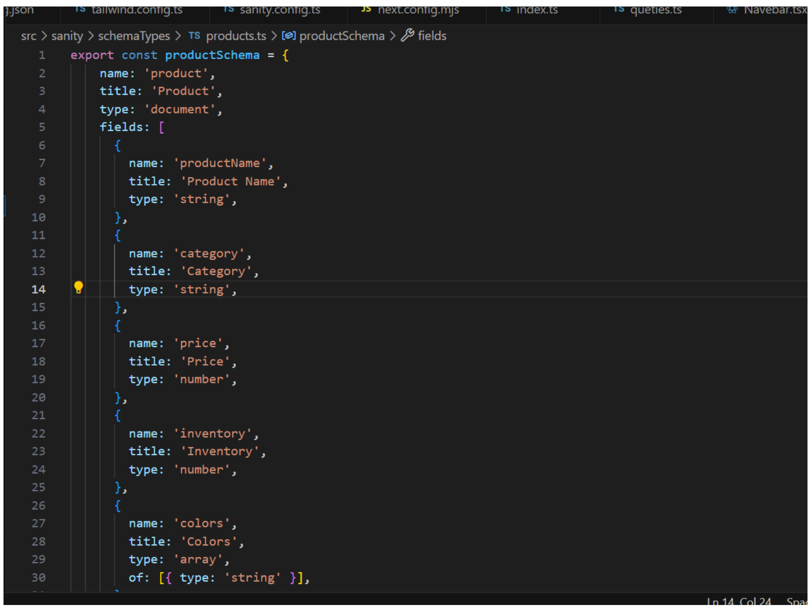
This function sends a GET request to the API, retrieves the product data, and logs it.

The fetched product data is later used to populate the Sanity CMS.

Schema Adjustments:

For the product data to be correctly migrated and stored in Sanity CMS, we adjusted the Sanity schema to match the product fields coming from the API. We made sure that the data structure aligns with the expected format in Sanity, such as productName, category, price, description, and image.

Product Schema:



```
export const productSchema = {
    name: 'product',
    title: 'Product',
    type: 'document',
    fields: [
      {
        name: 'productName',
        title: 'Product Name',
        type: 'string',
      },
      {
        name: 'category',
        title: 'Category',
        type: 'string',
      },
      {
        name: 'price',
        title: 'Price',
        type: 'number',
      },
      {
        name: 'inventory',
        title: 'Inventory',
        type: 'number',
      },
      {
        name: 'colors',
        title: 'Colors',
        type: 'array',
        of: [{ type: 'string' }],
```

Migration Steps and Tools Used

1. Fetching Data:

Used the axios library to fetch data from an external API.

2. Image Upload:

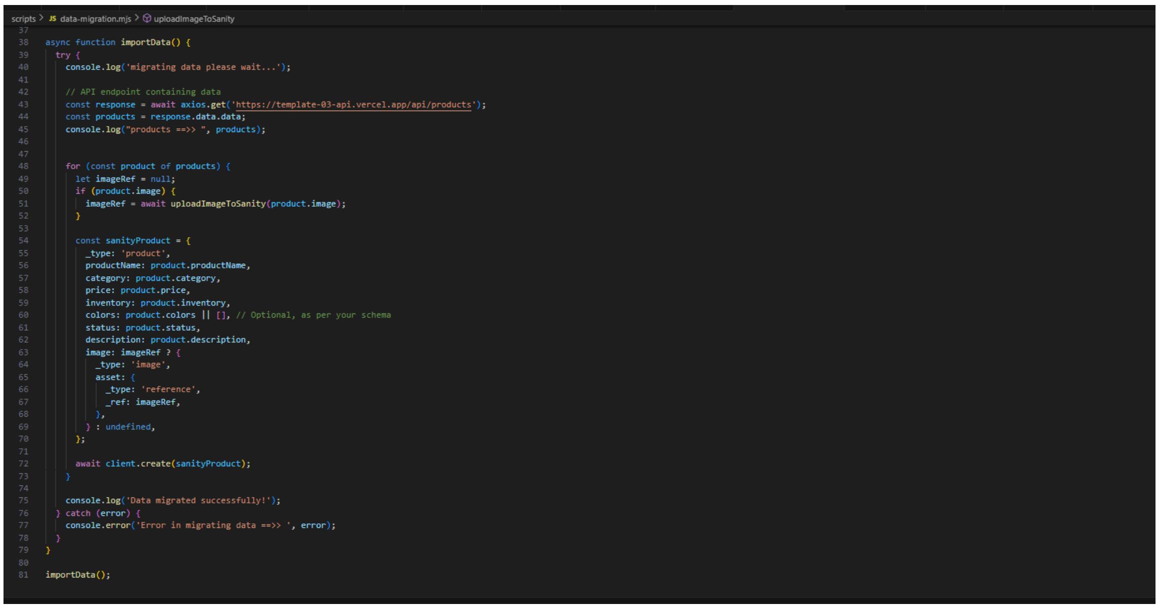
Fetched images as buffers and uploaded them to Sanity using the Sanity client.

3. Data Migration:

Iterated over the fetched product data.

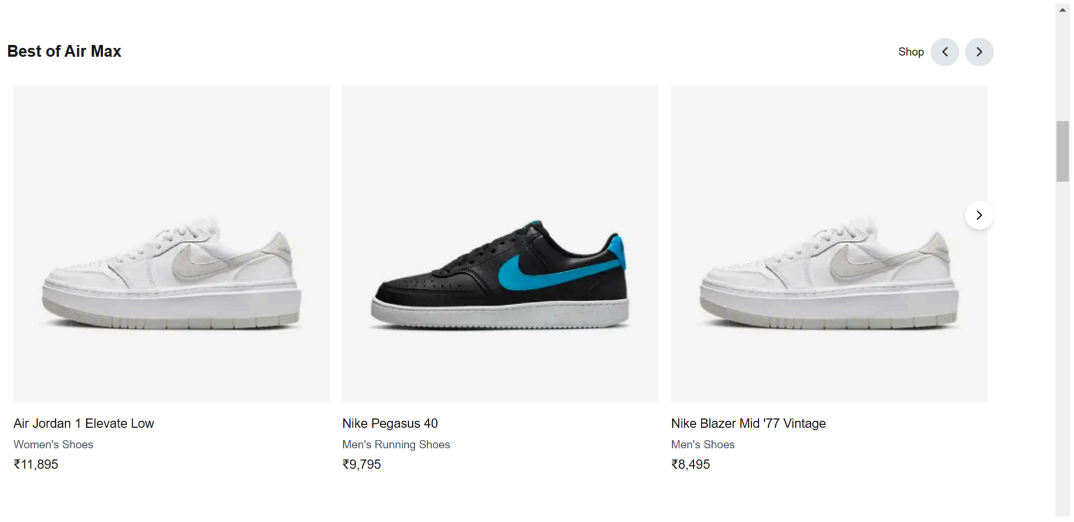
4. Sanity Client:

Used Sanity's client library (@sanity/client) to interact with the CMS.

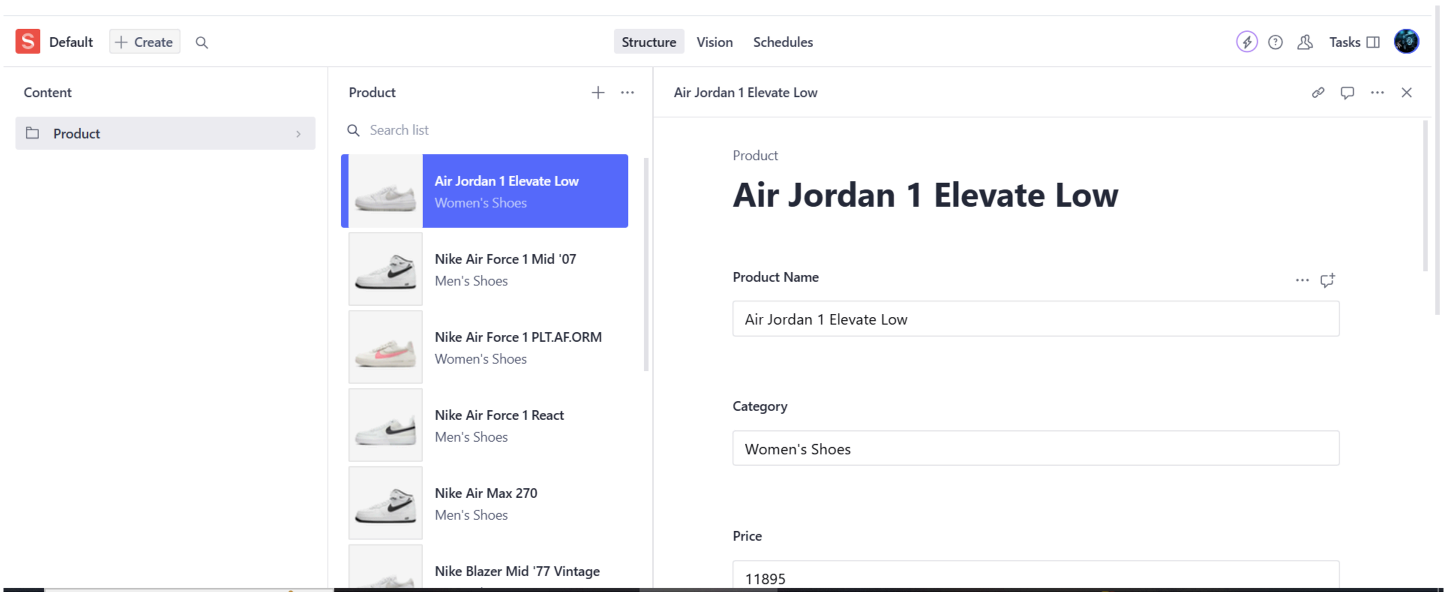


```javascript

async function importData() {
  try {
    console.log('migrating data please wait...');

    // API endpoint containing data
    const response = await axios.get('https://template-03-api.vercel.app/api/products');
    const products = response.data.data;
    console.log("products ==>> ", products);


    for (const product of products) {
      let imageRef = null;
      if (product.image) {
        imageRef = await uploadImageToSanity(product.image);
      }

      const sanityProduct = {
        _type: 'product',
        productName: product.productName,
        category: product.category,
        price: product.price,
        inventory: product.inventory,
        colors: product.colors || [], // Optional, as per your schema
        status: product.status,
        description: product.description,
        image: imageRef ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        } : undefined,
      };

      await client.create(sanityProduct);
    }

    console.log('Data migrated successfully!');
  } catch (error) {
    console.error('Error in migrating data ==>> ', error);
  }
}

importData();
```

Frontend Display:



Populated Sanity CMS Fields



Air Jordan 1 Elevate Low
Women's Shoes

Nike Air Force 1 Mid '07
Men's Shoes

Nike Air Force 1 PLT.AF.ORM
Women's Shoes

Nike Air Force 1 React
Men's Shoes

Nike Air Max 270
Men's Shoes

Nike Blazer Mid '77 Vintage

Air Jordan 1 Elevate Low

Product

Air Jordan 1 Elevate Low

Product Name

Air Jordan 1 Elevate Low

Category

Women's Shoes

Price

11895