Day 3 Of Hackathon

API Integration and Data Migration Report

Prepared by: **Mahnoor Adnan**

Introduction

This document outlines the API integration and data migration processes undertaken on Day 3 of the development for the marketplace project. The backend leverages Sanity CMS for content management, ensuring scalability, flexibility, and omnichannel capabilities. The integration of external APIs and the migration of structured data enhance cross-platform synchronization and backend efficiency.

API Integration Process

Objective

To connect the marketplace backend to external APIs, allowing real-time data retrieval and ensuring seamless integration with Sanity CMS for dynamic content management.

1.API Endpoint Analysis:

- Reviewed API documentation from the external source.
- Identified `/products` endpoint for retrieving product data.
- Verified the structure of API responses using Postman.

2.Data Migration Process

```
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_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('migrating data please wait...');

    // API endpoint containing car 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();
```

Objective

To populate Sanity CMS with structured and validated data retrieved from external APIs.

Steps Executed:

- Firstly Utilized @sanity/client for seamless CMS interaction.
- Handled image migration by uploading them to Sanity as assets.
- Used Axios for fetching and buffering images
- Created utility functions to fetch data
- Implemented error handling and fallback mechanisms to ensure robustness.
- Verified API responses for accuracy and completeness.
- Debugged API issues using Postman.
- Wrote a migration script to fetch and transform API data before importing it into Sanity CMS.
- Ensured migrated data adhered to schema constraints.
- Logged errors and re-ran the script for incomplete data.

Outcome

Data migration successfully populated Sanity CMS with validated and structured product data, ensuring alignment with frontend requirements.

3.Schema Adjustments

```
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' }],
    },
    {
      name: 'status',
      title: 'Status',
      type: 'string',
    },
    {
      name: 'image',
      title: 'Image',
      type: 'image',
      options: {
        hotspot: true,
      },
    },
    {
      name: 'description',
      title: 'Description',
      type: 'text',
    },
  ],
}
```

Objective

To align the external API data structure with the Sanity CMS schema for seamless data migration and backend functionality.

Adjustments Made

1.Field Name Alignment:

- Renamed product_title to productName for consistency.
- Added fields such as price, inventory, and colors to the Sanity schema**.**

2. Image Field Enhancement:

- Utilized Sanity's image type with hotspot options for better frontend rendering.

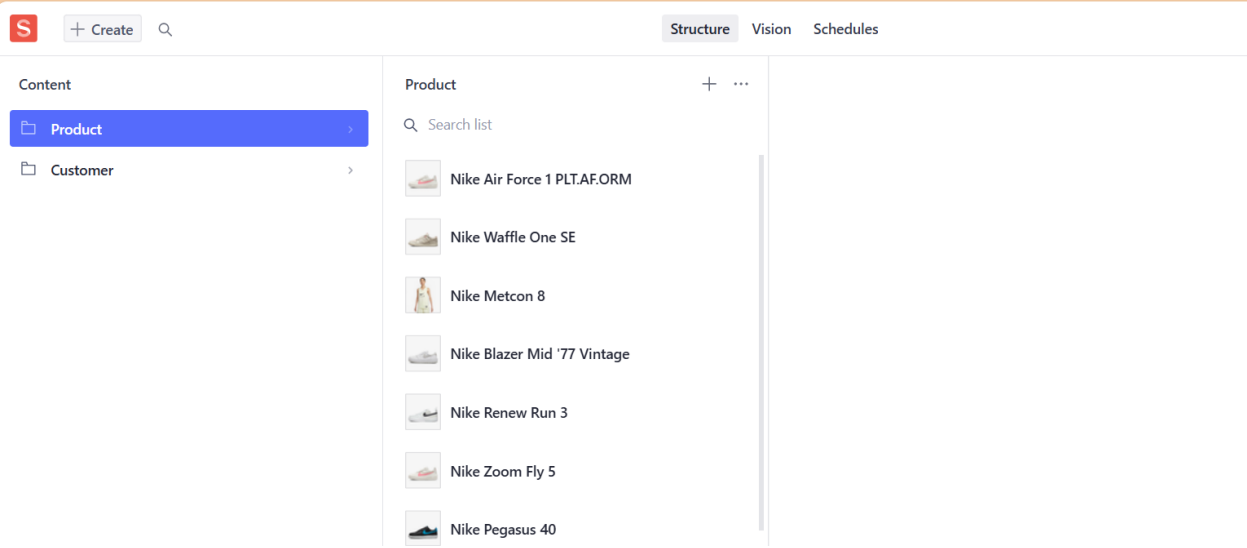
3.Optional Fields:

- Ensured optional fields, such as colors, handled undefined or empty values gracefully.

Outcome

Schema adjustments ensured compatibility between API data and Sanity CMS, facilitating smooth data migration and enhancing backend scalability.

4.Populated Sanity CMS Fields



Once the migration script is executed, the product data fetched from the API is directly stored in the corresponding fields of the Sanity CMS. This automation prevents the need for manual data entry, saving time and reducing errors.

Example:

In the Sanity Studio interface (running on localhost:3000/studio), you can see the following populated fields for each product:

- **Product Name:** The product name fetched from the API.
- **Category:** The category of the product as specified in the API data.
- **Price:** The numerical price of the product.
- **Inventory:** The stock quantity available.
- **Colors:** The array of color variants for the product.
- **Status:** The status of the product (e.g., available, out of stock).
- **Description:** A text description of the product.
- **Image:** The uploaded product image automatically linked to the Sanity CMS.

5.Frontend Integration

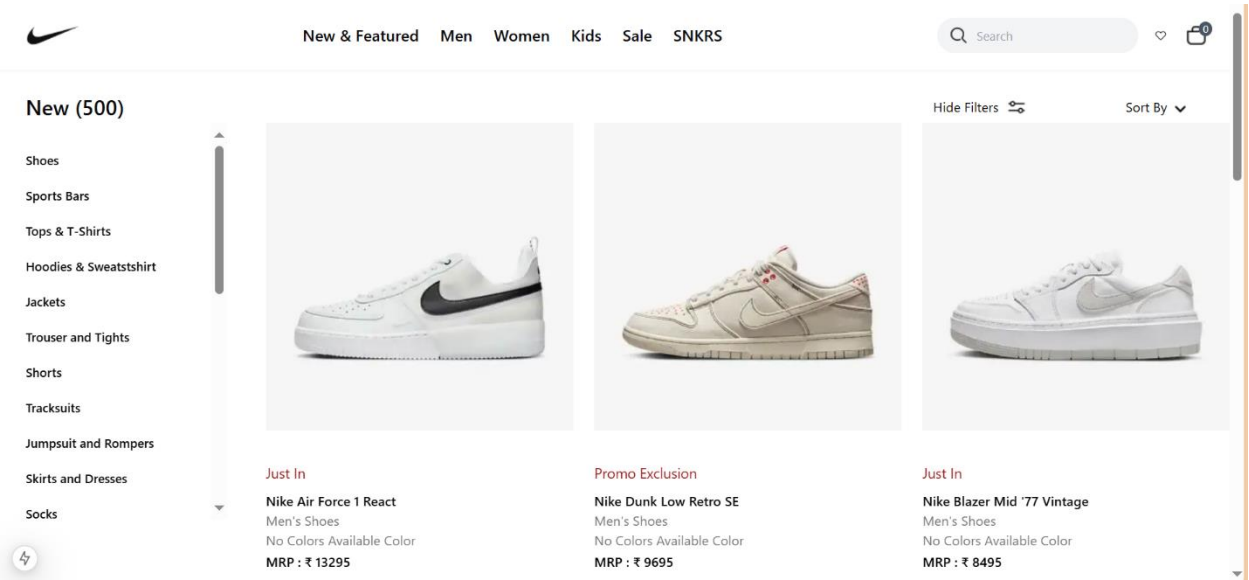
Objective

To dynamically display migrated data in the marketplace frontend using GROQ queries.

Steps Executed

- ✓ **Used GROQ queries to fetch data from Sanity CMS:**

```
export async function getProducts() {
  const query = '*[_type == "product"]';
  const products = await client.fetch(query);
  return products;
}
```



Outcome

The frontend successfully displayed migrated data, ensuring a dynamic and user-friendly marketplace experience.

Benefits :

1. Scalability:
    - Omnichannel capabilities ensure seamless integration of future APIs.
    - Centralized management in Sanity CMS facilitates cross-platform synchronization.
2. Efficiency:
    - Automated migration reduces manual effort and minimizes errors.
    - Dynamic rendering enhances user experience with real-time updates.
3. Flexibility:
    - Schema adaptability allows for future modifications with minimal overhead.

Conclusion

The API integration and data migration process successfully demonstrated the project's scalability and efficiency. By leveraging Sanity CMS and external APIs, the marketplace backend achieved real-time synchronization and dynamic frontend rendering, setting a strong foundation for future enhancements.