Day 3 - API Integration Report for Shop.co [Clothing Brand]

Overview

On Day 3, the focus was to integrate APIs into the shop.co [clothing brand] and populate Sanity CMS with product data sourced from a local API. This report documents the API integration process, schema adjustments, data migration steps, and the tools utilized. Screenshots and code snippets are included to provide a comprehensive understanding.

API Integration Process

1. **Data Source**
   Data was fetched from the API endpoint: https://template1-neon-nu.vercel.app/api/products
   - **Fields included:** Name, description, price, images, inventory, categories, and status.
2. **Integration Steps**
   - **Schema Design**: Created a custom schema for cloth brand in Sanity CMS to match the data structure.
   - **Import Script**: Developed a script to fetch product data, process it, and upload it to Sanity CMS.
   - **Image Handling**: Implemented logic to handle multiple product images and upload them as unique assets in Sanity CMS.
   - **Data Validation**: Ensured all fields like slug, price, and stock level adhered to validation rules.
   - **API Endpoints**: Created endpoints to fetch data from Sanity CMS for frontend usage

Schema Adjustments

Schema adjustments refer to modifications made to the structure of a database or data model to enhance functionality, improve efficiency, or accommodate new requirements. For the Shop.co clothing brand website, schema adjustments may involve refining product categories, optimizing inventory management, integrating customer preferences, or improving order processing. These changes ensure seamless data flow, enhance user experience, and support business growth by adapting to evolving market needs.

Schema Example

```
import { defineType } from "sanity"

export default 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:"new",
            type: 'boolean',
            title:"New",
        },
        {
            name:"colors",
            title:"Colors",
            type: 'array',
            of:[
                {type: 'string'}
            ]
        },
        {
            name:"sizes",
            title:"Sizes",
            type: 'array',
            of:[
                {type: 'string'}
            ]
        }
    ],
})
```

Migration Steps

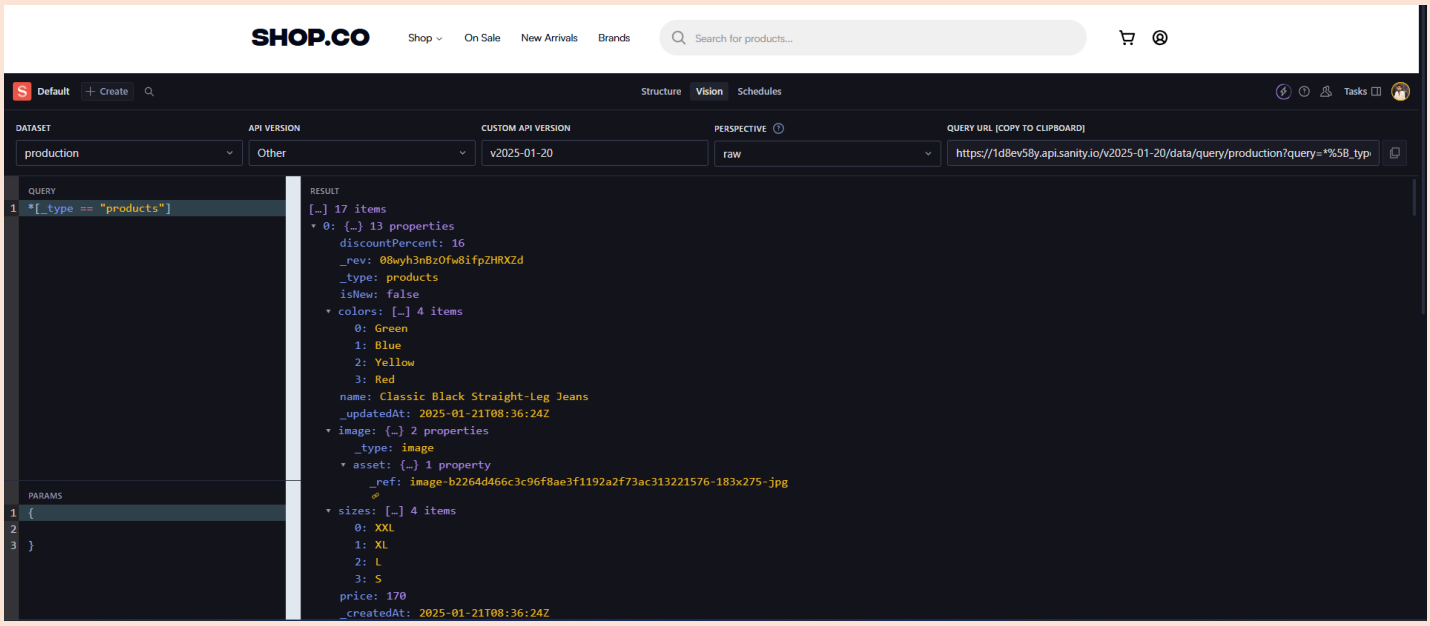
1. **Tools Used**
    - **Sanity Client**: For uploading data to Sanity CMS.
    - **Fetch**: For making API calls.
    - **.env**: To secure API keys and environment variables.

Migration Script Example

```
import { createClient } from '@sanity/client';
import fetch from 'node-fetch';

const client = createClient({
  projectId: "1d8ev58y",
  dataset: "production",
  useCdn: true,
  apiVersion: '2025-01-13',
  token: "sk1bEp5CJrSJpZyIis8TxKxAdvpDyw0ay2o7sVPWoCuuhT1J6cPOysiLk31eyeI4MEKAqN9phfmqGweO56tm8E3Y4u0NBO5XC9GYNVRS8ANu27By5XNbCceNvT8Nh5U3QMzvytP8yy9827lh331V6CUh9CQ2WIZnY1Q5dEsiCh7cMNMNHy5S",
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);

    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      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 uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: 'products',
        name: product.name,
        description: product.description,
        price: product.price,
        image: {
          _type: 'image',
          asset: {
            _ref: imageId,
          },
        },
        discountPercent: product.discountPercent,
        isNew: product.isNew,
        colors: product.colors,
        sizes: product.sizes,
      };

      const createdProduct = await client.create(document);
      console.log(`Product ${product.name} uploaded successfully:`, createdProduct);
    } else {
      console.log(`Product ${product.name} skipped due to image upload failure.`);
    }
  } catch (error) {
    console.error('Error uploading product:', error);
  }
}

async function importProducts() {
  try {
    const response = await fetch('https://template1-neon-nu.vercel.app/api/products');

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error('Error fetching products:', error);
  }
}

importProducts();
```

API Response Validation

- Screenshot of API responses tested in Postman or browser.



Conclusion

The integration established a seamless process for importing, validating, and displaying cloths product data from the local API into Sanity CMS. Custom schemas and migration scripts ensured data consistency and project alignment. Future steps include enhancing API queries and implementing advanced features like search and filtering.