Day 3 - API Integration Report - General E-commerce Website

Prepared By: Fatima Mudassir

1) API Integration Process

Overview: This document outlines the API integration process, focusing on fetching and migrating product data from an external API into the backend system for the general e-commerce website. The external API used is available at: https://template6-six.vercel.app/api/products.

Steps Followed:

A)
Understanding API Documentation:

Reviewed the external API's available endpoints.
Key endpoint used: /api/products, which provides a list of products with their details such as title, price, image URL, tags, discount percentage, description, and more.

B)
Authentication Setup:

The integration requires an API token for accessing the backend and performing data operations. Authentication was handled using the token in the request headers for secure data fetching.
Token used: ~your api token

C)
Endpoint Testing:

The external API was tested using Postman to ensure the endpoints were returning the correct data structure.
Successful response from /api/products endpoint verified data such as product title, price, description, and image URL.

D)
Backend Integration:

Implemented functions in the backend to fetch data from the API and process it for integration into the database.
Data was processed and images were uploaded to Sanity CMS using the following process:
Fetching the image from the provided URL.
Converting the image into a buffer and uploading it to Sanity CMS using the @sanity/client API.

Created reusable functions like uploadImageToSanity() and uploadProduct() to automate the process of product data upload, including the image and product details.

2. Schema Adjustments
Since products and images are being stored in Sanity CMS, the schema was adjusted to support the required data structure.

Product Schema Updates:
Added fields for title, price, image, tags, discount, description, and isNew.
Example schema: javascript
Copy
Edit export default
{   name:
'product',   title:
'Product',   type:
'document',
fields: [{
name: 'title',
title: 'Title',
type: 'string',
    },
    {

```
    name: 'price',
title: 'Price',      type:
'number',
    },
    {
    name: 'productImage',
title: 'Product Image',      type:
'image',
    },
    {
    name: 'tags',
title: 'Tags',      type:
'array',      of: [{ type:
'string' }],
    },
    {
    name: 'discountPercentage',
title: 'Discount Percentage',      type:
'number',
    },
    {
    name: 'description',
title: 'Description',      type:
'text',
    },
    {
    name: 'isNew',
title: 'Is New?',
type: 'boolean',
    },
  ],
};
```

This schema defines how product data is structured in Sanity. 3)

Migration Steps and Tools Used

To integrate data into my project, I followed a systematic process for migrating product data from an external API to Sanity CMS:

Data Fetching: I first fetched product data from an external API endpoint (https://template6-six.vercel.app/api/products) using the fetch API. This step allowed me to get a list of products, including their titles, images, prices, tags, and other attributes.

Image Upload: Each product contains an image URL, which needed to be uploaded to Sanity CMS. I used the fetch API to fetch the image and converted it into a Buffer for uploading to Sanity. This was done using the client.assets.upload() method to upload the images as assets to the CMS.

Product Document Creation: After successfully uploading the images, I created a document for each product in Sanity. The document included information such as the title, price, image reference, tags, discount percentage, and other relevant details. The image reference was added as a relation to the image asset uploaded in the previous step.

Data Validation: After uploading all the product data, I ensured that all products were correctly displayed in the frontend by checking the data in Sanity Studio. I also verified that the images were correctly linked to the product entries.

Tools Used:
Sanity CMS: Sanity CMS was used as the destination for storing and managing the product data, including images and other metadata.

Fetch API: The fetch API was used to retrieve product data from an external source and also to fetch images for upload.
Buffer and Blob: These tools were used to convert the image data into a format that could be uploaded to Sanity CMS as an asset.

Sanity Client SDK: The official Sanity client SDK (@sanity/client) was used for connecting to the Sanity CMS API, uploading assets, and creating documents in the CMS.

Challenges & Solutions

API authentication issues   Used a secure API token (should be stored in environment variables).
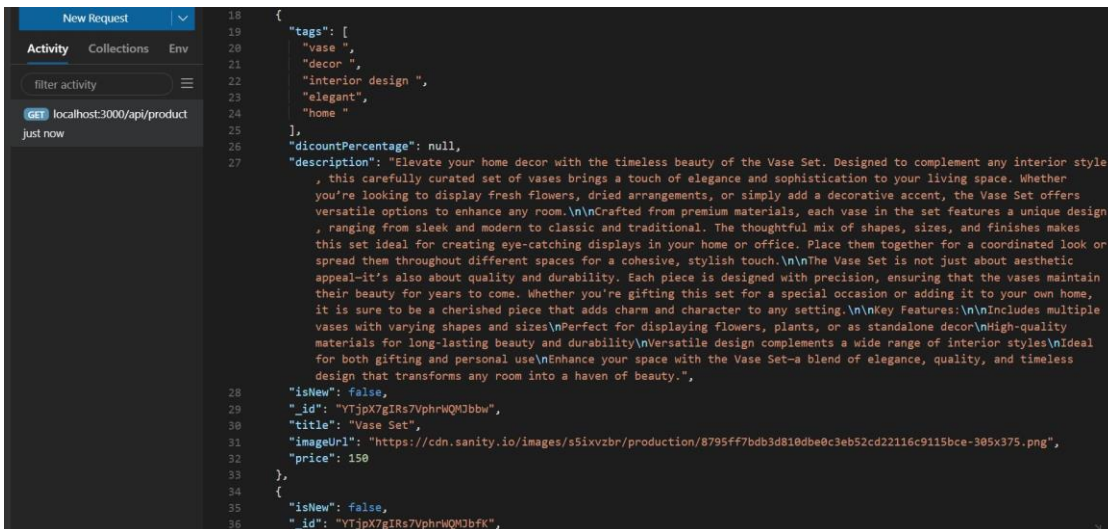
CORS policy errors   Configured CORS settings in Sanity dashboard.

Slow image uploads Implemented proper error handling and logging.

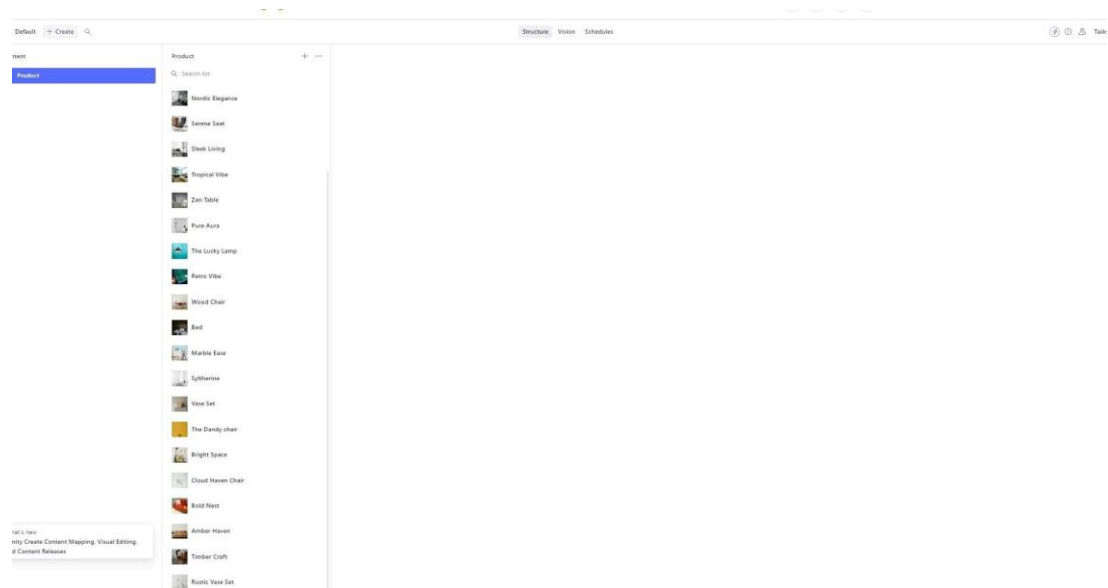Incorrect data format Used data transformation before saving to Sanity.
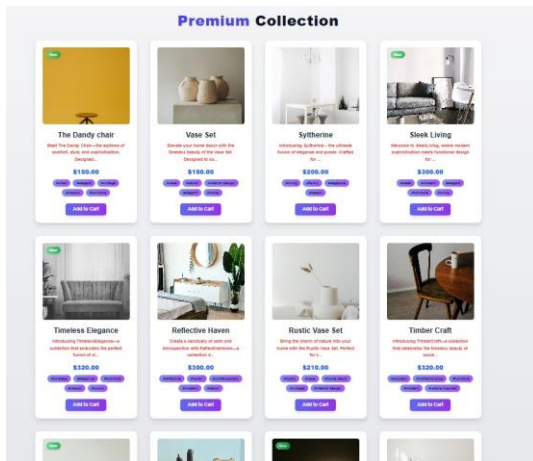
SCREENSHOTS

API CALLS

POPULATED SANITY CMS FIELD

DISPLAY FRONTEND

(Code Snippets for API Integration and Migration Scripts

CODE SNIPPETS FOR API

API Integration (Fetching Product Data):
This is the code used to fetch product data from the external API and send it to Sanity CMS:

```
// Fetching product data from an external API const response = await
fetch('https://template6-six.vercel.app/api/products'); if (!response.ok) {
throw new Error(`HTTP error! Status: ${response.status}`);
}
```

const products = await response.json(); console.log('Fetched
Products:', products); Image Upload (Sanity API
Integration):
This snippet demonstrates how I uploaded product images to Sanity CMS:

```
// Function to upload image to Sanity async function
uploadImageToSanity(imageUrl) {   const response =
await fetch(imageUrl);   if (!response.ok) {     throw new
Error(`Failed to fetch image: ${imageUrl}`);
  }

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

  const asset = await client.assets.upload('image', buffer, {
filename: imageUrl.split('/').pop(),
  });

  return asset._id;
}
```

MIGRATION SCRIPT

Final Import Function (Migration Script for Products):
This function is used to import the products and upload them to Sanity

```
async function importProducts() {   const response = await
fetch('https://template6-six.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);
  }
}

// Execute the import importProducts();
```