Day 3 - API Integration Report - General E-Commerce

API Integration Process

1. Overview of the API Integration

The API integration was intended to fetch product data from the Provided API and display it dynamically in the frontend. Key endpoints included "/products" for fetching product details. The integration made data synchronization between the API and the frontend components seamless.

2. Step-by-Step Integration

Step 1: select API endpoints.

Step 2: Setting up environment variables for API keys and other sensitive information.

Step 3: Implementing TypeScript interfaces to define the shape of data.

Step 4: Writing API calls using fetch.

Step 5: Error handling with retries and fallbacks.

Step 6: Connecting data to frontend

Add Changes Made to Schemas

Changes Made to Schemas

1. Changes to Sanity Schema

- No modifications were performed on the sanity schema in regard to the API integration.

Migration Steps and Tools Used

1. Tools Used for Migration

Describe all tools used within the migration steps

2. Migration

The migration utilized a given script to transfer as well as convert data when applicable. The Steps were:

Step 1: Run the script for migration based on the correct schema structure according to the specifications.

Step 2: Validate the output data to confirm successful transformation.

Step 3: Test the migrated data within the development environment.

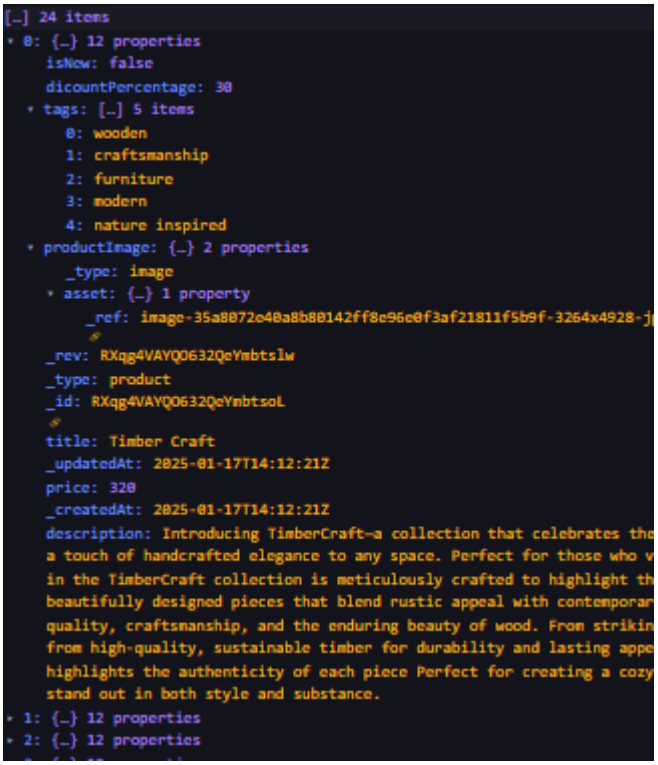
Step 4: Deploy the migration script in the production environment.

Step 5: Verify the migration by comparing the old and new data sources.

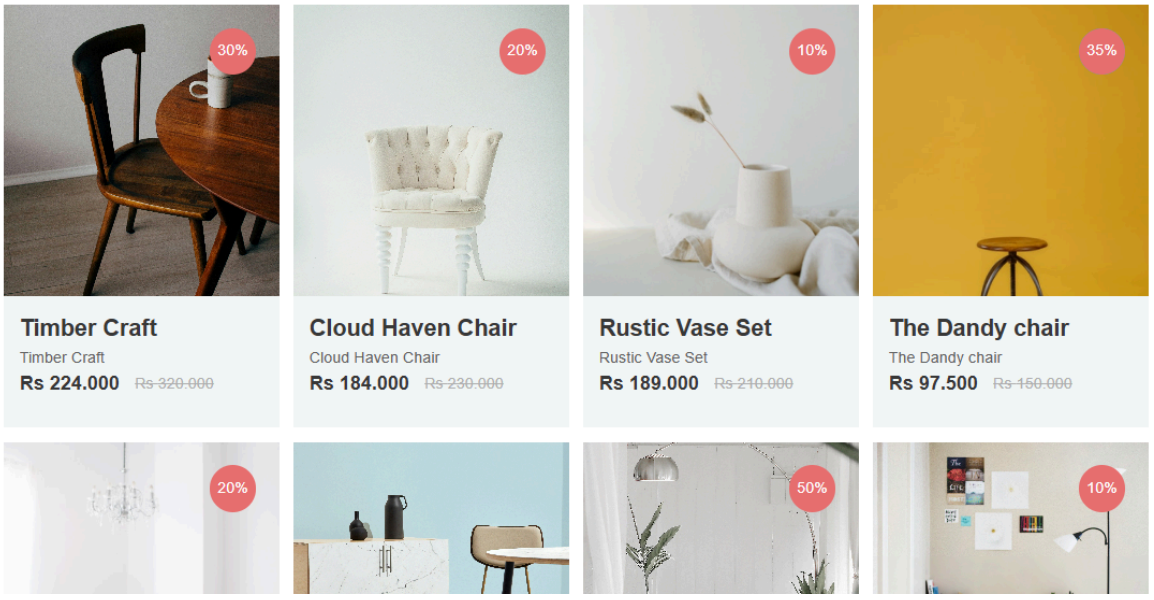
The script provided above made the migration much easier and reduced manual intervention, thereby minimizing errors.

Screenshots

1. API Calls



2. Data Display in Frontend



3. Sanity CMS Fields

```
import { defineType } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        {
            name: "title",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        },
        {
            name:"description",
            type:"text",
            validation: (rule) => rule.required(),
            title:"Description",
        },
        {
            name: "productImage",
            type: "image",
            validation: (rule) => rule.required(),
            title: "Product Image"
        },
        {
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        },
        {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }]
        },
        {
            name:"dicountPercentage",
            type:"number",
            title:"Discount Percentage",
        },
        {
            name:"isNew",
            type:"boolean",
            title:"New Badge",
        }
    ]
})
```

Code Snippets

1. API Integration Code

Provide snippets of:

- Fetching data from the API.
- Parsing and validating API responses.
- Handling errors and retries.

```
"use client";
import React, { createContext, useContext, useState, useEffect } from "react";
import { client } from "../sanity/lib/client";

const ProductContext = createContext<any[]>([]);

export const ProductProvider: React.FC<{ children: React.ReactNode }> = ({
  children,
}) => {
  const [products, setProducts] = useState<any[]>([]);

  useEffect(() => {
    const fetchProducts = async () => {
      try {
        const res =
          await client.fetch(`*[_type == "product"] {
          title,
            description,
            "imageUrl": productImage.asset->url,
            price,
            tags,
            dicountPercentage,
            "discountedPrice": price - (price * dicountPercentage / 100),
            isNew,
            "slug": slug.current
        }`);
        setProducts(res);
      } catch (error) {
        console.error("Error fetching products:", error);
      }
    };

    fetchProducts();
  }, []);

  return (
    <ProductContext.Provider value={products}>
      {children}
    </ProductContext.Provider>
  );
```

2. Migration Script Code


importData.js > client > projectId
import { createClient } from '@sanity/client';
const client = createClient({
projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
dataset: 'production',
useCdn: true,
apiVersion: '2025-01-13',
token: process.env.Token,
});
Tabnine | Edit | Test | Explain | Document
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;
}
}
Tabnine | Edit | Test | Explain | Document
async function uploadProduct(product) {