DAY-3: API INTEGRATION AND DATA MIGRATION IN SANITY

Present by Nazia Siraj

GICIC 00443199

Friday 9am-12pm

Mentor Sir Humza Syed

MarketPlace-Builder Hackathon

Day-3: API Integration and Data Migration in Sanity

EXPLANATION OF MIGRATING DATA FROM API TO SANITY IN NEXTJS

The focus for Day 3 of the hackathon was on API Integration and Data Migration. The task was to integrate external APIs into the marketplace project.

I implemented a solution that allows us to migrate and import data from an external API into Sanity CMS.

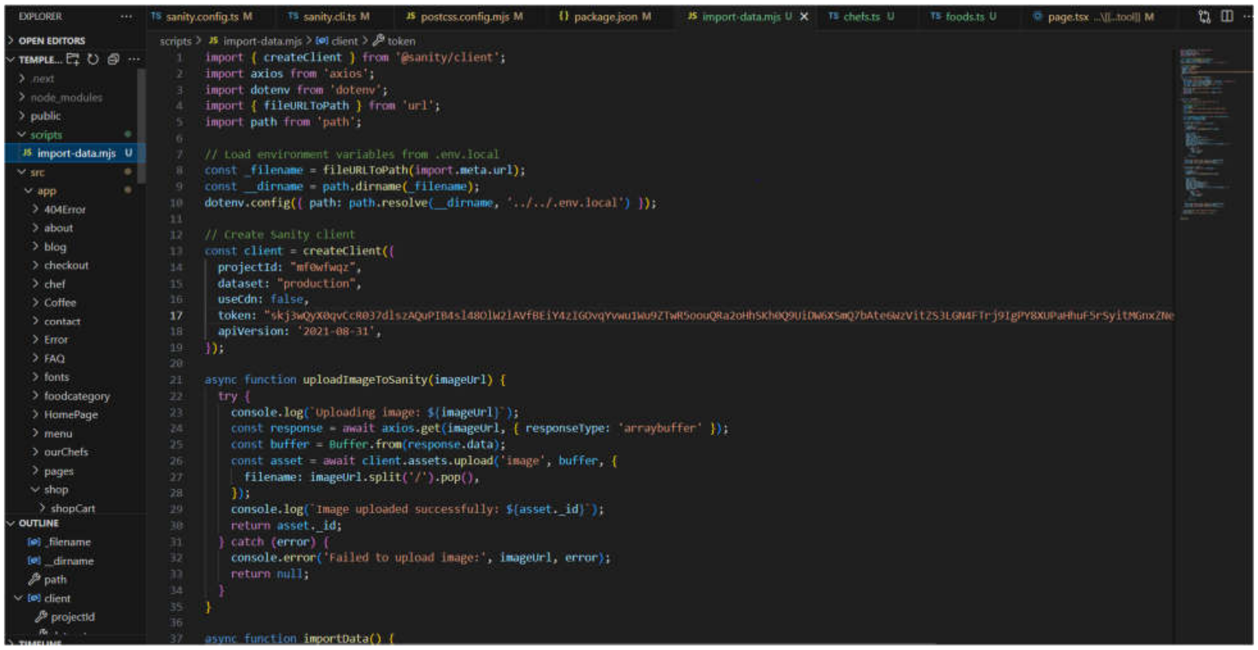
By the end of the day, the objective was to have a fully integrated marketplace with dynamic data fetched from APIs and correctly displayed on the frontend.

Below is a step-by-step explanation of how I achieved it:

2.Fetching Data From External API

The first step in this process was to fetch data from the external API.In my Next.js application.where the frontend code was already present,i created a folder named “scripts” and within that .I created a file called “data-migration.mjs” where I wrote the code to fetch data from the API.Afterward,I tested it both in the application and through Postman API, and my data was successfully fetched.

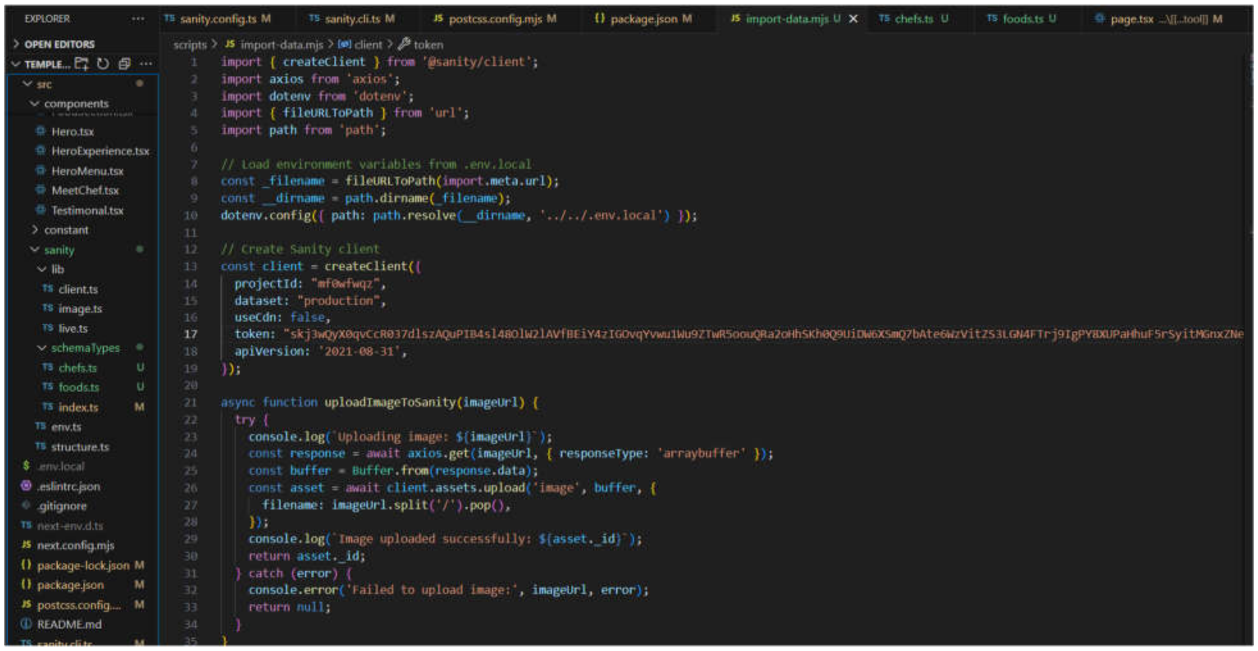
scripts/data-migration.mjs



```javascript
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: "mf0wfwqz",
  dataset: "production",
  useCdn: false,
  token: "skj3wQyX0qvCcR037dlszAQuPIB4sl48OlW2lAVfBEiY4zIGOvqYvwu1Wu9ZTwR5oouQRa2oHhSKh0Q9UiDW6XSmQ7bAte6WzVitZS3LGN4FTrj9IgPY8XUPaHhuF5rSyitMGnxZNe
  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() {
```

2.Comparing API data with Sanity Schema

After the data was fetched,the next task was to compare the data structure of the API with the Sanity schema.The Sanity schema will be used to handle the product data.The main purpose of this step was to ensure that the data coming from the API matches the Sanity schema.This Step ensures that the data from the API perfectly matches the Sanity schema .

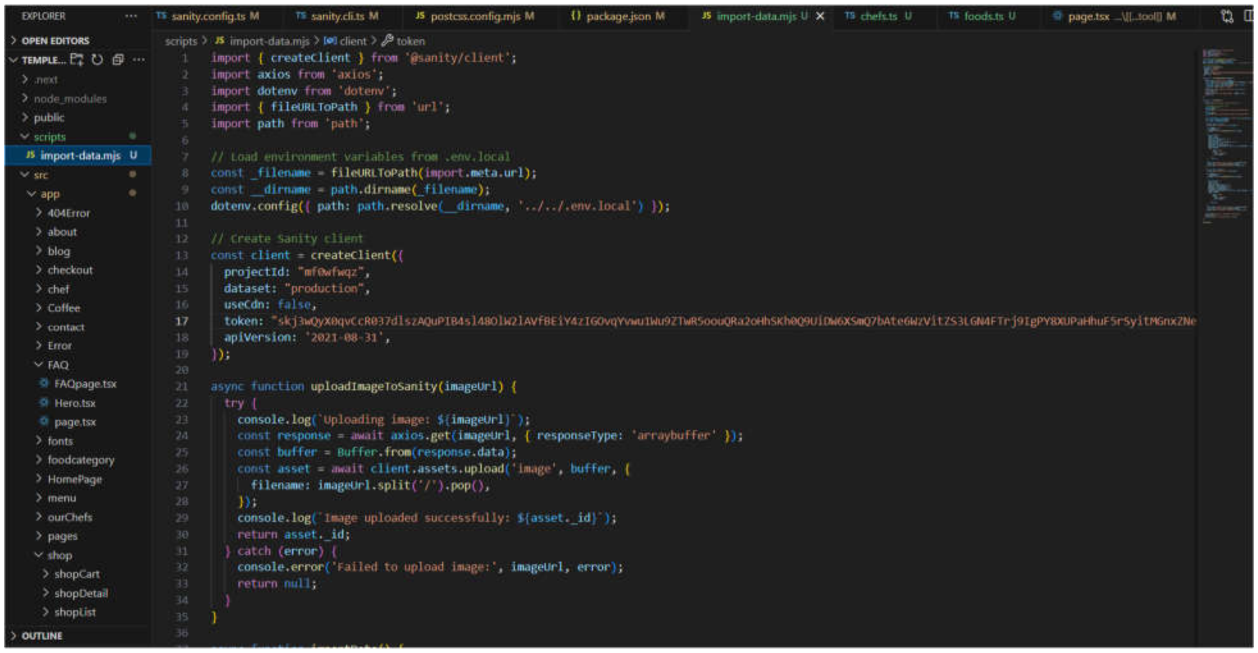


```javascript
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// 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: "mf0wfwqz",
  dataset: "production",
  useCdn: false,
  token: "skj3wQyX0qvCcR037dlszAQuPIB4sl48OlW2lAVfBEiY4zIGOvqYvwu1Wu9ZTwR5oouQRa2oHhSKh0Q9UiDW6XSmQ7bAte6WzVitZS3LGN4FTrj9IgPY8XUPaHhuF5rSyitMGnxZNe
  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;
  }
}
```

3. Data Migration Script

After coming the Schema the schema And the API data structure ,the next step was to write a migration script in existing file data-migration.mjs that wood automate the process of importing the fetch data into Sanity.A script was created that would:

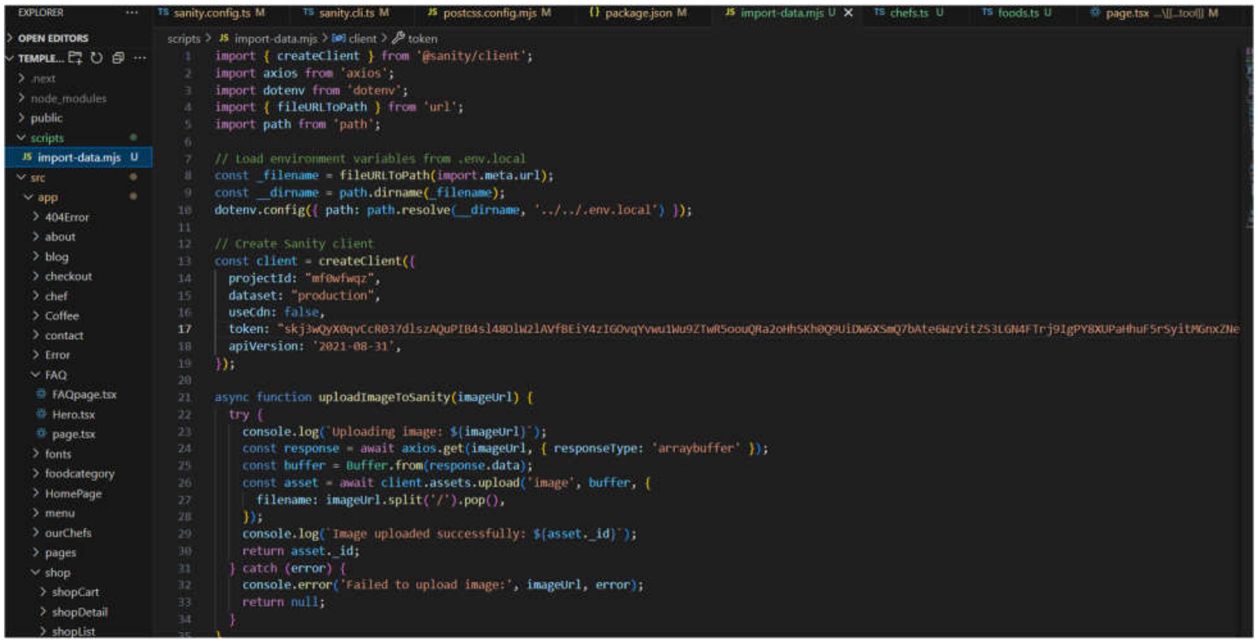
1.Fetch data from API
2.Validate data with sanity schema
3.Use the sanity client to push data into sanity



```javascript
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: "mf0wfwqz",
  dataset: "production",
  useCdn: false,
  token: "skj3wQyX0qvCcR037dlszAQuPIB4sl48OlW2lAVfBEiY4zIGOvqYvwu1Wu9ZTwR5oouQRa2oHhSKh0Q9UiDW6XSmQ7bAte6WzVitZS3LGN4FTrj9IgPY8XUPaHhuF5rSyitMGnxZNe
  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;
  }
}
```

4.Setting Up The Environment

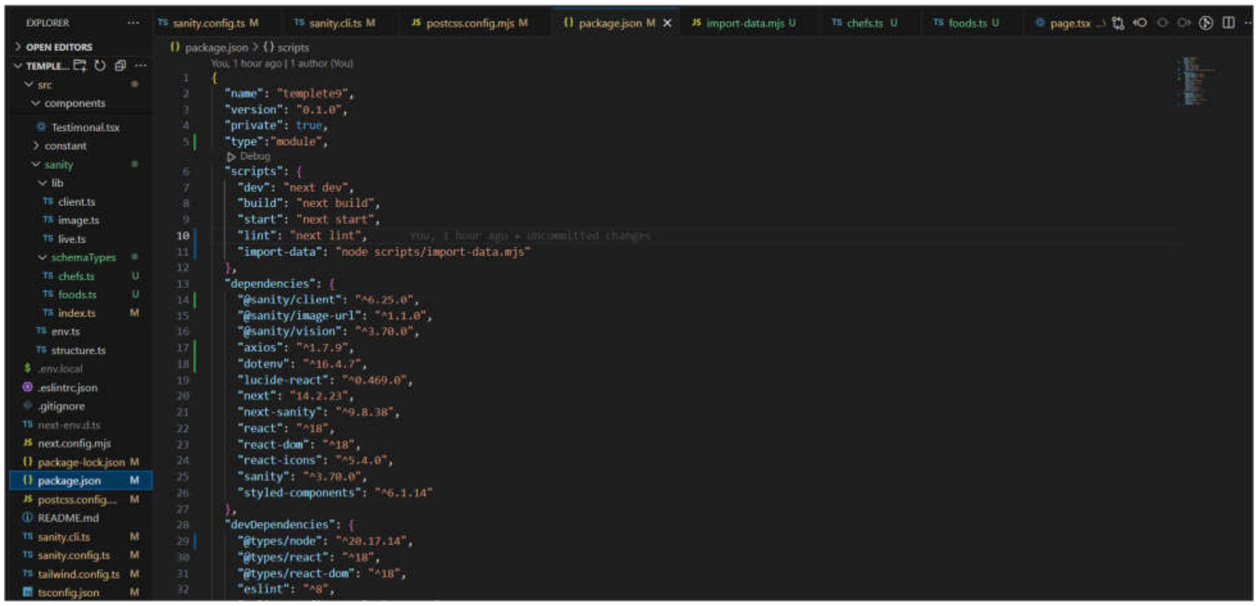
Once the script was written ,the next step was to configure Sanity .For the configuration, a token for the project needs to be generated from Sanity’s official website ,and by looking at the ID,DATA, and TOKEN,it needs to be written in the env.local file,and also provided in the data-migration.mjs file.



```
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: "mf0wfwqz",
  dataset: "production",
  useCdn: false,
  token: "skj3wQyX0qvCcR037dlszAQuPIB4sl48OlW2lAVfBEiY4zIGOvqYvwu1Wu9ZTwR5oouQRa2oHh5Kh0Q9UiDW6XSmQ7bAte6WzVitZS3LGN4FTrj9IgPY8XUPaHhuF5rSyitMGnxZNe
  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;
  }
}
```

5.modifying Package.json

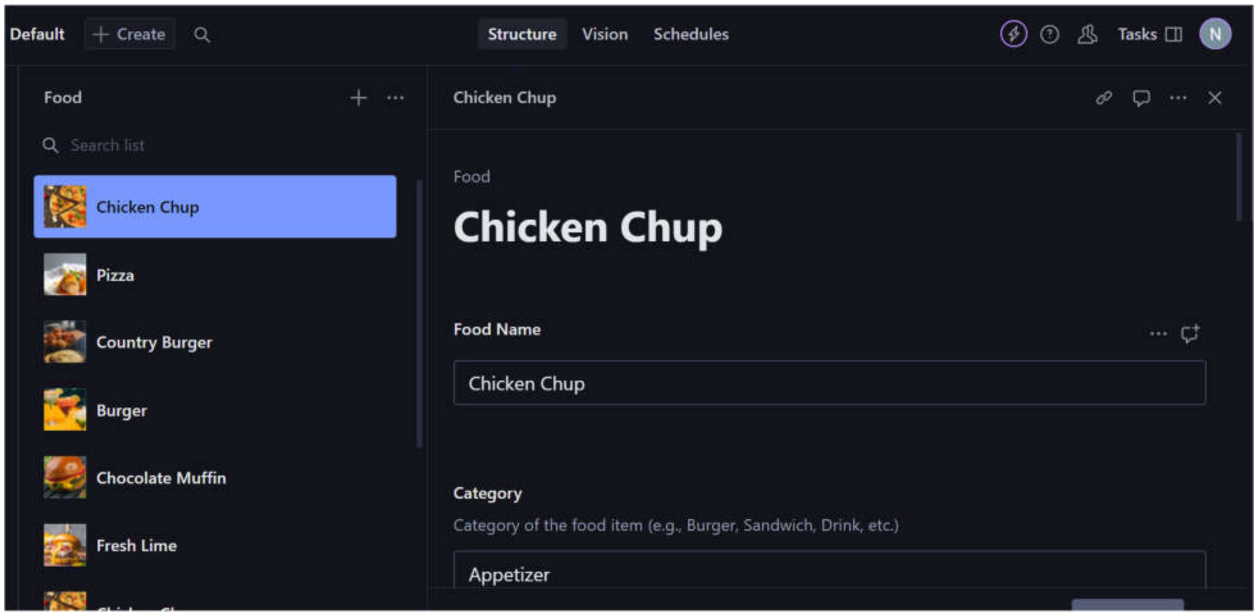
To the migration script,the necessary npm package was edit to the package.json file under the scripts section.This made it easier to run the script through the command line.



```
{
  "name": "templete9",
  "version": "0.1.0",
  "private": true,
  "type":"module",
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint",
    "import-data": "node scripts/import-data.mjs"
  },
  "dependencies": {
    "@sanity/client": "^6.25.0",
    "@sanity/image-url": "^1.1.0",
    "@sanity/vision": "^3.70.0",
    "axios": "^1.7.9",
    "dotenv": "^16.4.7",
    "lucide-react": "^0.469.0",
    "next": "14.2.23",
    "next-sanity": "^9.8.38",
    "react": "^18",
    "react-dom": "^18",
    "react-icons": "^5.4.0",
    "sanity": "^3.70.0",
    "styled-components": "^6.1.14"
  },
  "devDependencies": {
    "@types/node": "^20.17.14",
    "@types/react": "^18",
    "@types/react-dom": "^18",
    "eslint": "^8",
```

6. Importing Data into Sanity

With everything set up,I executed the migration script to send data into sanity by running the following command



Food

Chicken Chup

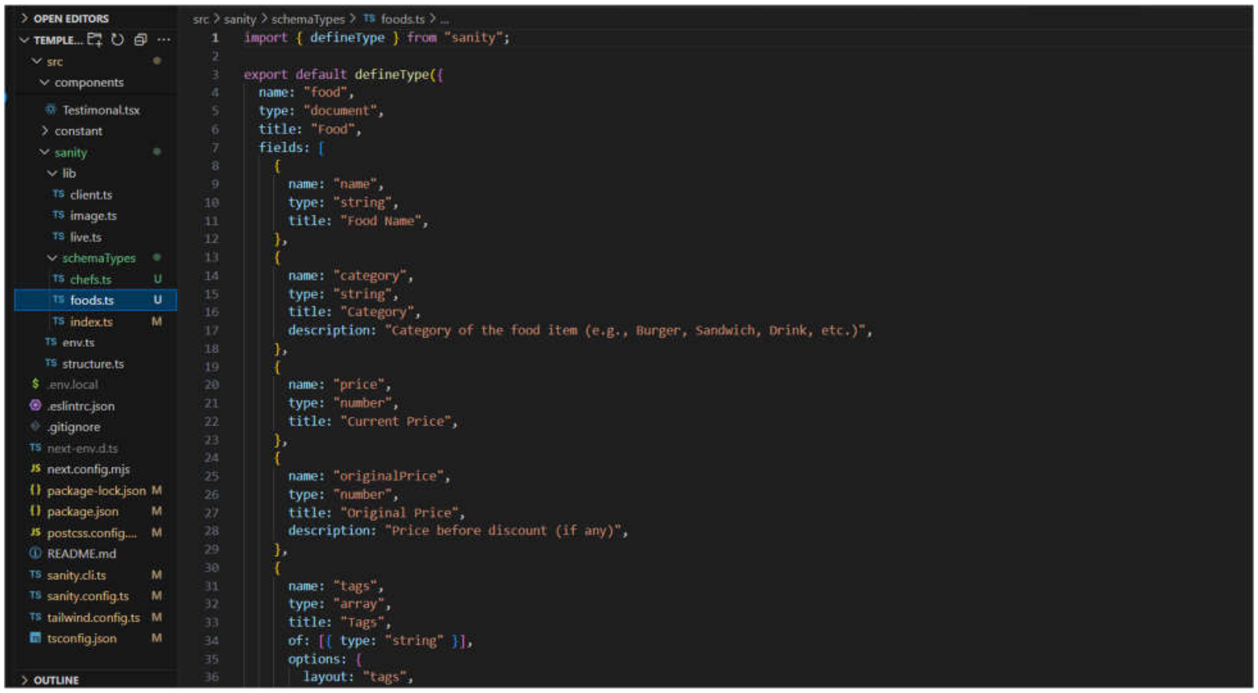
Food Name

Chicken Chup

Category

Category of the food item (e.g., Burger, Sandwich, Drink, etc.)

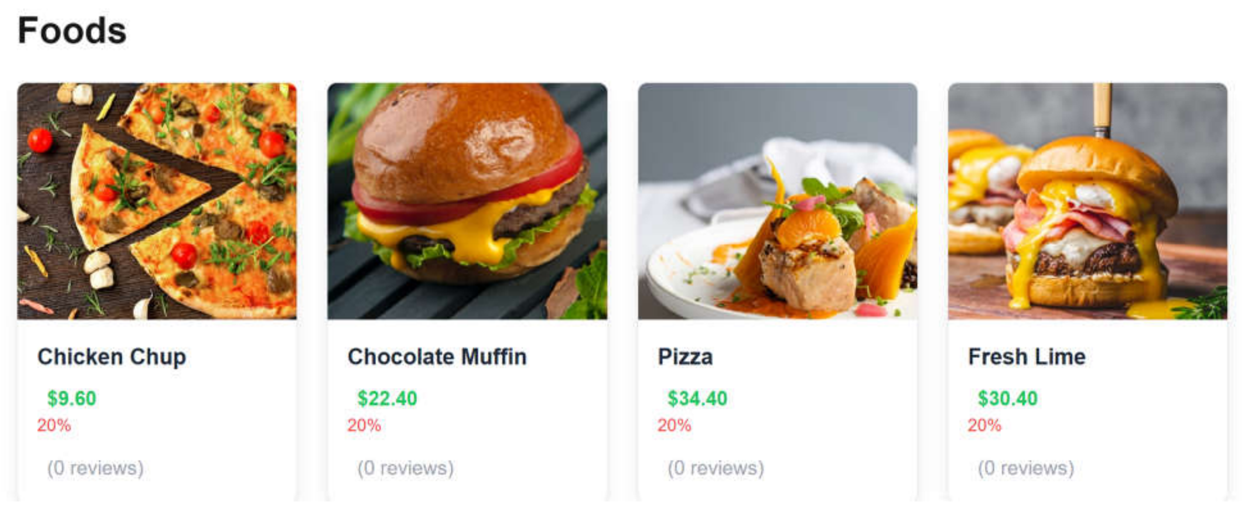
Appetizer



```
import { defineType } from "sanity";

export default defineType({
  name: "food",
  type: "document",
  title: "Food",
  fields: [
    {
      name: "name",
      type: "string",
      title: "Food Name",
    },
    {
      name: "category",
      type: "string",
      title: "Category",
      description: "Category of the food item (e.g., Burger, Sandwich, Drink, etc.)",
    },
    {
      name: "price",
      type: "number",
      title: "Current Price",
    },
    {
      name: "originalPrice",
      type: "number",
      title: "Original Price",
      description: "Price before discount (if any)",
    },
    {
      name: "tags",
      type: "array",
      title: "Tags",
      of: [{ type: "string" }],
      options: {
        layout: "tags",
```

7. Displaying data in frontend

After fetching data from Sanity.i mapped it. And printed it in cards ,and successfully my data was displayed on the frontend .

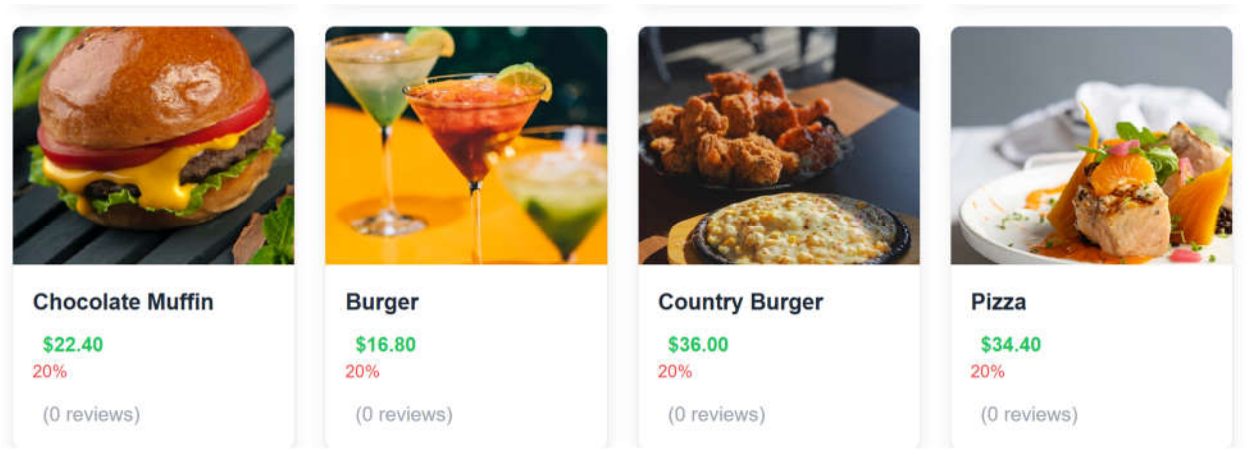

Foods

Chicken Chup
$9.60
20%
(0 reviews)

Chocolate Muffin
$22.40
20%
(0 reviews)

Pizza
$34.40
20%
(0 reviews)

Fresh Lime
$30.40
20%
(0 reviews)


Chocolate Muffin
$22.40
20%
(0 reviews)

Burger
$16.80
20%
(0 reviews)

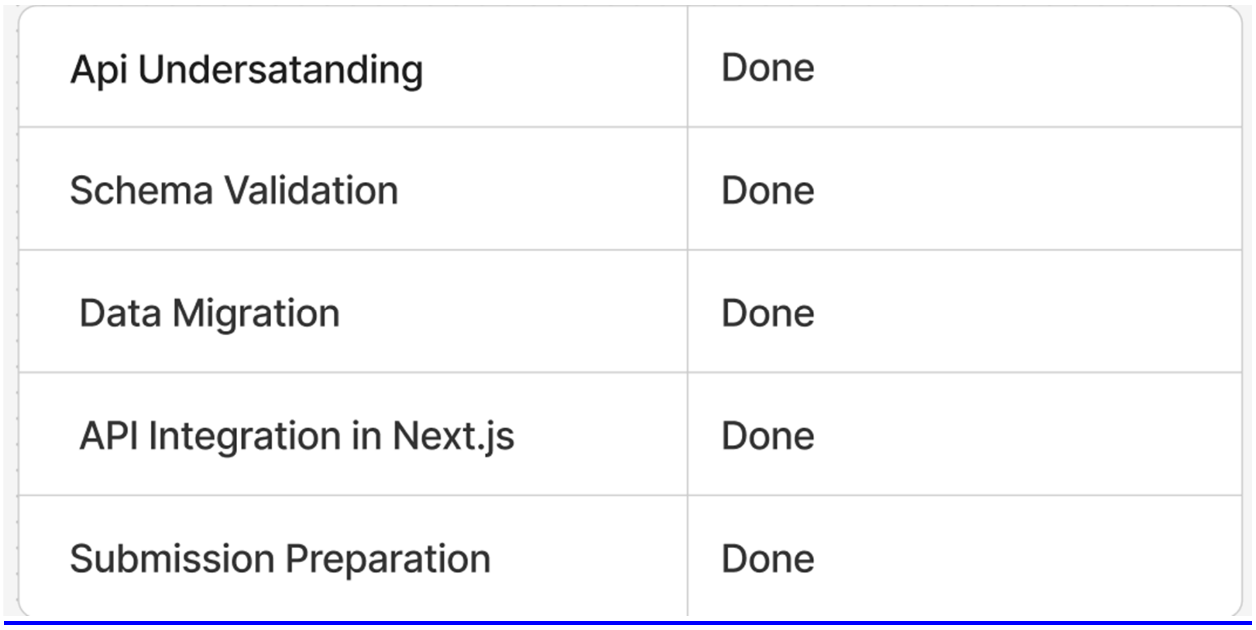
Country Burger
$36.00
20%
(0 reviews)

Pizza
$34.40
20%
(0 reviews)

● Conclusion

This Process Successfully Automarted the Migration of data from external API to Sanity CMS.The Key steps included:

1.Fetching Data from External APIs
2.Comparing API data with Sanity schema
3.Data Migration Script
4.Setting Up The Environment
5.Modifying Package.json Importing Data into Sanity
7.Displaying data in frontend



Api Undersatanding	Done
Schema Validation	Done
Data Migration	Done
API Integration in Next.js	Done
Submission Preparation	Done

Presented By: Nazia Siraj