Marketplace API Integration Report— Restaurant Website

API Integration Process:

This document outlines the steps I followed to integrate the Sanity API into my Next.js restaurant website. The process involved migrating data from a provided API using a migration script, fetching food data, and displaying it dynamically on the shop page within the existing design.

1- Migration of Data

I received a migration script from my instructor to transfer data from their Sanity API to the Sanity CMS used for my project. I executed the script locally, ensuring that all data was successfully migrated. Afterward, I verified the migration by checking the `food` and chefs documents in the Sanity Studio to confirm that all records were imported correctly.

2- Integration of Sanity API

To establish a connection with the Sanity API,I initialized sanity cms in my Next.js application and configured it with the project credentials, including the project ID and dataset name.This setup ensured a connection between my Next.js application and the Sanity backend.

3- Fetching Data

I created a query to retrieve food and chefs data from the Sanity CMS. The query targeted specific fields, including the name, category, price, original price, tags, image, description, and availability status.

Using this query, I implemented a utility function to fetch the data and integrated it into the website. This step allowed me to retrieve all the food items stored in the CMS for use in the application.

4- Displaying Data on the Specific Page

I updated the shop page to fetch data dynamically. This was done by utilizing the utility function to retrieve the data and passing it to the shop page. I then ensured that the food items were displayed in the existing design structure.

Adjustments Made To Schemas:

Creation of a New Schema for Chefs:

As part of the project requirements, I created a new schema specifically for chefs. The `chef` schema is designed to capture the necessary information about the chefs working in the restaurant. This schema includes the following fields:

- **`name`**: The name of the chef.
- **`position`**: The chef's role or title within the restaurant (e.g., Head Chef, Sous Chef).
- **`experience`**: The number of years of experience the chef has in the culinary field.
- **`specialty`**: A description of the chef's culinary specialties.
- **`image`**: An image of the chef to add a personal touch to the profile.
- **`description`**: A detailed description of the chef, including background, achievements, and notable contributions to the restaurant.
- **`available`**: A boolean field indicating whether the chef is available for a specific role or event.

Marketplace API Integration Report— Restaurant Website

Adjustments to the Food Schema:

The Sanity API provided did not include a `slug` field in the `food` schema. After migrating the data from their API to my website's Sanity CMS, I noticed the need for slugs to be automatically generated for each food item. To address this, I added a `slug` field to the `food` schema. The `slug` is now auto-generated based on the food's name, making it easier to create user-friendly and SEO-friendly URLs for each food item. This adjustment ensures consistency and improves the structure of the website's URLs.

Migration Steps And Tools Used:

1. **Access the Provided Sanity API**
   The migration process began by accessing the Sanity API provided by my sir. I used the provided script to fetch data from the API endpoints that contain the food and chef information. The data was fetched in JSON format.
2. **Set Up Sanity Client for the Target CMS**
   I configured the Sanity client to connect to my website's Sanity CMS. This required specifying the project ID, dataset, and API token, which I securely stored in environment variables using the .env package. With the client successfully configured, I was able to interact with my Sanity project and upload documents.
3. **Data Transformation and Mapping**
   After fetching the data, I transformed it to match the structure required by my Sanity CMS. This involved the following steps:
   - **Mapping Fields**: Ensuring that the data from the provided API was correctly mapped to the fields in my Sanity schema.
   - **Slug Generation**: Since the original API did not include slugs, I added a slug field to my schema and implemented

an auto-generation mechanism based on the food name. This created SEO-friendly URLs for each food item.

Tools Used

1. **Sanity API**: The Sanity API provided by my sir was used as the source of the data. It served as the endpoint from which I fetched the food and chef data.
2. **Sanity Client**: The Sanity client was used to interact with my website's Sanity CMS. I utilized it to upload data to my Sanity project, create documents, and manage the content.
3. **.env**: The .env package was used to manage environment variables securely. It allowed me to store sensitive information like API tokens and project IDs, keeping them safe and easily configurable.