HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]

API Integration Process

Overview

This integration focuses on establishing a connection between an external API that supplies data for foods and chefs and a **Sanity CMS** project. The goal is to enable smooth data migration and efficient management, ensuring seamless synchronization between the external source and the CMS.

API EndPoints:

Foods : https://sanity-nextjs-rouge.vercel.app/api/foods

Chefs : https://sanity-nextjs-rouge.vercel.app/api/chefs

This API returns a list of food and chefs, each containing detail such as

- Id
- Name
- Position
- Experience
- Specialty
- Image URL
- Description
- Available

We use Axios to fetch the data and then set it in the **React state** for rendering

[BY LAIQA EMAN]

HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]

Schema Adjustments

For the product data to be correctly migrated and stored in Sanity CMS, we adjusted the Sanity schema to match the food and chefs fields coming from API.

Sanity Schema:

Foods

```
export default {
    name: 'chef',
    type: 'document',
    title: 'Chef',
    fields: [
        name: 'name',
        type: 'string',
        title: 'Chef Name',
      },
      ],
Experience',
      {
        name: 'image',
        type: 'image',
        title: 'Chef Image',
        options: {
          hotspot: true,
        },
      },
      {
        name: 'description',
```

HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]

```
      type: 'text',
      title: 'Description',
      description: 'Short bio or introduction about the chef',
    },
  ],
};
```

We have ensured that the necessary elds are available in the schema for importing data like food, chefs, description, images, etc.

Migration Steps and Tools Used

Data Integration:

We used **Axios** to fetch data from an external API for food and chef information. This allowed us to retrieve product data and display it on the frontend.

Image Upload:

We uploaded food and chef images to Sanity by treating them as buffers and using the **Sanity client** to store them in the CMS, ensuring seamless image management.

Data Migration:

After fetching the food and chef data, we iterated over it and used client. create() to create new documents in Sanity, populating the CMS with the necessary product and chef details.

Sanity Client Setup:

HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]

To interact with Sanity, we set up the **Sanity Client** (@sanity/client) by importing it and configuring it with project credentials in the .env.local file.

Migration Script:

- **Setting Up Sanity Client**:
- import { createClient } from '@sanity/client';
- **Project ID & Dataset**: We used the unique **project ID** and **dataset** to set up the Sanity client.
- **CDN Configuration**: Set useCDN to false during development for real-time data fetching.

Tools and Technologies Used:

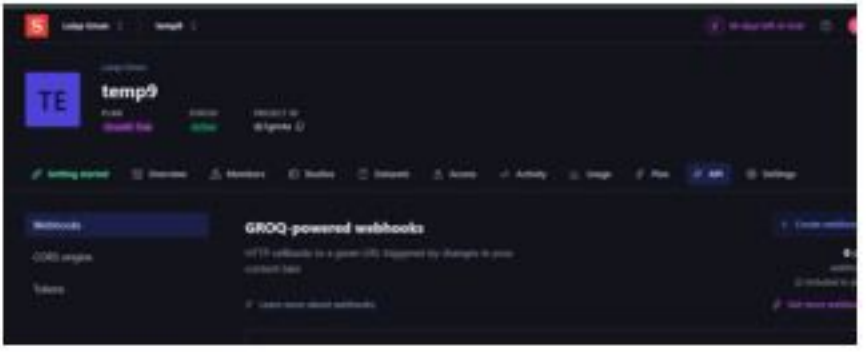
- **Sanity CMS**: Used for managing and storing product data.
- **Axios**: For fetching data from the external API.
- **Sanity Client**: For interacting with Sanity to upload images and create documents.
- **Next.js**: React-based framework for building the frontend.
- **.env**: For managing environment variables securely.

[BY LAIQA EMAN]

HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]

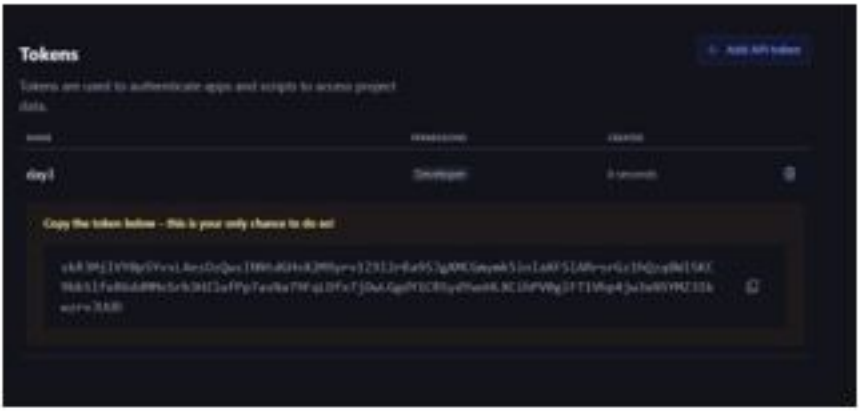
Visual Documentation:

Steps Tokens



[BY LAIQA EMAN]

HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]



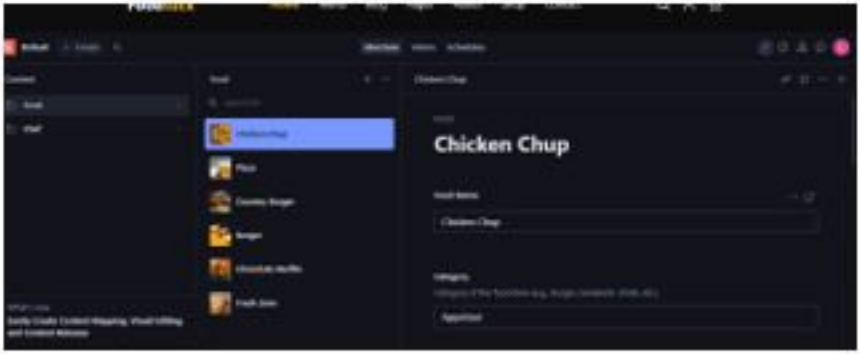
API Calls

[BY LAIQA EMAN]

HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]

Populated Sanity CMS Fiels

Foods



Chefs

[BY LAIQA EMAN]

HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]

Code Snippets for Api Integration and Migration Scripts

Chefs

[BY LAIQA EMAN]

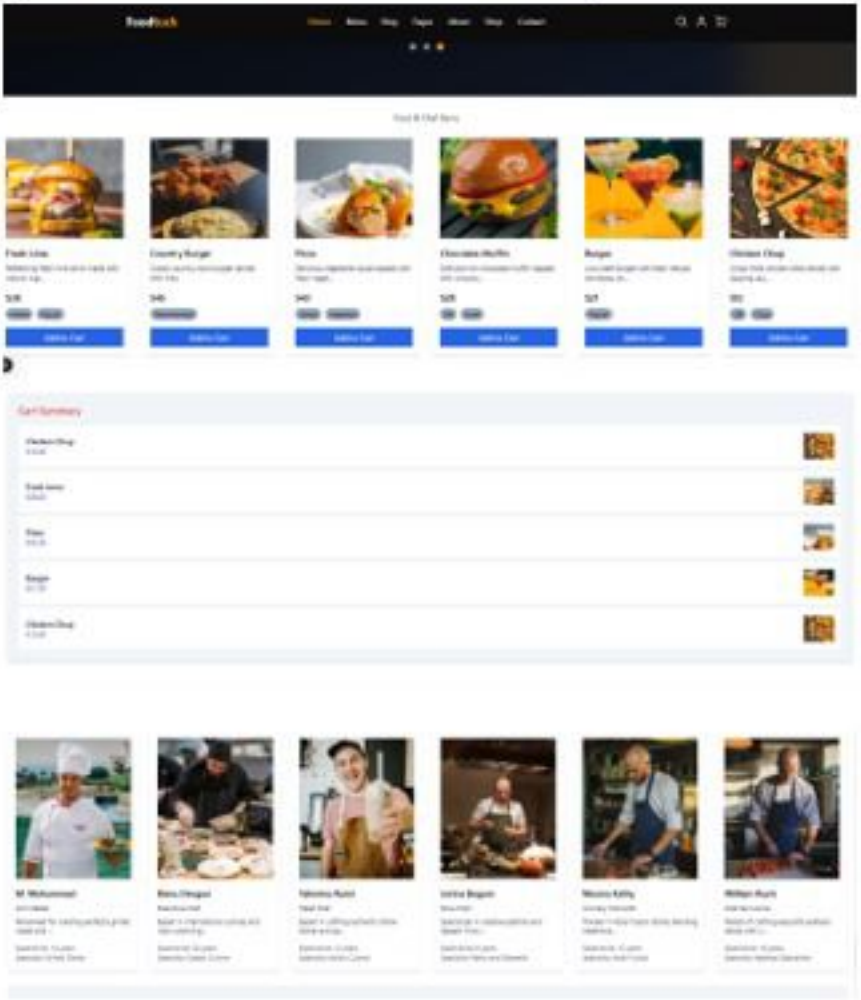
HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]

Foods

[BY LAIQA EMAN]

HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]

Data successfully displayed in the frontend.



[BY LAIQA EMAN]

HACKATHON 3
Day 3 - API Integration Report - [Q-commerce]

Final Check

- ☐ API Understanding: ✓
- ☐ Schema Validation: ✓
- ☐ Data Migration: ✓
- ☐ API Integration in Next.js: ✓
- ☐ Submission Preparation: ✓

[BY LAIQA EMAN]