

# 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

# 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 fields 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:

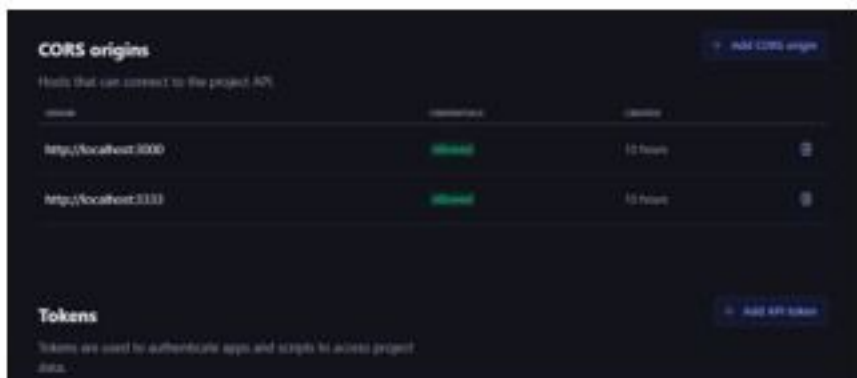
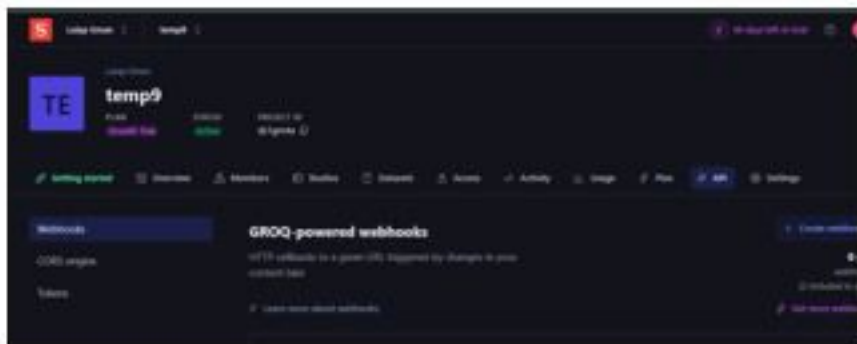
- **Sanity CMS:** Used for managing and storing product data.
- **Axios:** For fetching data from the external API.
- **Sanity Client:** For interacting with Sanity to upload images and create documents.
- **Next.js:** React-based framework for building the frontend.
- **.env:** For managing environment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

# HACKATHON 3

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

### Visual Documentation:

#### Steps Tokens





# HACKATHON 3

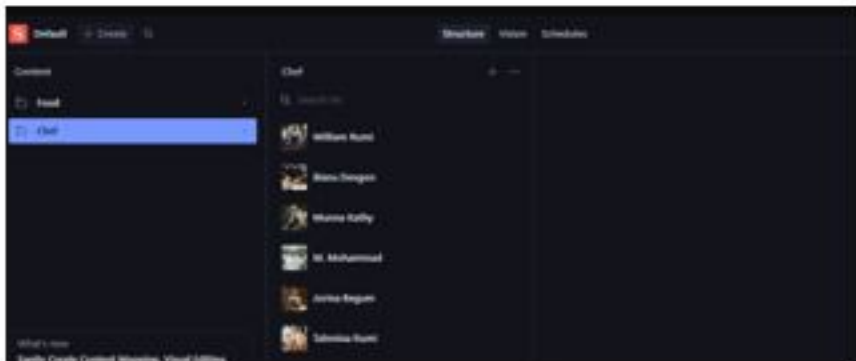
## Day 3 - API Integration Report - [Q-commerce]

### Populated Sanity CMS Fields

#### Foods



#### Chefs







# HACKATHON 3

## Day 3 - API Integration Report -

### [Q-commerce]

#### Foods

```
1 export default {
2   name: 'Food',
3   type: 'document',
4   title: 'Food',
5   fields: {
6
7     name: {
8       type: 'string',
9       title: 'Food Name',
10    },
11
12    category: {
13      type: 'string',
14      title: 'Category',
15      description:
16        'Category of the food item (e.g., Burger, Sandwich, Drink, etc.)',
17    },
18
19    price: {
20      type: 'number',
21      title: 'Current Price',
22    },
23
24    originalPrice: {
25      type: 'number',
26      title: 'Original Price',
27      description: 'Price before discount (if any)',
28    },
29
30    tags: {
31      type: 'array',
32      title: 'Tags',
33      of: { type: 'string' },
34      options: {
35        input: 'tags',
36      },
37      description: 'Tags for categorization (e.g., Best seller, Popular, New)',
38    },
39
40    image: {
41      type: 'image',
42      title: 'Food Image',
43      options: {
44        hotspot: true,
45      },
46    },
47
48    description: {
49      type: 'text',
50      title: 'Description',
51      description: 'Short description of the food item',
52    },
53
54    available: {
55      type: 'boolean',
56      title: 'Available',
57      description: 'Availability status of the food item',
58    },
59  },
60 }
```

# HACKATHON 3

## Day 3 - API Integration Report -

### [Q-commerce]

Data successfully displayed in the frontend.

The screenshot displays the Foodtuck web application interface. At the top, there is a navigation bar with the Foodtuck logo and links for Home, Menu, Map, Pages, About, Blog, and Contact. A search bar is located on the right side of the navigation bar. Below the navigation bar, there is a section titled "Food & Drink Items" which displays a grid of six food items, each with a photo, name, description, price, and a "View Item" button.

Below the food items section, there is a section titled "Get Summary" which displays a list of restaurants, each with a photo, name, address, and a "View Item" button.

At the bottom of the page, there is a section titled "Meet Our Team" which displays a grid of six team members, each with a photo, name, title, and a "View Item" button.

**Food & Drink Items:**

- French Fries:** Satisfying french fries with ketchup and salt. **5.0**
- Curry & Burger:** Juicy burger with curry sauce and fries. **5.0**
- Pizza:** Delicious pepperoni pizza with extra cheese. **5.0**
- Chocolate Muffin:** Soft and moist chocolate muffin with white chocolate chips. **5.0**
- Burger:** Juicy beef burger with cheese, tomato, and lettuce. **5.0**
- Chicken Wrap:** Crispy chicken wrap with cheese and tomato sauce. **5.0**

**Get Summary:**

- Chicken Wrap:** 5.0
- French Fries:** 5.0
- Pizza:** 5.0
- Burger:** 5.0
- Chocolate Muffin:** 5.0

**Meet Our Team:**

- Mr. Muhammad:** 5.0
- Mr. Ali:** 5.0
- Mr. Ali:** 5.0
- Mr. Ali:** 5.0
- Mr. Ali:** 5.0
- Mr. Ali:** 5.0

[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: ✓