Day 3 - API Integration Report - Q-Commerce Web

----------Food Tuck ----------

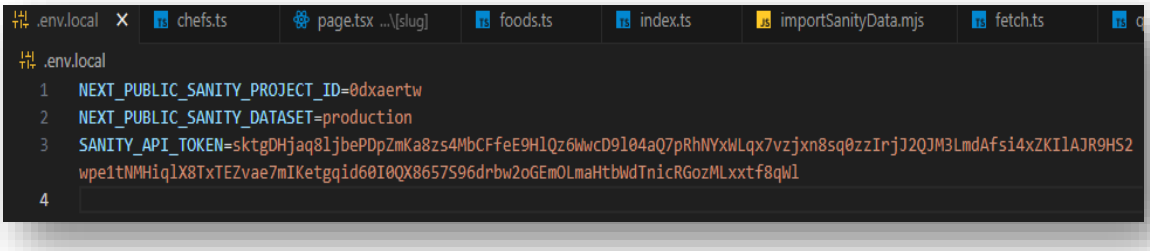
Prepared by: Kashaf Noor

API Integration Process

API Integration Process

The API integration for the Q-Commerce web application was completed in several steps:

1. **Understanding API Documentation:**
   - I thoroughly reviewed the API documentation of the third-party services used for food and chef data.
   - Focused on identifying the key endpoints required for integration, such as food items, chef details, and real-time updates.
2. **Setting Up Environment Variables:**
   - I set up environment variables in .env.local to securely store the necessary API keys and tokens for accessing the API:



3. **Endpoint Integration:**
   - I integrated various endpoints to fetch food item data and chef details.
   - Added error handling and response validation to ensure the integration is reliable and consistent.

Adjustments Made to Schemas

1. **Sanity Schema Updates:**
   - I updated the food schema to accommodate additional fields such as price, ingredients, and availability, ensuring it aligns with the API responses:

FOOD SCHEMA

```
export default {
    name: 'food',
    type: 'document',
    title: 'Food',
    fields: [
      {
        name: 'name',
        type: 'string',
        title: 'Food Name',
      },
      {
        name:'slug',
        type : 'slug',
        title : 'Slug',
        options:{
          source:'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',
```

```
        },
        description: 'Tags for categorization (e.g., Best Seller, Popular, New)',
      },
      {
        name: 'image',
        type: 'image',
        title: 'Food Image',
        options: {
          hotspot: true,
        },
      },
      {
        name: 'description',
        type: 'text',
        title: 'Description',
        description: 'Short description of the food item',
      },
      {
        name: 'available',
        type: 'boolean',
        title: 'Available',
        description: 'Availability status of the food item',
      },
    ],
  };

```

I also modified the chef schema, adding fields for expertise and experience to enrich the chef data:

CHEF SCHEMA

```
export default {
    name: 'chef',
    type: 'document',
    title: 'Chef',
    fields: [
      {
        name: 'name',
        type: 'string',
        title: 'Chef Name',
      },
      {
        name: 'position',
        type: 'string',
        title: 'Position',
        description: 'Role or title of the chef (e.g., Head Chef, Sous Chef)',
      },
```

```
      {
        name: 'experience',
        type: 'number',
        title: 'Years of Experience',
        description: 'Number of years the chef has worked in the culinary
field',
      },
      {
        name: 'specialty',
        type: 'string',
        title: 'Specialty',
        description: 'Specialization of the chef (e.g., Italian Cuisine,
Pastry)',
      },
      {
        name: 'image',
        type: 'image',
        title: 'Chef Image',
        options: {
          hotspot: true,
        },
      },
      {
        name: 'description',
        type: 'text',
        title: 'Description',
        description: 'Short bio or introduction about the chef',
      },
      {
        name: 'available',
        type: 'boolean',
        title: 'Currently Active',
        description: 'Availability status of the chef',
      },
    ],
  };
```

Migration Steps and Tools Used

1. **Tools Utilized:**
    - I used @sanity/client to migrate the data from the API into the Sanity CMS.

2. **Migration Steps:**
    - I fetched food and chef data from the external API.
    - Transformed the API data to fit the updated Sanity schemas.

- Uploaded the transformed data to Sanity using the following migration script:

```
import { createClient } from 'next-sanity'

import { apiVersion, dataset, projectId } from '../env'

export const client = createClient({
  projectId,
  dataset,
  apiVersion,
  useCdn: true, // Set to false if statically generating pages, using ISR or tag-based revalidation
})

```

```
import { defineQuery, groq } from "next-sanity";

export const allchefs = defineQuery(`
        *[_type == "chef"]{
          _id,
         name,
         position,
         experience,
         specialty,
          image {
           asset -> {
             _id,
             url
           }
         },
         description,
         }`);


export const fourchefs = groq`*[_type == "chef"][0..3]`;


export const allfoods =groq`*[_type == "food"]`;



export const fourfoods =groq`*[_type == "food"][0..3]`
```

```
export interface Food {
    _id: string;            // Unique identifier for the food item
```

```
    name: string;           // Name of the food item
    _type : "food";
    category?: string;      // Category of the food item (optional)
    price: number;          // Current price of the food item
    originalPrice?: number; // Original price before discount (optional)
    tags?: string[];        // Tags for categorization (optional)
    image: { asset: { _ref: string; _type: "image"} };   // Image associated
with the food item (optional)
    description?: string;  // Description of the food item (optional)
    available: boolean;
    slug :{
        _type : "slug";
        current :string
    }
}
```

For chef

```
type Chef = {
    _id: string;            // Unique identifier for the chef document
    name: string;           // Chef's name
    position: string;       // Position or role of the chef
    experience: number;     // Years of experience the chef has
    specialty: string;      // Chef's area of expertise (e.g., Italian, Pastry)
    image: { asset: { _id: string; url: string } };          // Image
associated with the chef, with optional asset ID and URL
    description: string;   // A short bio or description of the chef

  };


  export default async function Page() {
     const chefs : Chef[] = await sanityFetch({query: allchefs})
```

for food

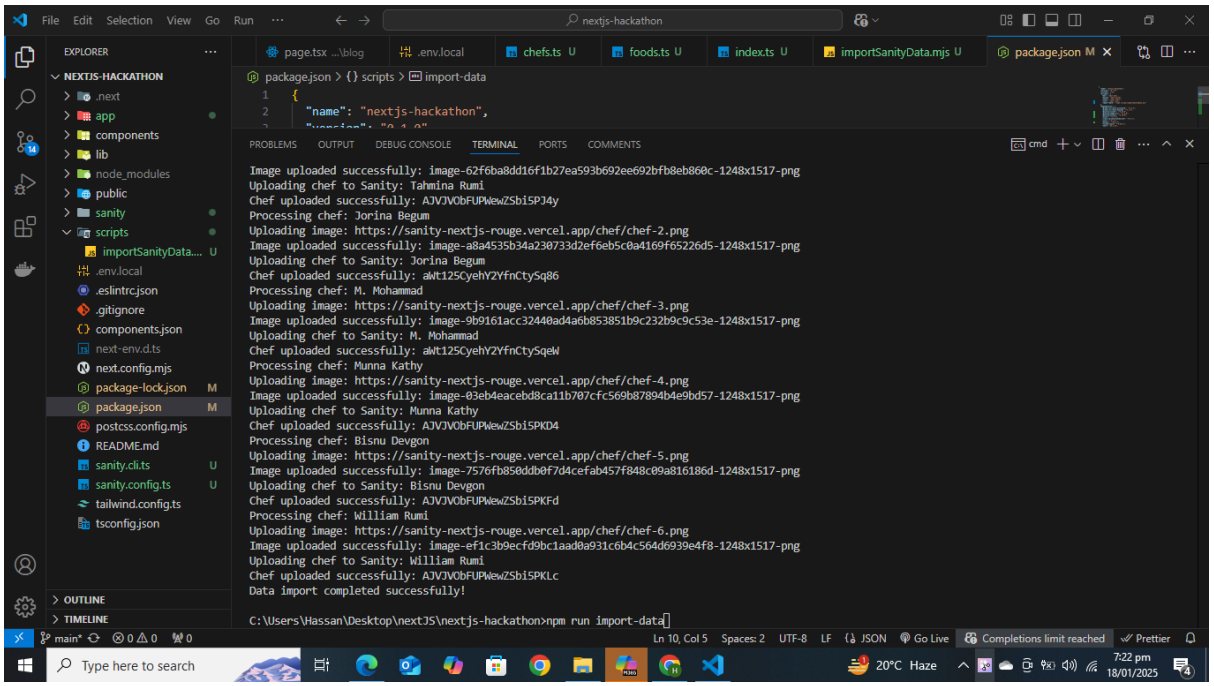
```
interface FoodPageProps{
  params: Promise<{slug:string}>
}
 async function getProduct(slug:string): Promise<Food> {
  return client.fetch(
    groq`*[_type == "food" && slug.current == $slug][0]{

    _id,
    name,
    category,
```

```
    price,
    originalPrice,
    tags,
    image {
      asset -> {
        _id,
        url
      }
    },
    description,
    available,
    }`,{slug}
  )
}


export default async function FoodPage({params}:FoodPageProps){
    const {slug} = await params
    const food =await getProduct(slug)
```

1. **API Calls:**



2. **Data Displayed on Frontend:**

CHEF

Tahmina Rumi

Position: Head Chef

Experience: 12 years

Specialty: Italian Cuisine

Description: Expert in crafting authentic Italian dishes and pastries.

Munna Kathy

Position: Culinary Instructor

Experience: 15 years

Specialty: Asian Fusion

Description: Pioneer in Asian fusion dishes blending traditional flavors with modern techniques.

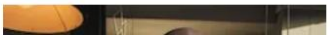
Bisnu Devgon

Position: Executive Chef

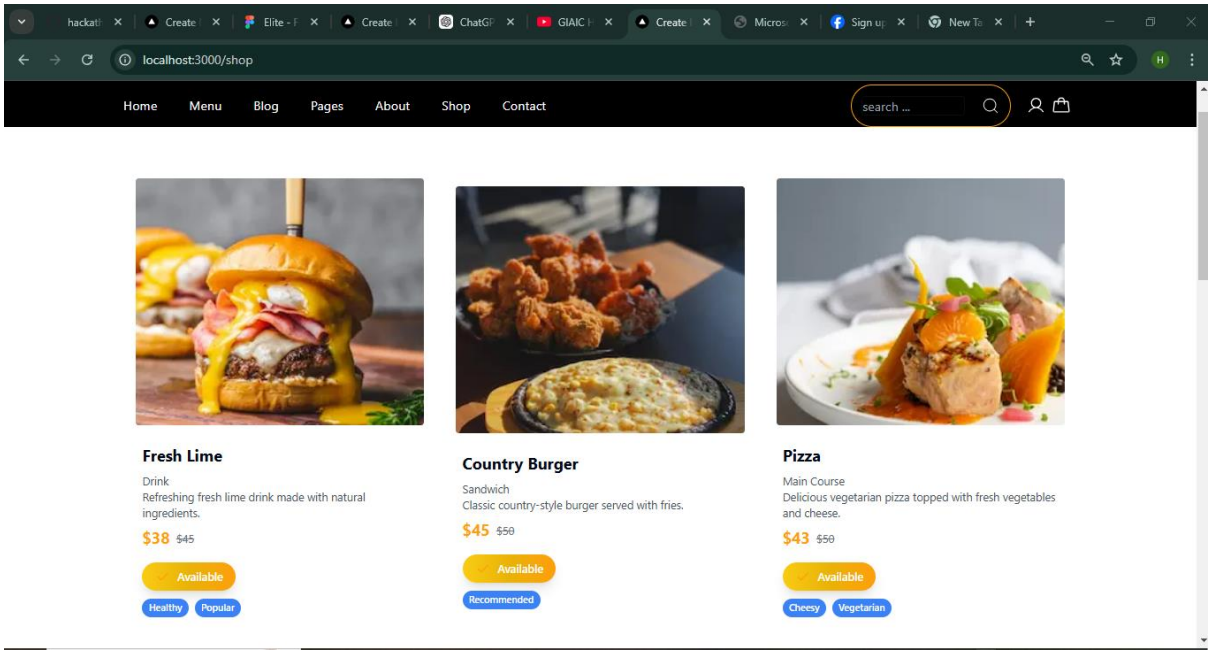
Experience: 20 years

Specialty: Global Cuisine

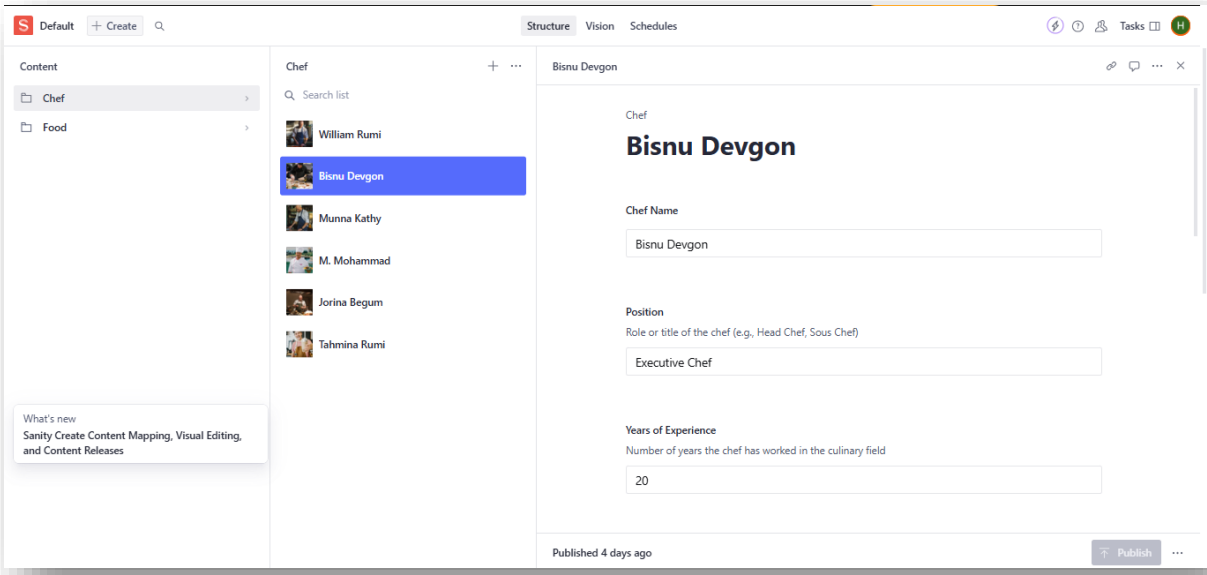
Description: Expert in international cuisines and menu planning.



FOOD



3. **Populated Sanity CMS Fields:**



This report outlines the steps I took to integrate APIs into the Q-commerce website, update schemas, and migrate the data into Sanity CMS.