

Day 3 - API Integration Report - [FOODTUCK]

Explanation of What I Have Done in Day 3 - Hackathon

1. Cloning the Repository

- I started by cloning the repository containing the Sanity project and import scripts.
- I used the following command to clone the repository:

```
git clone https://github.com/mubashirimtiaz/sanity-nextjs.git
cd sanity-nextjs
```

- This gave me access to the project files, including the Sanity schema and the import script.
-

2. Installing Dependencies

- After cloning the repository, I installed all the required packages by running:

```
npm install
```

- This installed all the dependencies listed in the package.json file, including Sanity client, Next.js, and other necessary libraries.
-

3. Configuring Environment Variables

- I created a .env.local file in the root of the project directory to store sensitive environment variables.
- I added the following variables to the .env.local file:

```
NEXT_PUBLIC_SANITY_PROJECT_ID="{your-sanity-project-id}"
NEXT_PUBLIC_SANITY_DATASET="production"
SANITY_API_TOKEN="{your-sanity-api-token}"
```

- **Explanation of Variables:**
 - NEXT_PUBLIC_SANITY_PROJECT_ID: Found in my Sanity project settings.
 - SANITY_API_TOKEN: Generated by navigating to **Settings > API > Add API Token** in the Sanity dashboard. I gave the token **read/write permissions**.
 - NEXT_PUBLIC_SANITY_DATASET: Set to production as the dataset name.
-

4. Importing Data

- I ran the following command to import sample data for the **Food** and **Chef** models:

```
npm run import-data
```

- This executed the import-data.mjs script, which:
 1. Fetched food and chef data from the external API (<https://sanity-nextjs-rouge.vercel.app/api/foods> and <https://sanity-nextjs-rouge.vercel.app/api/chefs>).

2. Uploaded images to Sanity and created references.
 3. Created documents in Sanity for each food and chef item.
-

5. Verifying the Data in Sanity Studio

- After running the import script, I opened my **Sanity Studio** project.
 - I verified that two models were created:
 1. **Food**: Contains fields like name, category, price, originalPrice, tags, image, description, and available.
 2. **Chef**: Contains fields like name, position, experience, specialty, image, description, and available.
 - I confirmed that the sample data was successfully populated in both models.
-

What I Have Done After This

After completing the data import, I worked on the following:

1. Updated Schemas

- I updated the **Food** and **Chef** schemas in Sanity to match the data structure from the API.
 - For example, I added fields like originalPrice and tags to the food schema and experience and specialty to the chef schema.
-

2. Built the Next.js Frontend

- I created two components in my Next.js application:
 1. **Food Component**:
 - Fetches food data from Sanity using a GROQ query.
 - Displays food items in a grid layout with sorting, filtering, and pagination features.
 2. **Chef Component**:
 - Fetches chef data from Sanity using a GROQ query.
 - Displays chefs in a grid layout with hover effects and availability status.
-

1. API Integration Process

1. Understanding the API

- The external API provided two endpoints:
 1. **Food Data:**
 - Endpoint: `https://sanity-nextjs-rouge.vercel.app/api/foods`
 - Returns a list of food items with fields like name, category, price, originalPrice, tags, image, description, and available.
 2. **Chef Data:**
 - Endpoint: `https://sanity-nextjs-rouge.vercel.app/api/chefs`
 - Returns a list of chefs with fields like name, position, experience, specialty, image, description, and available.
-

2. Fetching Data from the API

- I used **Axios** to fetch data from the API endpoints.
- Example for fetching **Food Data**:

```
const foodsResponse = await axios.get('https://sanity-nextjs-rouge.vercel.app/api/foods');
const foods = foodsResponse.data;
```

- Example for fetching **Chef Data**:

```
const chefsResponse = await axios.get('https://sanity-nextjs-rouge.vercel.app/api/chefs');
const chefs = chefsResponse.data;
```

3. Uploading Images to Sanity

- For each food and chef item, I checked if an image URL was provided.

```
async function uploadImageToSanity(imageUrl) {
  try {
    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(),
    });
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}
```

4. Creating Documents in Sanity

- After uploading images, I created documents in Sanity for each food and chef item.
- Example for **Food**:

```
for (const food of foods) {
```

```
const sanityFood = {
  _type: 'food',
  name: food.name,
  category: food.category || null,
  price: food.price,
  originalPrice: food.originalPrice || null,
  tags: food.tags || [],
  description: food.description || "",
  available: food.available !== undefined ? food.available : true,
  image: imageRef
  ? {
    _type: 'image',
    asset: {
      _type: 'reference',
      _ref: imageRef,
    },
  }
  : undefined,
};
await client.create(sanityFood);
}
```

- Example for **Chef**:

```
for (const chef of chefs) {
  const sanityChef = {
    _type: 'chef',
    name: chef.name,
    position: chef.position || null,
    experience: chef.experience || 0,
    specialty: chef.specialty || "",
    description: chef.description || "",
    available: chef.available !== undefined ? chef.available : true,
    image: imageRef
    ? {
      _type: 'image',
      asset: {
        _type: 'reference',
        _ref: imageRef,
      },
    }
    : undefined,
  };
  await client.create(sanityChef);
}
```

5. Running the Migration Script

- I ran the migration script using the following command:

```
npm run import-data
```

- This executed the import-data.mjs script, which:
 1. Fetched data from the API.

2. Uploaded images to Sanity.
 3. Created documents in Sanity for each food and chef item.
-

6. Verifying the Data in Sanity Studio

- After running the script, I opened **Sanity Studio** to verify that the data was successfully imported.
 - I checked the **Food** and **Chef** models to ensure that all fields were populated correctly.
-

Tools Used

1. Sanity Client

- The **Sanity client** (@sanity/client) was used to interact with the Sanity CMS.
- It allowed me to:
 - Upload images using client.assets.upload.
 - Create documents using client.create.

2. Axios

- **Axios** was used to fetch data from the external API.
- It handled HTTP requests and responses efficiently.

3. Node.js

- The migration script was executed using **Node.js**.
- Node.js provided the runtime environment for running the script.

4. Environment Variables

- I used a .env.local file to store sensitive environment variables like:
 - NEXT_PUBLIC_SANITY_PROJECT_ID
 - SANITY_API_TOKEN
 - NEXT_PUBLIC_SANITY_DATASET

5. Sanity Studio

- **Sanity Studio** was used to:
 - Define schemas for the **Food** and **Chef** models.
 - Verify that the imported data was correctly populated.
-

Summary of API Integration Process

1. **Fetches Data from the API:**
 - Used Axios to fetch food and chef data from the external API.
 2. **Uploaded Images to Sanity:**
 - Used the Sanity client to upload images and create references.
 3. **Created Documents in Sanity:**
 - Used the Sanity client to create documents for each food and chef item.
 4. **Ran the Migration Script:**
 - Executed the script using `npm run import-data`.
 5. **Verified the Data:**
 - Checked Sanity Studio to ensure the data was imported correctly.
 6. **Tools Used:**
 - Sanity Client, Axios, Node.js, Environment Variables, and Sanity Studio.
-

What Worked Well

- The migration script successfully fetched data from the API and uploaded it to Sanity CMS.
- The use of environment variables ensured that sensitive information was securely stored.
- Sanity Studio made it easy to verify that the data was correctly imported.

2. Adjustments Made to Schemas:

1. Food Schema

- I updated the **Food schema** in Sanity CMS to match the data structure from the external API.
- The updated schema includes the following fields:

```
const food = defineType({
  name: 'food',
  title: 'Food',
  type: 'document',
  fields: [
    defineField({
      name: 'name',
      title: 'Food Name',
      type: 'string',
    }),
    defineField({
      name: 'category',
      title: 'Category',
      type: 'string',
      description: 'Category of the food item (e.g., Burger, Sandwich, Drink, etc.)',
    }),
    defineField({
      name: 'price',
      title: 'Current Price',
      type: 'number',
    })
  ]
})
```

```

    }},
    defineField({
      name: 'originalPrice',
      title: 'Original Price',
      type: 'number',
      description: 'Price before discount (if any)',
    }),
    defineField({
      name: 'tags',
      title: 'Tags',
      type: 'array',
      of: [{ type: 'string' }],
      options: {
        layout: 'tags',
      },
      description: 'Tags for categorization (e.g., Best Seller, Popular, New)',
    }),
    defineField({
      name: 'image',
      title: 'Food Image',
      type: 'image',
      options: {
        hotspot: true,
      },
    }),
    defineField({
      name: 'description',
      title: 'Description',
      type: 'text',
      description: 'Short description of the food item',
    }),
    defineField({
      name: 'available',
      title: 'Available',
      type: 'boolean',
      description: 'Availability status of the food item',
    }),
  ],
});

```

Key Adjustments:

- Added `originalPrice` to store the original price of the food item (before discounts).
- Added `tags` as an array of strings to categorize food items (e.g., Best Seller, Popular, New).
- Added `description` to provide a short description of the food item.
- Added `available` as a boolean field to indicate the availability status of the food item.

2. Chef Schema

- I updated the **Chef schema** in Sanity CMS to match the data structure from the external API.
- The updated schema includes the following fields:

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


Key Adjustments:

- Added experience as a number field to store the number of years the chef has worked.
 - Added specialty as a string field to store the chef's specialization (e.g., Italian Cuisine, Pastry).
 - Added description to provide a short bio or introduction about the chef.
 - Added available as a boolean field to indicate the availability status of the chef.
-

Why These Adjustments Were Made

- **Original Price:** To display discounted prices and show the original price for comparison.
 - **Tags:** To categorize food items and improve searchability.
 - **Experience and Specialty:** To provide more details about the chefs and their expertise.
 - **Description:** To give users more information about food items and chefs.
 - **Availability:** To indicate whether a food item or chef is currently available.
-

How the Adjustments Were Implemented

1. Updated Schemas in Sanity Studio:

- I modified the food and chef schemas in Sanity Studio to include the new fields.

2. Updated the Migration Script:

- I ensured that the migration script (import-data.mjs) mapped the API data to the updated schema fields.
- Example for **Food**:

```
const sanityFood = {
  _type: 'food',
  name: food.name,
  category: food.category || null,
  price: food.price,
  originalPrice: food.originalPrice || null,
  tags: food.tags || [],
  description: food.description || '',
  available: food.available !== undefined ? food.available : true,
  image: imageRef
  ? {
    _type: 'image',
    asset: {
      _type: 'reference',
      _ref: imageRef,
    },
  }
  : undefined,
};
```

- Example for **Chef**:

```
const sanityChef = {
  _type: 'chef',
  name: chef.name,
```

```

position: chef.position || null,
experience: chef.experience || 0,
specialty: chef.specialty || "",
description: chef.description || "",
available: chef.available !== undefined ? chef.available : true,
image: imageRef
? {
  _type: 'image',
  asset: {
    _type: 'reference',
    _ref: imageRef,
  },
}
: undefined,
};

```

3. Tested the Schemas:

- I ran the migration script and verified that the data was correctly populated in Sanity Studio.
- I checked that all fields were mapped correctly and displayed as expected.

Summary of Schema Adjustments

- Food Schema:**
 - Added originalPrice, tags, description, and available fields.
 - Updated the schema to match the API data structure.
- Chef Schema:**
 - Added experience, specialty, description, and available fields.
 - Updated the schema to match the API data structure.
- Migration Script:**
 - Updated the script to map API data to the new schema fields.
- Testing:**
 - Verified that the data was correctly populated in Sanity Studio.

What Worked Well

- The schema adjustments ensured that the data from the API was correctly mapped and stored in Sanity CMS.
- The new fields provided more detailed information about food items and chefs, improving the user experience.

3. Migration Steps:

1. Setting Up the Migration Script

- I used the `import-data.mjs` script provided in the repository to migrate data from the external API to Sanity CMS.

- The script is written in JavaScript and uses the **Sanity client** and **Axios** libraries to fetch and upload data.
-

2. Fetching Data from the API

- The script fetches data from two API endpoints:

1. **Food Data:**

```
const foodsResponse = await axios.get('https://sanity-nextjs-rouge.vercel.app/api/foods');
const foods = foodsResponse.data;
```

2. **Chef Data:**

```
const chefsResponse = await axios.get('https://sanity-nextjs-rouge.vercel.app/api/chefs');
const chefs = chefsResponse.data;
```

3. Uploading Images to Sanity

- For each food and chef item, the script checks if an image URL is provided.
- If an image exists, it uploads the image to Sanity using the `client.assets.upload` method:

```
async function uploadImageToSanity(imageUrl) {
  try {
    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(),
    });
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}
```

4. Creating Documents in Sanity

- After uploading images, the script creates documents in Sanity for each food and chef item.
- Example for **Food**:

```
for (const food of foods) {
  const sanityFood = {
    _type: 'food',
    name: food.name,
    category: food.category || null,
    price: food.price,
    originalPrice: food.originalPrice || null,
    tags: food.tags || [],
    description: food.description || '',
    available: food.available !== undefined ? food.available : true,
```

```

image: imageRef
? {
  _type: 'image',
  asset: {
    _type: 'reference',
    _ref: imageRef,
  },
}
: undefined,
};
await client.create(sanityFood);
}

```

- Example for **Chef**:

```

for (const chef of chefs) {
  const sanityChef = {
    _type: 'chef',
    name: chef.name,
    position: chef.position || null,
    experience: chef.experience || 0,
    specialty: chef.specialty || "",
    description: chef.description || "",
    available: chef.available !== undefined ? chef.available : true,
    image: imageRef
    ? {
      _type: 'image',
      asset: {
        _type: 'reference',
        _ref: imageRef,
      },
    }
    : undefined,
  };
  await client.create(sanityChef);
}

```

5. Running the Migration Script

- I ran the migration script using the following command:

```
npm run import-data
```

- This executed the `import-data.mjs` script, which:
 1. Fetched data from the API.
 2. Uploaded images to Sanity.
 3. Created documents in Sanity for each food and chef item.
-

6. Verifying the Data in Sanity Studio

- After running the script, I opened **Sanity Studio** to verify that the data was successfully imported.

- I checked the **Food** and **Chef** models to ensure that all fields were populated correctly.
-

Tools Used

1. Sanity Client

- The **Sanity client** (@sanity/client) was used to interact with the Sanity CMS.
- It allowed me to:
 - Upload images using `client.assets.upload`.
 - Create documents using `client.create`.

2. Axios

- **Axios** was used to fetch data from the external API.
- It handled HTTP requests and responses efficiently.

3. Node.js

- The migration script was executed using **Node.js**.
- Node.js provided the runtime environment for running the script.

4. Environment Variables

- I used a `.env.local` file to store sensitive environment variables like:
 - `NEXT_PUBLIC_SANITY_PROJECT_ID`
 - `SANITY_API_TOKEN`
 - `NEXT_PUBLIC_SANITY_DATASET`

5. Sanity Studio

- **Sanity Studio** was used to:
 - Define schemas for the **Food** and **Chef** models.
 - Verify that the imported data was correctly populated.
-

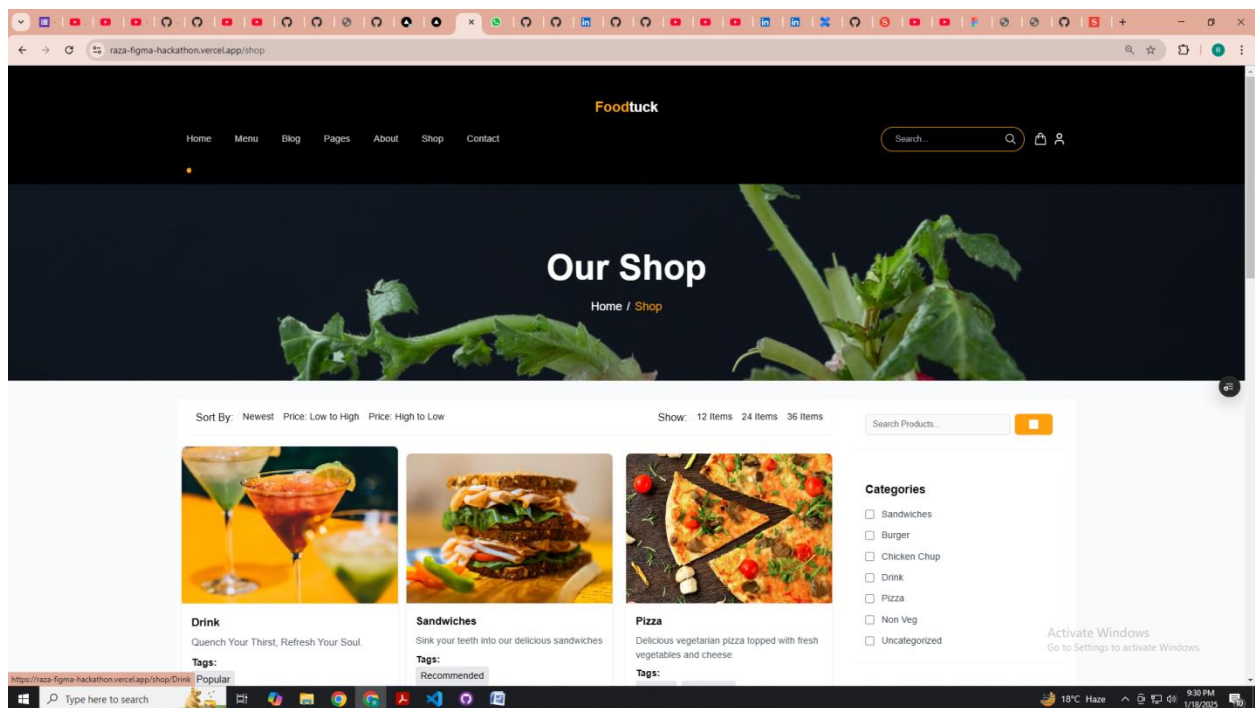
Summary of Migration Steps and Tools

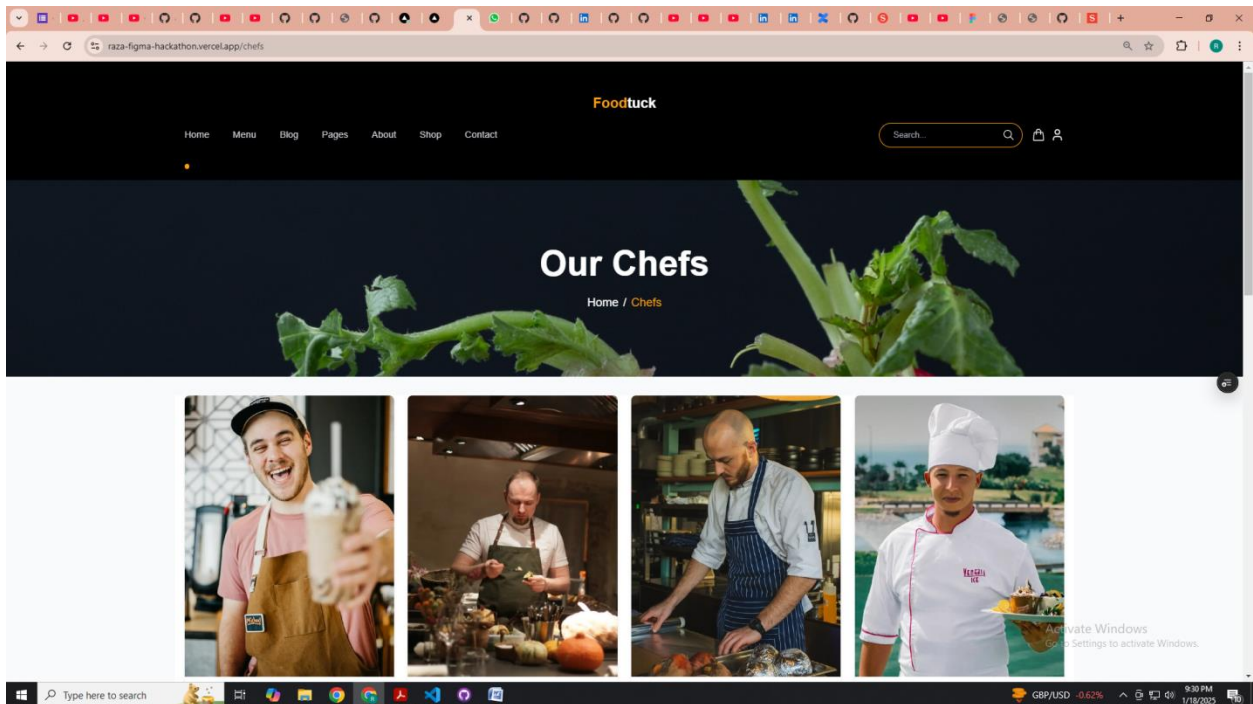
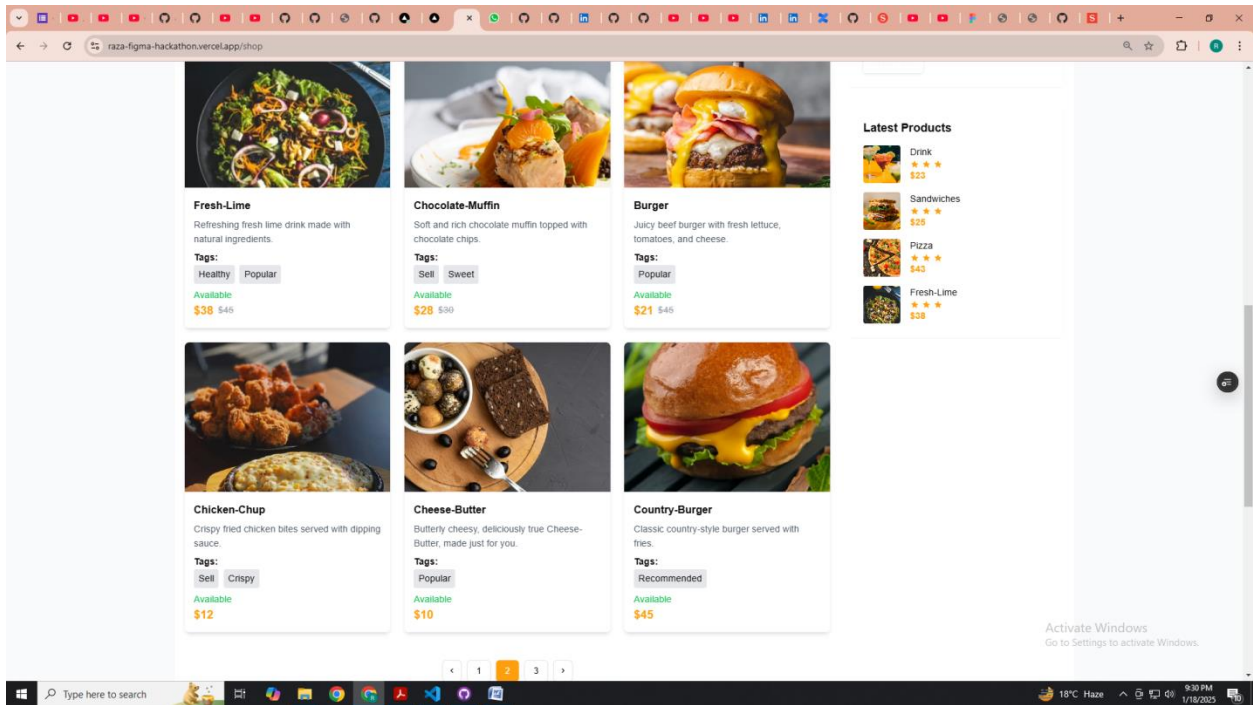
1. **Set Up the Migration Script:**
 - Used `import-data.mjs` to fetch and upload data.
2. **Fetches Data from the API:**
 - Used Axios to fetch food and chef data from the external API.
3. **Uploaded Images to Sanity:**

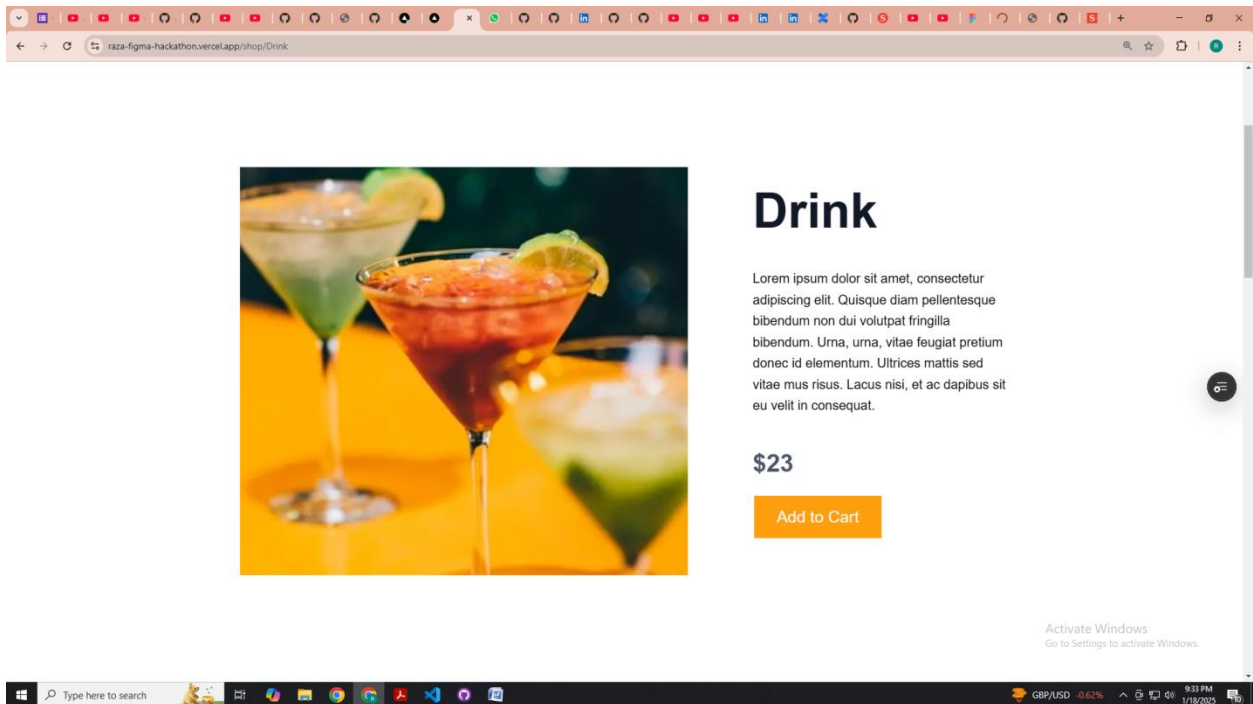
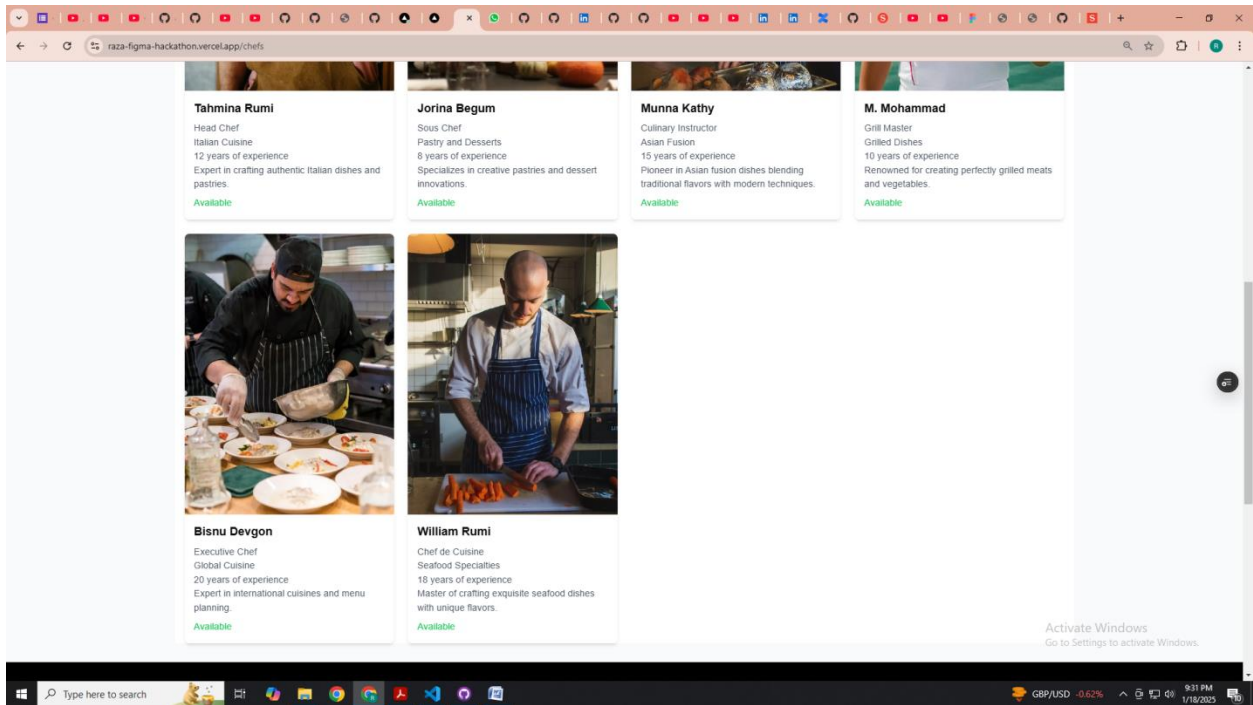
- Used the Sanity client to upload images and create references.
- 4. **Created Documents in Sanity:**
 - Used the Sanity client to create documents for each food and chef item.
- 5. **Ran the Migration Script:**
 - Executed the script using `npm run import-data`.
- 6. **Verified the Data:**
 - Checked Sanity Studio to ensure the data was imported correctly.
- 7. **Tools Used:**
 - Sanity Client, Axios, Node.js, Environment Variables, and Sanity Studio.

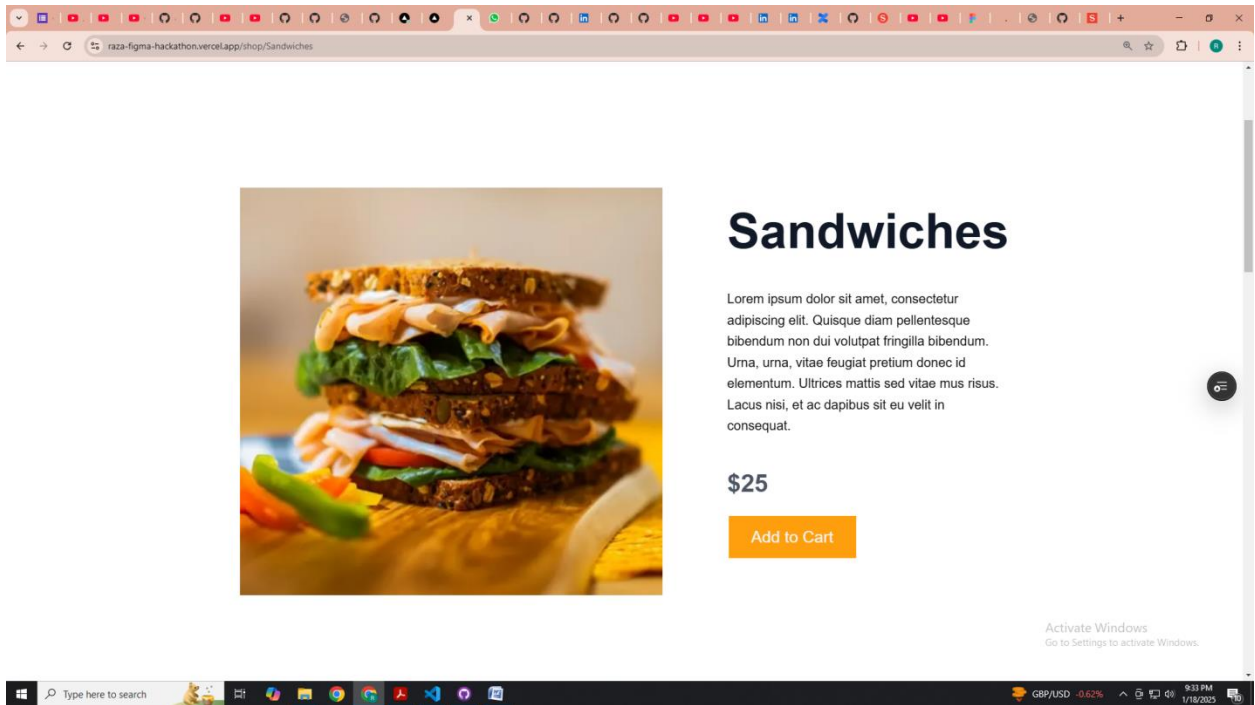
What Worked Well

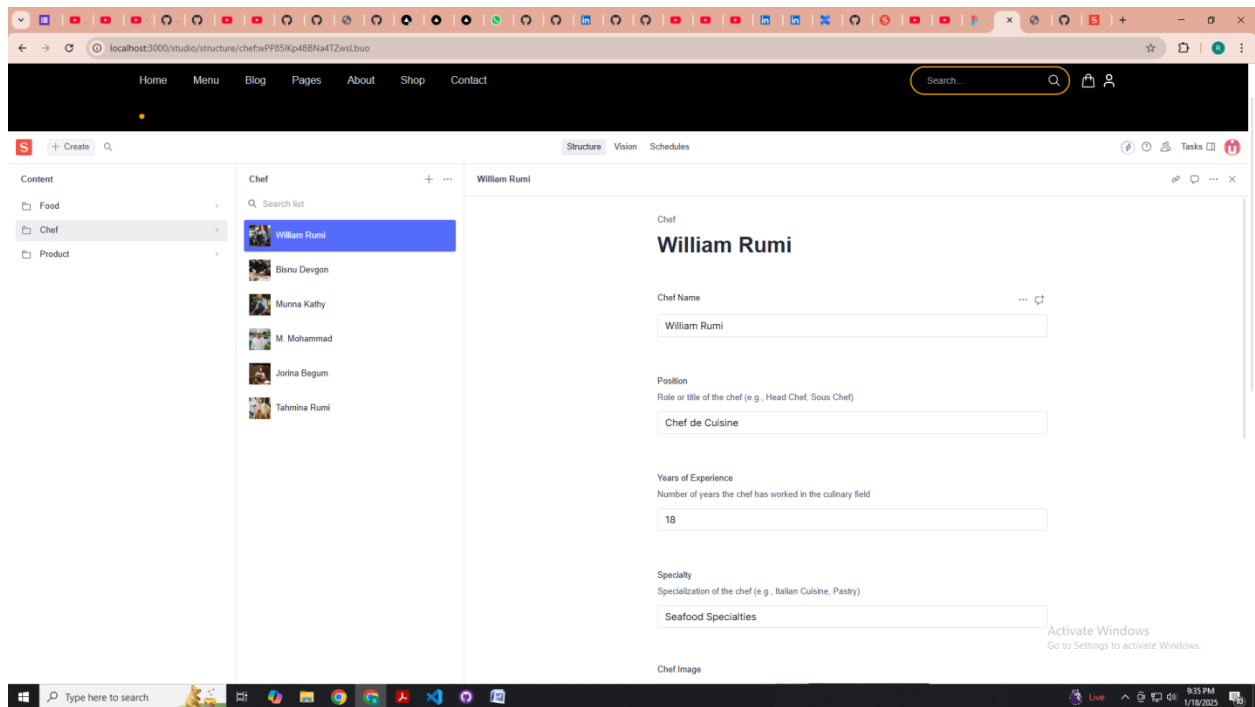
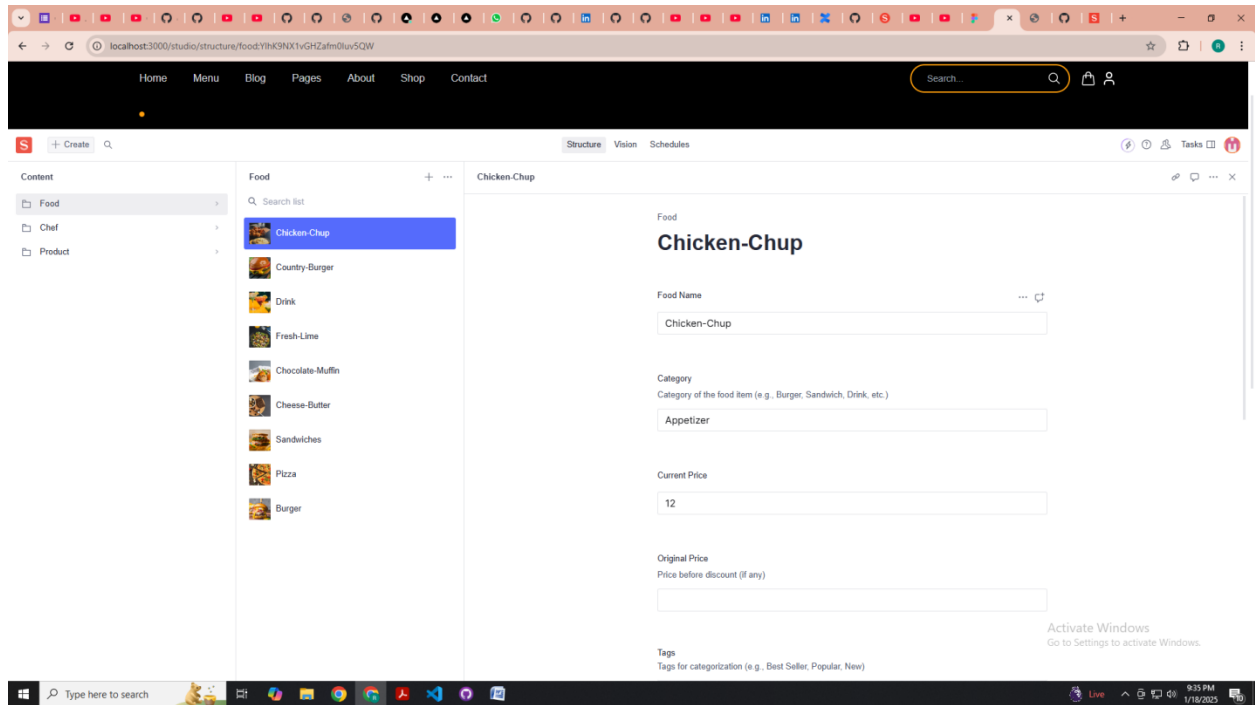
- The migration script successfully fetched data from the API and uploaded it to Sanity CMS.
- The use of environment variables ensured that sensitive information was securely stored.
- Sanity Studio made it easy to verify that the data was correctly imported.

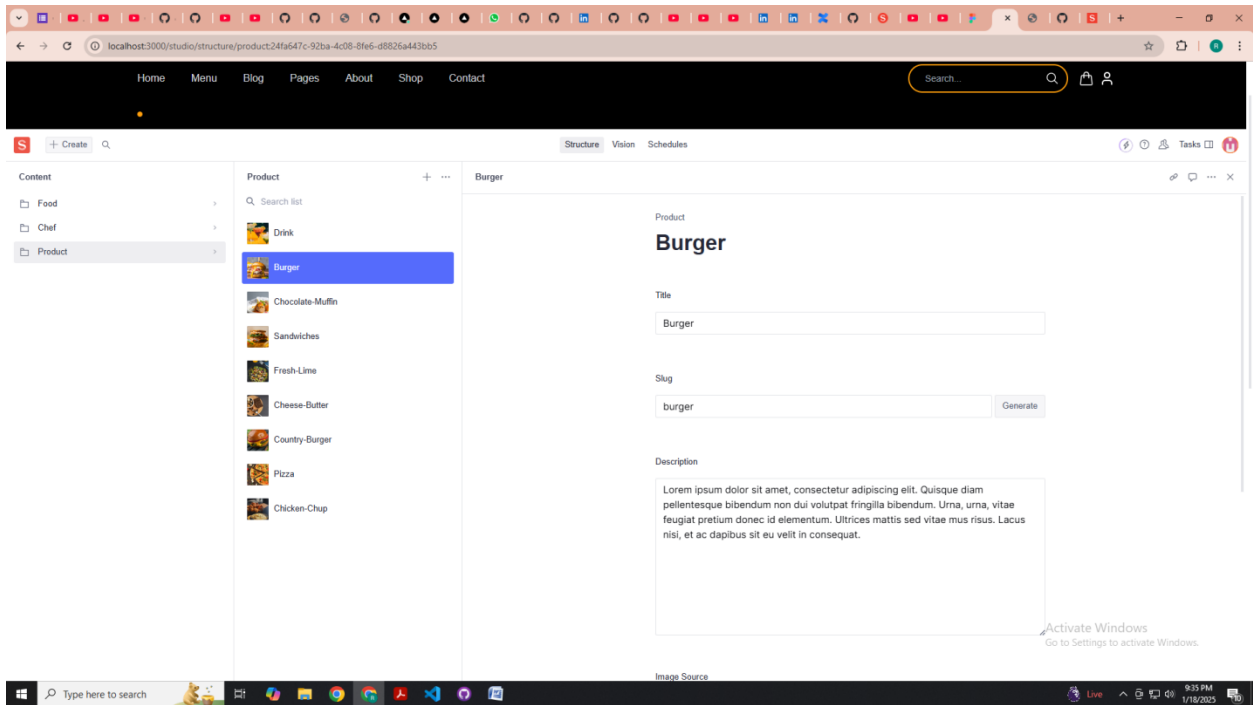












```

1 import { createClient } from '@sanity/client';
2 import axios from 'axios';
3 import dotenv from 'dotenv';
4 import { fileURLToPath } from 'url';
5 import path from 'path';
6
7 // Load environment variables from .env.local
8 const __filename = fileURLToPath(import.meta.url);
9 const __dirname = path.dirname(__filename);
10 dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12 // Create Sanity client
13 const client = createClient({
14   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16   useCdn: false,
17   token: process.env.NEXT_SANITY_TOKEN,
18   apiVersion: '2021-08-31',
19 });
20
21 async function uploadImageToSanity(imageUrl) {
22   try {
23     console.log('Uploading image: ' + imageUrl);
24     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
25     const buffer = Buffer.from(response.data);
26     const asset = await client.assets.upload('image', buffer, {
27       filename: imageUrl.split('/').pop(),
28     });
29     console.log('Image uploaded successfully: ' + asset.id);
30     return asset.id;
31   } catch (error) {
32     console.error('Failed to upload image: ' + imageUrl, error);
33     return null;
34   }
35 }
36
37 async function importData() {
38   try {
39     console.log('Fetching food, chef data from API...');
40
41     // API endpoint containing data
42     const $Promise = [];
43     $Promise.push(
44       axios.get('https://sanity-nextjs-rouge.vercel.app/api/foods')
45     );
46     $Promise.push(
47       axios.get('https://sanity-nextjs-rouge.vercel.app/api/chefs')
48     );
49
50     const [foodsResponse, chefsResponse] = await Promise.all($Promise);
51     const foods = foodsResponse.data;
52     const chefs = chefsResponse.data;
53
54     for (const food of foods) {
55       console.log('Processing food: ' + food.name);
56
57       let imageRef = null;
58       if (food.image) {
59         imageRef = await uploadImageToSanity(food.image);
60       }
61
62       const sanityFood = {
63         _type: 'food',
64         name: food.name,
65         category: food.category || null,
66         price: food.price,
67         originalPrice: food.originalPrice || null,
68         tags: food.tags || [],
69         description: food.description || '',
70         available: food.available !== undefined ? food.available : true,
71         image: imageRef
72       };
73
74       console.log('Uploading food to Sanity:', sanityFood.name);
75       const result = await client.create(sanityFood);
76       console.log('Food uploaded successfully: ' + result._id);
77
78       for (const chef of chefs) {
79         console.log('Processing chef: ' + chef.name);
80
81         let imageRef = null;
82         if (chef.image) {
83           imageRef = await uploadImageToSanity(chef.image);
84         }
85
86         const sanityChef = {
87           _type: 'chef',
88           name: chef.name,
89           position: chef.position || null,
90           experience: chef.experience || 0,
91           specialty: chef.specialty || '',
92           description: chef.description || '',
93           available: chef.available !== undefined ? chef.available : true,
94           image: imageRef
95         };
96
97         console.log('Uploading chef to Sanity:', sanityChef.name);
98         const result = await client.create(sanityChef);
99         console.log('Chef uploaded successfully: ' + result._id);
100       }
101
102       console.log('Data import completed successfully!');
103     } catch (error) {
104       console.error('Error importing data:', error);
105     }
106   }
107 }
108
109 importData();

```

```

1 // sanity client
2
3 // client
4 const client = createClient({
5   projectId: 'xxxxxx',
6   dataset: 'production',
7   useCdn: false,
8   token: 'xxxxxx',
9   apiVersion: '2021-08-31',
10 });
11
12 // sanity client
13 const client = createClient({
14   projectId: 'xxxxxx',
15   dataset: 'production',
16   useCdn: false,
17   token: 'xxxxxx',
18   apiVersion: '2021-08-31',
19 });
20
21 // sanity client
22 const client = createClient({
23   projectId: 'xxxxxx',
24   dataset: 'production',
25   useCdn: false,
26   token: 'xxxxxx',
27   apiVersion: '2021-08-31',
28 });
29
30 // sanity client
31 const client = createClient({
32   projectId: 'xxxxxx',
33   dataset: 'production',
34   useCdn: false,
35   token: 'xxxxxx',
36   apiVersion: '2021-08-31',
37 });
38
39 // sanity client
40 const client = createClient({
41   projectId: 'xxxxxx',
42   dataset: 'production',
43   useCdn: false,
44   token: 'xxxxxx',
45   apiVersion: '2021-08-31',
46 });
47
48 // sanity client
49 const client = createClient({
50   projectId: 'xxxxxx',
51   dataset: 'production',
52   useCdn: false,
53   token: 'xxxxxx',
54   apiVersion: '2021-08-31',
55 });
56
57 // sanity client
58 const client = createClient({
59   projectId: 'xxxxxx',
60   dataset: 'production',
61   useCdn: false,
62   token: 'xxxxxx',
63   apiVersion: '2021-08-31',
64 });
65
66 // sanity client
67 const client = createClient({
68   projectId: 'xxxxxx',
69   dataset: 'production',
70   useCdn: false,
71   token: 'xxxxxx',
72   apiVersion: '2021-08-31',
73 });
74
75 // sanity client
76 const client = createClient({
77   projectId: 'xxxxxx',
78   dataset: 'production',
79   useCdn: false,
80   token: 'xxxxxx',
81   apiVersion: '2021-08-31',
82 });
83
84 // sanity client
85 const client = createClient({
86   projectId: 'xxxxxx',
87   dataset: 'production',
88   useCdn: false,
89   token: 'xxxxxx',
90   apiVersion: '2021-08-31',
91 });
92
93 // sanity client
94 const client = createClient({
95   projectId: 'xxxxxx',
96   dataset: 'production',
97   useCdn: false,
98   token: 'xxxxxx',
99   apiVersion: '2021-08-31',
100 });
101
102 // sanity client
103 const client = createClient({
104   projectId: 'xxxxxx',
105   dataset: 'production',
106   useCdn: false,
107   token: 'xxxxxx',
108   apiVersion: '2021-08-31',
109 });
110
111 // sanity client
112 const client = createClient({
113   projectId: 'xxxxxx',
114   dataset: 'production',
115   useCdn: false,
116   token: 'xxxxxx',
117   apiVersion: '2021-08-31',
118 });
119
120 // sanity client
121 const client = createClient({
122   projectId: 'xxxxxx',
123   dataset: 'production',
124   useCdn: false,
125   token: 'xxxxxx',
126   apiVersion: '2021-08-31',
127 });
128
129 // sanity client
130 const client = createClient({
131   projectId: 'xxxxxx',
132   dataset: 'production',
133   useCdn: false,
134   token: 'xxxxxx',
135   apiVersion: '2021-08-31',
136 });
137
138 // sanity client
139 const client = createClient({
140   projectId: 'xxxxxx',
141   dataset: 'production',
142   useCdn: false,
143   token: 'xxxxxx',
144   apiVersion: '2021-08-31',
145 });
146
147 // sanity client
148 const client = createClient({
149   projectId: 'xxxxxx',
150   dataset: 'production',
151   useCdn: false,
152   token: 'xxxxxx',
153   apiVersion: '2021-08-31',
154 });
155
156 // sanity client
157 const client = createClient({
158   projectId: 'xxxxxx',
159   dataset: 'production',
160   useCdn: false,
161   token: 'xxxxxx',
162   apiVersion: '2021-08-31',
163 });
164
165 // sanity client
166 const client = createClient({
167   projectId: 'xxxxxx',
168   dataset: 'production',
169   useCdn: false,
170   token: 'xxxxxx',
171   apiVersion: '2021-08-31',
172 });
173
174 // sanity client
175 const client = createClient({
176   projectId: 'xxxxxx',
177   dataset: 'production',
178   useCdn: false,
179   token: 'xxxxxx',
180   apiVersion: '2021-08-31',
181 });
182
183 // sanity client
184 const client = createClient({
185   projectId: 'xxxxxx',
186   dataset: 'production',
187   useCdn: false,
188   token: 'xxxxxx',
189   apiVersion: '2021-08-31',
190 });
191
192 // sanity client
193 const client = createClient({
194   projectId: 'xxxxxx',
195   dataset: 'production',
196   useCdn: false,
197   token: 'xxxxxx',
198   apiVersion: '2021-08-31',
199 });
200
201 // sanity client
202 const client = createClient({
203   projectId: 'xxxxxx',
204   dataset: 'production',
205   useCdn: false,
206   token: 'xxxxxx',
207   apiVersion: '2021-08-31',
208 });
209
210 // sanity client
211 const client = createClient({
212   projectId: 'xxxxxx',
213   dataset: 'production',
214   useCdn: false,
215   token: 'xxxxxx',
216   apiVersion: '2021-08-31',
217 });
218
219 // sanity client
220 const client = createClient({
221   projectId: 'xxxxxx',
222   dataset: 'production',
223   useCdn: false,
224   token: 'xxxxxx',
225   apiVersion: '2021-08-31',
226 });
227
228 // sanity client
229 const client = createClient({
230   projectId: 'xxxxxx',
231   dataset: 'production',
232   useCdn: false,
233   token: 'xxxxxx',
234   apiVersion: '2021-08-31',
235 });
236
237 // sanity client
238 const client = createClient({
239   projectId: 'xxxxxx',
240   dataset: 'production',
241   useCdn: false,
242   token: 'xxxxxx',
243   apiVersion: '2021-08-31',
244 });
245
246 // sanity client
247 const client = createClient({
248   projectId: 'xxxxxx',
249   dataset: 'production',
250   useCdn: false,
251   token: 'xxxxxx',
252   apiVersion: '2021-08-31',
253 });
254
255 // sanity client
256 const client = createClient({
257   projectId: 'xxxxxx',
258   dataset: 'production',
259   useCdn: false,
260   token: 'xxxxxx',
261   apiVersion: '2021-08-31',
262 });
263
264 // sanity client
265 const client = createClient({
266   projectId: 'xxxxxx',
267   dataset: 'production',
268   useCdn: false,
269   token: 'xxxxxx',
270   apiVersion: '2021-08-31',
271 });
272
273 // sanity client
274 const client = createClient({
275   projectId: 'xxxxxx',
276   dataset: 'production',
277   useCdn: false,
278   token: 'xxxxxx',
279   apiVersion: '2021-08-31',
280 });
281
282 // sanity client
283 const client = createClient({
284   projectId: 'xxxxxx',
285   dataset: 'production',
286   useCdn: false,
287   token: 'xxxxxx',
288   apiVersion: '2021-08-31',
289 });
290
291 // sanity client
292 const client = createClient({
293   projectId: 'xxxxxx',
294   dataset: 'production',
295   useCdn: false,
296   token: 'xxxxxx',
297   apiVersion: '2021-08-31',
298 });
299
300 // sanity client
301 const client = createClient({
302   projectId: 'xxxxxx',
303   dataset: 'production',
304   useCdn: false,
305   token: 'xxxxxx',
306   apiVersion: '2021-08-31',
307 });
308
309 // sanity client
310 const client = createClient({
311   projectId: 'xxxxxx',
312   dataset: 'production',
313   useCdn: false,
314   token: 'xxxxxx',
315   apiVersion: '2021-08-31',
316 });
317
318 // sanity client
319 const client = createClient({
320   projectId: 'xxxxxx',
321   dataset: 'production',
322   useCdn: false,
323   token: 'xxxxxx',
324   apiVersion: '2021-08-31',
325 });
326
327 // sanity client
328 const client = createClient({
329   projectId: 'xxxxxx',
330   dataset: 'production',
331   useCdn: false,
332   token: 'xxxxxx',
333   apiVersion: '2021-08-31',
334 });
335
336 // sanity client
337 const client = createClient({
338   projectId: 'xxxxxx',
339   dataset: 'production',
340   useCdn: false,
341   token: 'xxxxxx',
342   apiVersion: '2021-08-31',
343 });
344
345 // sanity client
346 const client = createClient({
347   projectId: 'xxxxxx',
348   dataset: 'production',
349   useCdn: false,
350   token: 'xxxxxx',
351   apiVersion: '2021-08-31',
352 });
353
354 // sanity client
355 const client = createClient({
356   projectId: 'xxxxxx',
357   dataset: 'production',
358   useCdn: false,
359   token: 'xxxxxx',
360   apiVersion: '2021-08-31',
361 });
362
363 // sanity client
364 const client = createClient({
365   projectId: 'xxxxxx',
366   dataset: 'production',
367   useCdn: false,
368   token: 'xxxxxx',
369   apiVersion: '2021-08-31',
370 });
371
372 // sanity client
373 const client = createClient({
374   projectId: 'xxxxxx',
375   dataset: 'production',
376   useCdn: false,
377   token: 'xxxxxx',
378   apiVersion: '2021-08-31',
379 });
380
381 // sanity client
382 const client = createClient({
383   projectId: 'xxxxxx',
384   dataset: 'production',
385   useCdn: false,
386   token: 'xxxxxx',
387   apiVersion: '2021-08-31',
388 });
389
390 // sanity client
391 const client = createClient({
392   projectId: 'xxxxxx',
393   dataset: 'production',
394   useCdn: false,
395   token: 'xxxxxx',
396   apiVersion: '2021-08-31',
397 });
398
399 // sanity client
400 const client = createClient({
401   projectId: 'xxxxxx',
402   dataset: 'production',
403   useCdn: false,
404   token: 'xxxxxx',
405   apiVersion: '2021-08-31',
406 });
407
408 // sanity client
409 const client = createClient({
410   projectId: 'xxxxxx',
411   dataset: 'production',
412   useCdn: false,
413   token: 'xxxxxx',
414   apiVersion: '2021-08-31',
415 });
416
417 // sanity client
418 const client = createClient({
419   projectId: 'xxxxxx',
420   dataset: 'production',
421   useCdn: false,
422   token: 'xxxxxx',
423   apiVersion: '2021-08-31',
424 });
425
426 // sanity client
427 const client = createClient({
428   projectId: 'xxxxxx',
429   dataset: 'production',
430   useCdn: false,
431   token: 'xxxxxx',
432   apiVersion: '2021-08-31',
433 });
434
435 // sanity client
436 const client = createClient({
437   projectId: 'xxxxxx',
438   dataset: 'production',
439   useCdn: false,
440   token: 'xxxxxx',
441   apiVersion: '2021-08-31',
442 });
443
444 // sanity client
445 const client = createClient({
446   projectId: 'xxxxxx',
447   dataset: 'production',
448   useCdn: false,
449   token: 'xxxxxx',
450   apiVersion: '2021-08-31',
451 });
452
453 // sanity client
454 const client = createClient({
455   projectId: 'xxxxxx',
456   dataset: 'production',
457   useCdn: false,
458   token: 'xxxxxx',
459   apiVersion: '2021-08-31',
460 });
461
462 // sanity client
463 const client = createClient({
464   projectId: 'xxxxxx',
465   dataset: 'production',
466   useCdn: false,
467   token: 'xxxxxx',
468   apiVersion: '2021-08-31',
469 });
470
471 // sanity client
472 const client = createClient({
473   projectId: 'xxxxxx',
474   dataset: 'production',
475   useCdn: false,
476   token: 'xxxxxx',
477   apiVersion: '2021-08-31',
478 });
479
480 // sanity client
481 const client = createClient({
482   projectId: 'xxxxxx',
483   dataset: 'production',
484   useCdn: false,
485   token: 'xxxxxx',
486   apiVersion: '2021-08-31',
487 });
488
489 // sanity client
490 const client = createClient({
491   projectId: 'xxxxxx',
492   dataset: 'production',
493   useCdn: false,
494   token: 'xxxxxx',
495   apiVersion: '2021-08-31',
496 });
497
498 // sanity client
499 const client = createClient({
500   projectId: 'xxxxxx',
501   dataset: 'production',
502   useCdn: false,
503   token: 'xxxxxx',
504   apiVersion: '2021-08-31',
505 });
506
507 // sanity client
508 const client = createClient({
509   projectId: 'xxxxxx',
510   dataset: 'production',
511   useCdn: false,
512   token: 'xxxxxx',
513   apiVersion: '2021-08-31',
514 });
515
516 // sanity client
517 const client = createClient({
518   projectId: 'xxxxxx',
519   dataset: 'production',
520   useCdn: false,
521   token: 'xxxxxx',
522   apiVersion: '2021-08-31',
523 });
524
525 // sanity client
526 const client = createClient({
527   projectId: 'xxxxxx',
528   dataset: 'production',
529   useCdn: false,
530   token: 'xxxxxx',
531   apiVersion: '2021-08-31',
532 });
533
534 // sanity client
535 const client = createClient({
536   projectId: 'xxxxxx',
537   dataset: 'production',
538   useCdn: false,
539   token: 'xxxxxx',
540   apiVersion: '2021-08-31',
541 });
542
543 // sanity client
544 const client = createClient({
545   projectId: 'xxxxxx',
546   dataset: 'production',
547   useCdn: false,
548   token: 'xxxxxx',
549   apiVersion: '2021-08-31',
550 });
551
552 // sanity client
553 const client = createClient({
554   projectId: 'xxxxxx',
555   dataset: 'production',
556   useCdn: false,
557   token: 'xxxxxx',
558   apiVersion: '2021-08-31',
559 });
560
561 // sanity client
562 const client = createClient({
563   projectId: 'xxxxxx',
564   dataset: 'production',
565   useCdn: false,
566   token: 'xxxxxx',
567   apiVersion: '2021-08-31',
568 });
569
570 // sanity client
571 const client = createClient({
572   projectId: 'xxxxxx',
573   dataset: 'production',
574   useCdn: false,
575   token: 'xxxxxx',
576   apiVersion: '2021-08-31',
577 });
578
579 // sanity client
580 const client = createClient({
581   projectId: 'xxxxxx',
582   dataset: 'production',
583   useCdn: false,
584   token: 'xxxxxx',
585   apiVersion: '2021-08-31',
586 });
587
588 // sanity client
589 const client = createClient({
590   projectId: 'xxxxxx',
591   dataset: 'production',
592   useCdn: false,
593   token: 'xxxxxx',
594   apiVersion: '2021-08-31',
595 });
596
597 // sanity client
598 const client = createClient({
599   projectId: 'xxxxxx',
600   dataset: 'production',
601   useCdn: false,
602   token: 'xxxxxx',
603   apiVersion: '2021-08-31',
604 });
605
606 // sanity client
607 const client = createClient({
608   projectId: 'xxxxxx',
609   dataset: 'production',
610   useCdn: false,
611   token: 'xxxxxx',
612   apiVersion: '2021-08-31',
613 });
614
615 // sanity client
616 const client = createClient({
617   projectId: 'xxxxxx',
618   dataset: 'production',
619   useCdn: false,
620   token: 'xxxxxx',
621   apiVersion: '2021-08-31',
622 });
623
624 // sanity client
625 const client = createClient({
626   projectId: 'xxxxxx',
627   dataset: 'production',
628   useCdn: false,
629   token: 'xxxxxx',
630   apiVersion: '2021-08-31',
631 });
632
633 // sanity client
634 const client = createClient({
635   projectId: 'xxxxxx',
636   dataset: 'production',
637   useCdn: false,
638   token: 'xxxxxx',
639   apiVersion: '2021-08-31',
640 });
641
642 // sanity client
643 const client = createClient({
644   projectId: 'xxxxxx',
645   dataset: 'production',
646   useCdn: false,
647   token: 'xxxxxx',
648   apiVersion: '2021-08-31',
649 });
650
651 // sanity client
652 const client = createClient({
653   projectId: 'xxxxxx',
654   dataset: 'production',
655   useCdn: false,
656   token: 'xxxxxx',
657   apiVersion: '2021-08-31',
658 });
659
660 // sanity client
661 const client = createClient({
662   projectId: 'xxxxxx',
663   dataset: 'production',
664   useCdn: false,
665   token: 'xxxxxx',
666   apiVersion: '2021-08-31',
667 });
668
669 // sanity client
670 const client = createClient({
671   projectId: 'xxxxxx',
672   dataset: 'production',
673   useCdn: false,
674   token: 'xxxxxx',
675   apiVersion: '2021-08-31',
676 });
677
678 // sanity client
679 const client = createClient({
680   projectId: 'xxxxxx',
681   dataset: 'production',
682   useCdn: false,
683   token: 'xxxxxx',
684   apiVersion: '2021-08-31',
685 });
686
687 // sanity client
688 const client = createClient({
689   projectId: 'xxxxxx',
690   dataset: 'production',
691   useCdn: false,
692   token: 'xxxxxx',
693   apiVersion: '2021-08-31',
694 });
695
696 // sanity client
697 const client = createClient({
698   projectId: 'xxxxxx',
699   dataset: 'production',
700   useCdn: false,
701   token: 'xxxxxx',
702   apiVersion: '2021-08-31',
703 });
704
705 // sanity client
706 const client = createClient({
707   projectId: 'xxxxxx',
708   dataset: 'production',
709   useCdn: false,
710   token: 'xxxxxx',
711   apiVersion: '2021-08-31',
712 });
713
714 // sanity client
715 const client = createClient({
716   projectId: 'xxxxxx',
717   dataset: 'production',
718   useCdn: false,
719   token: 'xxxxxx',
720   apiVersion: '2021-08-31',
721 });
722
723 // sanity client
724 const client = createClient({
725   projectId: 'xxxxxx',
726   dataset: 'production',
727   useCdn: false,
728   token: 'xxxxxx',
729   apiVersion: '2021-08-31',
730 });
731
732 // sanity client
733 const client = createClient({
734   projectId: 'xxxxxx',
735   dataset: 'production',
736   useCdn: false,
737   token: 'xxxxxx',
738   apiVersion: '2021-08-31',
739 });
740
741 // sanity client
742 const client = createClient({
743   projectId: 'xxxxxx',
744   dataset: 'production',
745   useCdn: false,
746   token: 'xxxxxx',
747   apiVersion: '2021-08-31',
748 });
749
750 // sanity client
751 const client = createClient({
752   projectId: 'xxxxxx',
753   dataset: 'production',
754   useCdn: false,
755   token: 'xxxxxx',
756   apiVersion: '2021-08-31',
757 });
758
759 // sanity client
760 const client = createClient({
761   projectId: 'xxxxxx',
762   dataset: 'production',
763   useCdn: false,
764   token: 'xxxxxx',
765   apiVersion: '2021-08-31',
766 });
767
768 // sanity client
769 const client = createClient({
770   projectId: 'xxxxxx',
771   dataset: 'production',
772   useCdn: false,
773   token: 'xxxxxx',
774   apiVersion: '2021-08-31',
775 });
776
777 // sanity client
778 const client = createClient({
779   projectId: 'xxxxxx',
780   dataset: 'production',
781   useCdn: false,
782   token: 'xxxxxx',
783   apiVersion: '2021-08-31',
784 });
785
786 // sanity client
787 const client = createClient({
788   projectId: 'xxxxxx',
789   dataset: 'production',
790   useCdn: false,
791   token: 'xxxxxx',
792   apiVersion: '2021-08-31',
793 });
794
795 // sanity client
796 const client = createClient({
797   projectId: 'xxxxxx',
798   dataset: 'production',
799   useCdn: false,
800   token: 'xxxxxx',
801   apiVersion: '2021-08-31',
802 });
803
804 // sanity client
805 const client = createClient({
806   projectId: 'xxxxxx',
807   dataset: 'production',
808   useCdn: false,
809   token: 'xxxxxx',
810   apiVersion: '2021-08-31',
811 });
812
813 // sanity client
814 const client = createClient({
815   projectId: 'xxxxxx',
816   dataset: 'production',
817   useCdn: false,
818   token: 'xxxxxx',
819   apiVersion: '2021-08-31',
820 });
821
822 // sanity client
823 const client = createClient({
824   projectId: 'xxxxxx',
825   dataset: 'production',
826   useCdn: false,
827   token: 'xxxxxx',
828   apiVersion: '2021-08-31',
829 });
830
831 // sanity client
832 const client = createClient({
833   projectId: 'xxxxxx',
834   dataset: 'production',
835   useCdn: false,
836   token: 'xxxxxx',
837   apiVersion: '2021-08-31',
838 });
839
840 // sanity client
841 const client = createClient({
842   projectId: 'xxxxxx',
843   dataset: 'production',
844   useCdn: false,
845   token: 'xxxxxx',
846   apiVersion: '2021-08-31',
847 });
848
849 // sanity client
850 const client = createClient({
851   projectId: 'xxxxxx',
852   dataset: 'production',
853   useCdn: false,
854   token: 'xxxxxx',
855   apiVersion: '2021-08-31',
856 });
857
858 // sanity client
859 const client = createClient({
860   projectId: 'xxxxxx',
861   dataset: 'production',
862   useCdn: false,
863   token: 'xxxxxx',
864   apiVersion: '2021-08-31',
865 });
866
867 // sanity client
868 const client = createClient({
869   projectId: 'xxxxxx',
870   dataset: 'production',
871   useCdn: false,
872   token: 'xxxxxx',
873   apiVersion: '2021-08-31',
874 });
875
876 // sanity client
877 const client = createClient({
878   projectId: 'xxxxxx',
879   dataset: 'production',
880   useCdn: false,
881   token: 'xxxxxx',
882   apiVersion: '2021-08-31',
883 });
884
885 // sanity client
886 const client = createClient({
887   projectId: 'xxxxxx',
888   dataset: 'production',
889   useCdn: false,
890   token: 'xxxxxx',
891   apiVersion: '2021-08-31',
892 });
893
894 // sanity client
895 const client = createClient({
896   projectId: 'xxxxxx',
897   dataset: 'production',
898   useCdn: false,
899   token: 'xxxxxx',
900   apiVersion: '2021-08-31',
901 });
902
903 // sanity client
904 const client = createClient({
905   projectId: 'xxxxxx',
906   dataset: 'production',
907   useCdn: false,
908   token: 'xxxxxx',
909   apiVersion: '2021-08-31',
910 });
911
912 // sanity client
913 const client = createClient({
914   projectId: 'xxxxxx',
915   dataset: 'production',
916   useCdn: false,
917   token: 'xxxxxx',
918   apiVersion: '2021-08-31',
919 });
920
921 // sanity client
922 const client = createClient({
923   projectId: 'xxxxxx',
924   dataset: 'production',
925   useCdn: false,
926   token: 'xxxxxx',
927   apiVersion: '2021-08-31',
928 });
929
930 // sanity client
931 const client = createClient({
932   projectId: 'xxxxxx',
933   dataset: 'production',
934   useCdn: false,
935   token: 'xxxxxx',
936   apiVersion: '2021-08-31',
937 });
938
939 // sanity client
940 const client = createClient({
941   projectId: 'xxxxxx',
942   dataset: 'production',
943   useCdn: false,
944   token: 'xxxxxx',
945   apiVersion: '2021-08-31',
946 });
947
948 // sanity client
949 const client = createClient({
950   projectId: 'xxxxxx',
951   dataset: 'production',
952   useCdn: false,
953   token: 'xxxxxx',
954   apiVersion: '2021-08-31',
955 });
956
957 // sanity client
958 const client = createClient({
959   projectId: 'xxxxxx',
960   dataset: 'production',
961   useCdn: false,
962   token: 'xxxxxx',
963   apiVersion: '2021-08-31',
964 });
965
966 // sanity client
967 const client = createClient({
968   projectId: 'xxxxxx',
969   dataset: 'production',
970   useCdn: false,
971   token: 'xxxxxx',
972   apiVersion: '2021-08-31',
973 });
974
975 // sanity client
976 const client = createClient({
977   projectId: 'xxxxxx',
978   dataset: 'production',
979   useCdn: false,
980   token: 'xxxxxx',
981   apiVersion: '2021-08-31',
982 });
983
984 // sanity client
985 const client = createClient({
986   projectId: 'xxxxxx',
987   dataset: 'production',
988   useCdn: false,
989   token: 'xxxxxx',
990   apiVersion: '2021-08-31',
991 });
992
993 // sanity client
994 const client = createClient({
995   projectId: 'xxxxxx',
996   dataset: 'production',
997   useCdn: false,
998   token: 'xxxxxx',
999   apiVersion: '2021-08-31',
1000 });

```

```

1  "use client";
2
3  import { client } from "@sanity/lib/client";
4  import { useState, useEffect } from "react";
5  import { urlFor } from "@sanity/lib/image";
6  import Image from "next/image";
7
8  interface Chef {
9    _id: string;
10   name: string;
11   position: string;
12   experience: number;
13   specialty: string;
14   image: { asset: { url: string } };
15   description: string;
16   available: boolean;
17 }
18
19 export default function ChefComponent() {
20   const [chefs, setChefs] = useState<Chef[]>([]);
21   const [isLoading, setIsLoading] = useState(true);
22   const [error, setError] = useState<string | null>(null);
23
24   // Fetch chef data on component mount
25   useEffect(() => {
26     const fetchChefs = async () => {
27       setIsLoading(true);
28       try {
29         // Fetch all chefs
30         const chefQuery = `*[ _type == "chef" ] {
31           _id,
32           name,
33           position,
34           experience,
35           specialty,
36           image,
37           description,
38           available
39         }`;
40         const chefs = await client.fetch(chefQuery);
41         setChefs(chefs);
42       } catch (error) {
43         console.error("Error fetching chefs:", error);
44         setError("Failed to fetch chefs. Please try again later.");
45       } finally {
46         setIsLoading(false);
47       }
48     };
49
50     fetchChefs();
51   }, []);
52
53   if (isLoading) {
54     return <div className=" bg-white flex justify-center items-center h-screen">loading chefs...</div>;
55   }
56
57   if (error) {
58     return <div className=" bg-white flex justify-center items-center h-screen">{error}</div>;
59   }
60
61   if (!chefs || chefs.length === 0) {
62     return (
63       <div className=" bg-white flex justify-center items-center h-screen">
64         <h1 className=" bg-white">Chef List</h1>
65         <p className=" bg-white">No chefs available. Please check the following:</p>
66         <ul className=" bg-white">
67           <li className=" bg-white">Is the Sanity client correctly configured?</li>
68           <li className=" bg-white">Does the dataset contain chef data?</li>
69           <li className=" bg-white">Is the GROQ query correct?</li>
70           <li className=" bg-white">Are there any errors in the terminal logs?</li>
71         </ul>
72       </div>
73     );
74   }
75
76   return (
77     <div className=" bg-gray-50 min-h-screen py-8">
78       <div className=" bg-white container mx-auto px-4">
79         <div className=" bg-white grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-6">
80           {chefs.map((chef) => (
81             <div key={chef._id} className=" bg-white rounded-lg shadow-md overflow-hidden transform transition-transform hover:scale-105">
82               <div className=" bg-white relative h-[30rem] w-full">
83                 <img
84                   src={urlFor(chef.image).url()}
85                   alt={chef.name}
86                   fill
87                   className=" bg-white object-cover"
88                 />
89               </div>
90               <div>
91                 <div className=" bg-white p-4">
92                   <h2 className=" bg-white text-xl font-semibold mb-2">{chef.name}</h2>
93                   <p className=" bg-white text-gray-600">{chef.position}</p>
94                   <p className=" bg-white text-gray-600">{chef.specialty}</p>
95                   <p className=" bg-white text-gray-600">{chef.experience} years of experience</p>
96                   <p className=" bg-white text-gray-600">{chef.description}</p>
97                   <p className=" bg-white mt-2">{(chef.available ? "text-green-500" : "text-red-500")} }>
98                     {chef.available ? "Available" : "Not Available"}
99                   </p>
100                 </div>
101               </div>
102             </div>
103           ))}
104         </div>
105       </div>
106     );
107   }

```

