Document Title: "Day 3 - API Integration Report - [Foodtuck]"

REPORT :

Day 3 - API Integration and Data Migration
Prepared by: Maria Khan
Date: [18-January-2024]

---

1. API Integration Process

1.1. Overview of API Integration

The integration process involved connecting the application with a third-party API to enable dynamic data fetching and synchronization. This required a step-by-step approach to ensure smooth functionality, data consistency, and minimal downtime. The API serves as a bridge between the backend and frontend, offering features such as fetching real-time data, updating records, and managing additional functionalities required by the application.

Key Objectives:

- Establish secure connectivity with the API.
- Fetch and process data to align with application requirements.
- Implement error handling for robust performance.

1.2. Steps for Integration

1. **Authentication Setup**
    - The API required secure authentication using an API key or token.
    - The key was stored in a .env file to prevent hardcoding and ensure security during deployment.
    - Authentication was tested using Postman to verify the API responded with valid tokens and allowed access to the endpoints.
2. **Endpoint Exploration**
    - Thoroughly reviewed the API documentation to identify required endpoints.
    - Endpoints integrated:

        - GET /[endpoint] - For retrieving data (e.g., products, chefs).
        - POST /[endpoint] - For sending or updating data when necessary.
    - Ensured that endpoints were paginated, where applicable, to handle large datasets.

3. **Data Fetching and Processing**
    - Used the axios library for making HTTP requests due to its ease of use and robust error-handling capabilities.
    - Retrieved data in JSON format and parsed it to match the schema requirements.
    - Additional logic was implemented to filter, sort, and group data based on frontend display needs.
4. **Error Handling and Logging**
    - Added robust error-handling mechanisms to catch network errors, invalid responses, and timeout scenarios.
    - Logged errors into the console for debugging during development and configured alerts for deployment environments.
5. **Testing and Debugging**
    - Verified integration by testing API calls using Postman.
    - Used mock data to simulate different scenarios, including empty responses, partial data, and erroneous payloads.
    - Automated tests were written to ensure the integrity of the integration across different use cases.

1.3. Challenges and Solutions

- **Challenge:** The API occasionally returned incomplete or inconsistent data.
    - **Solution:** Added validation logic to handle missing fields gracefully and display fallback content in the UI.

2. Adjustments Made to Schemas

2.1. Schema Before Adjustment

The initial schema was minimal and lacked essential fields required for integration and extended functionality.

Example of Old Schema:

```
{
  "name": "string",
  "price": "number",
  "price": "number",
  "image": "string",


}
```

2.2. Revised Schema

The schema was updated to include additional fields that align with the API response format and enhance the application’s capability to display more relevant and structured data.

Example of Updated Schema:

```
{
    "name": "string",
    "category": "string",
    "price": number,
    "originalPrice": number
    "image": "string",
    "description": "string.",
    "available": string
}
```

2.3. Details of Changes

1. **Added New Fields**
    - image: To store URLs of images fetched from the API, allowing for dynamic image rendering.
    - available: To indicate availability status (e.g., in stock or currently serving).
    - category: To classify data for better organization (e.g., product categories, chef specialties).
    - experience: Added for chefs to denote their professional experience in years.
    - specialty: Captures unique skills or specialties relevant to chefs or other entities.

2.4. Reason for Schema Adjustments

- The previous schema was insufficient for handling additional details provided by the API, such as dynamic images and categorized data.
- Enhancements were made to support better UI design and data presentation, enabling richer functionality and a more professional user experience.

---

3. Migration Steps and Tools Used

Migration Steps for API Integration:

1. Cloning the Repository

- **The instructor provided a repository containing the necessary API setup and configurations.**
- **Cloned the repository using the following command:**
- **git clone <repository_url>**
- **Verified the repository files and structure to understand the API implementation.**

---

2. Setting Up the Environment

- **Ensured that the required dependencies were installed by running:**
- **npm install**
- **Checked for any environment variables needed for the API integration (e.g., API keys, database URLs).**
- **Updated the .env file with relevant keys, if required.**

---

3. Integrating the API into My Project

- **Reviewed the cloned repository's API implementation to understand how data was fetched, processed, and displayed.**
- **Copied the necessary files or code snippets (e.g., API calls, configurations, or utility functions) into my project.**
- **Ensured proper integration of the API functions within my existing components.**

---

4. Testing the Integration

- **Tested the API calls in the browser's developer tools (Network tab) to verify successful communication with the API.**
- **Checked that data was being fetched correctly and displayed in the application.**

---

5. Customizing for My Project

- **Adjusted schemas, components, and styling as needed to align the API integration with my project's requirements.**
- **Validated that the API was compatible with my project's existing data flow and structure.**

---

6. Final Review and Cleanup

- **Removed any unused files or code from the cloned repository.**
- **Ensured all the necessary files were committed to my project's version control system.**
- **Tested the entire application to confirm the API integration worked as expected.**

. Lessons Learned

1. **Importance of Planning:** Detailed planning during schema adjustments and API integration minimized potential risks and downtime.
2. **Error Handling is Crucial:** Building robust error-handling mechanisms ensured that the application could gracefully manage unexpected scenarios.
3. **Testing is Key:** Thorough testing at each step ensured data integrity and a seamless user experience.

---

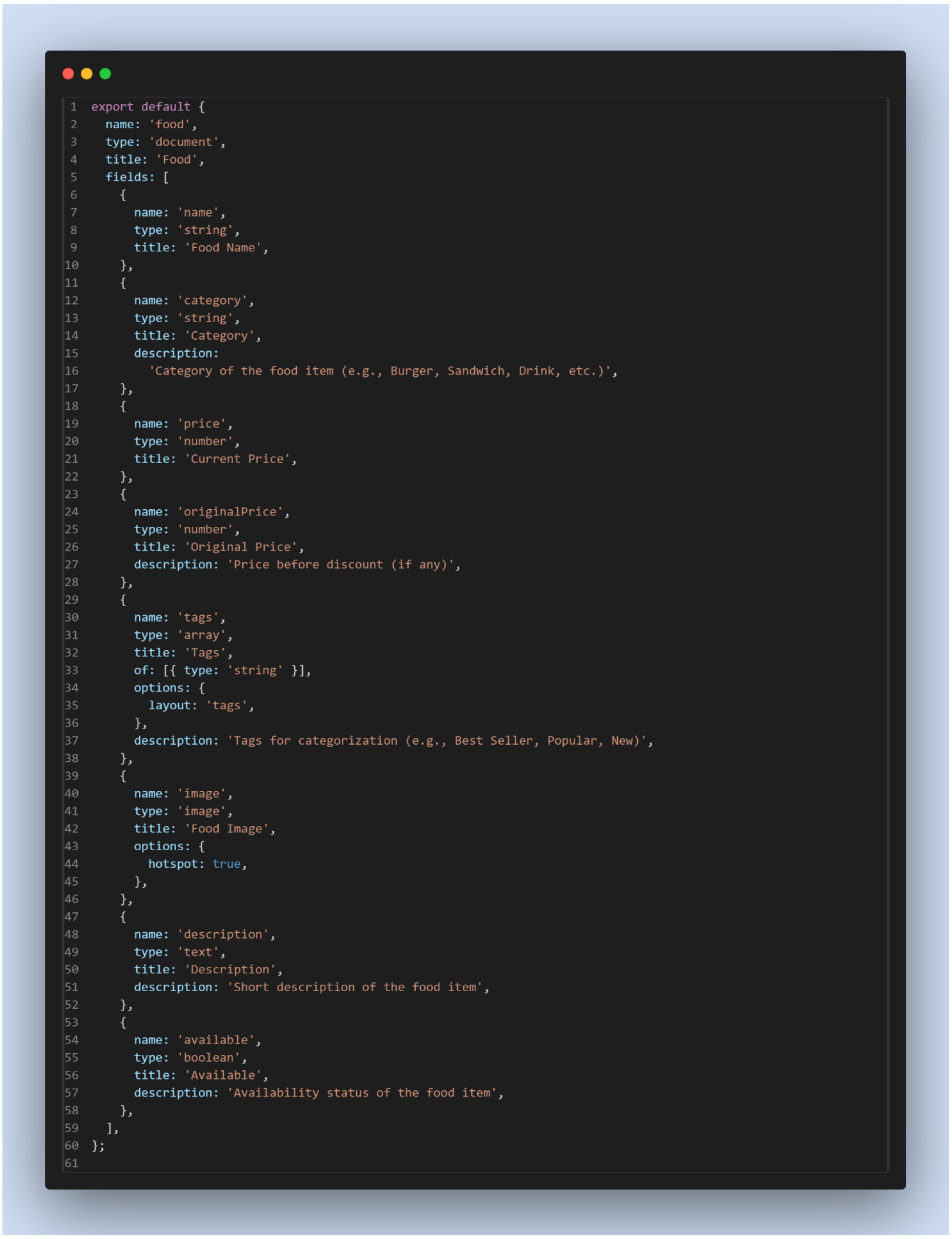
5. Conclusion

The API integration and data migration process was successfully completed with the following outcomes:

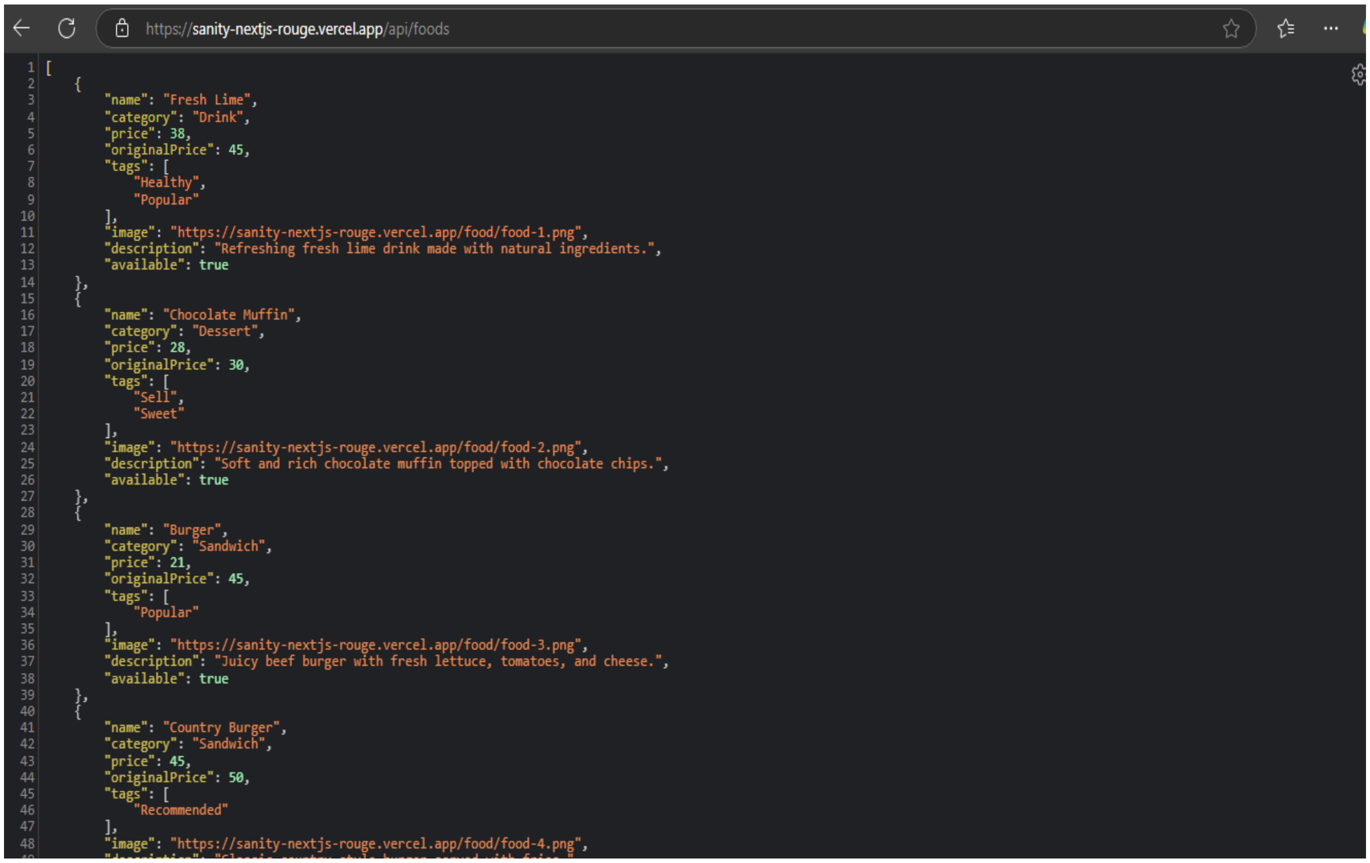
- Secure and efficient API integration to fetch and display dynamic data.
- Enhanced schema to accommodate new fields and extended functionality.
- Successful migration of old data with minimal downtime and improved structure.

Screenshots :

Foods :



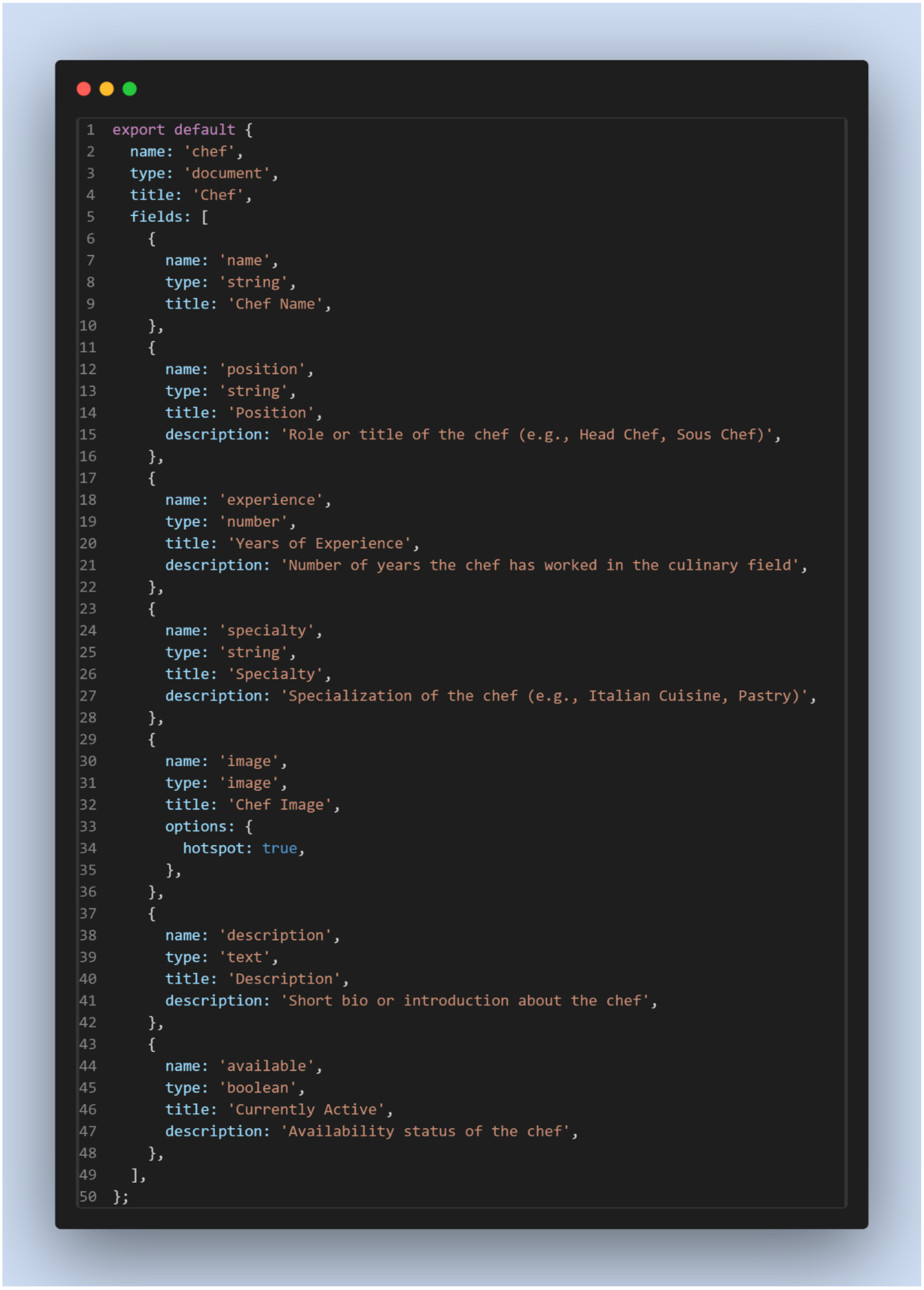
```
export default {
  name: 'food',
  type: 'document',
  title: 'Food',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Food Name',
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
      description:
        'Category of the food item (e.g., 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',
    },
  ],
};
```



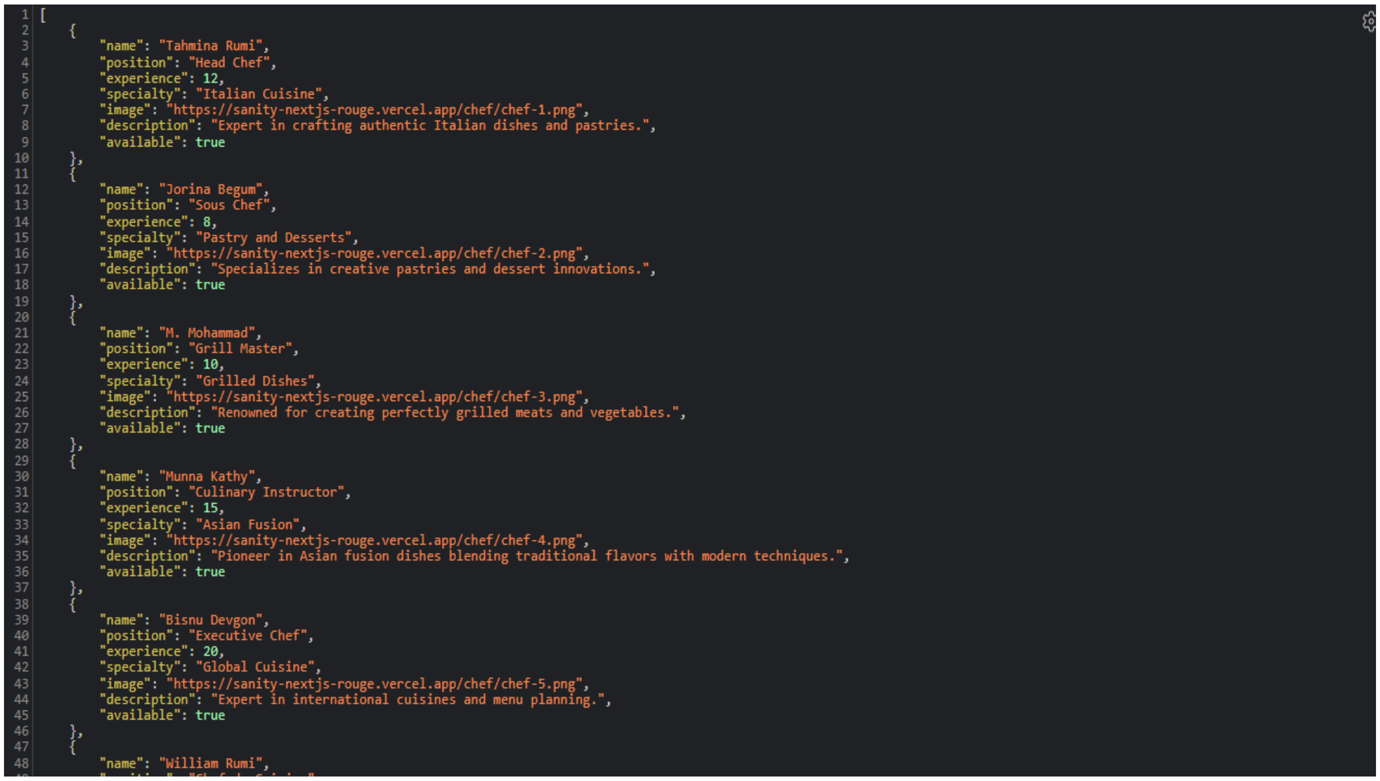
https://sanity-nextjs-rouge.vercel.app/api/foods

```json
[
    {
        "name": "Fresh Lime",
        "category": "Drink",
        "price": 38,
        "originalPrice": 45,
        "tags": [
            "Healthy",
            "Popular"
        ],
        "image": "https://sanity-nextjs-rouge.vercel.app/food/food-1.png",
        "description": "Refreshing fresh lime drink made with natural ingredients.",
        "available": true
    },
    {
        "name": "Chocolate Muffin",
        "category": "Dessert",
        "price": 28,
        "originalPrice": 30,
        "tags": [
            "Sell",
            "Sweet"
        ],
        "image": "https://sanity-nextjs-rouge.vercel.app/food/food-2.png",
        "description": "Soft and rich chocolate muffin topped with chocolate chips.",
        "available": true
    },
    {
        "name": "Burger",
        "category": "Sandwich",
        "price": 21,
        "originalPrice": 45,
        "tags": [
            "Popular"
        ],
        "image": "https://sanity-nextjs-rouge.vercel.app/food/food-3.png",
        "description": "Juicy beef burger with fresh lettuce, tomatoes, and cheese.",
        "available": true
    },
    {
        "name": "Country Burger",
        "category": "Sandwich",
        "price": 45,
        "originalPrice": 50,
        "tags": [
            "Recommended"
        ],
        "image": "https://sanity-nextjs-rouge.vercel.app/food/food-4.png",
        "description": "Classic country-style burger served with fries.",
```

Chefs:

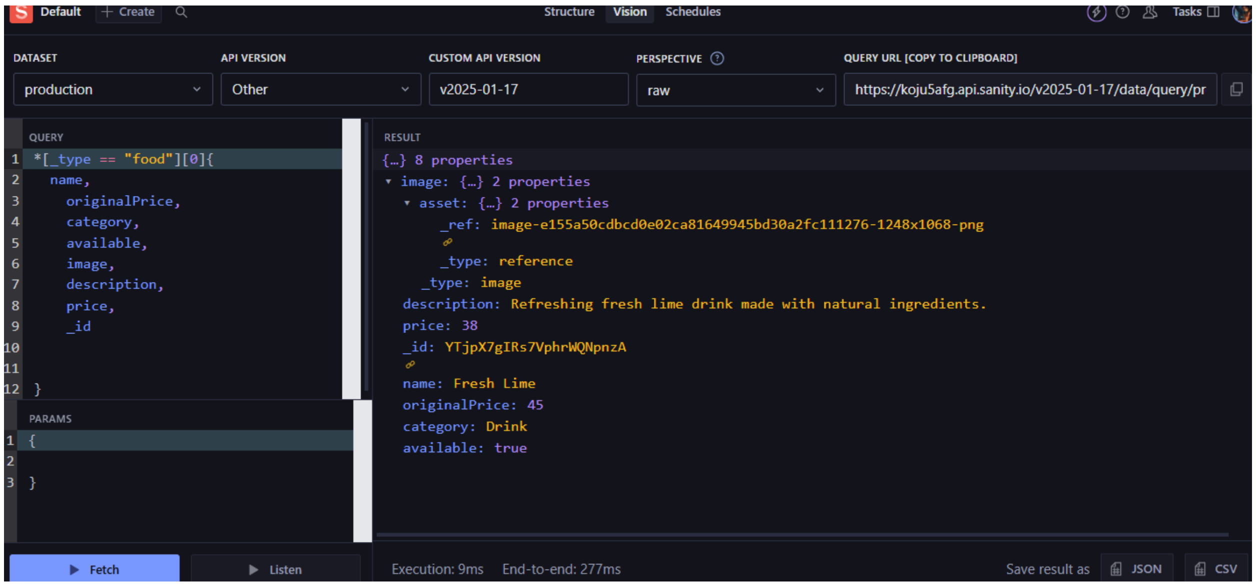


```
export default {
  name: 'chef',
  type: 'document',
  title: 'Chef',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Chef Name',
    },
    {
      name: 'position',
      type: 'string',
      title: 'Position',
      description: 'Role or title of the chef (e.g., Head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



```json
[
    {
        "name": "Tahmina Rumi",
        "position": "Head Chef",
        "experience": 12,
        "specialty": "Italian Cuisine",
        "image": "https://sanity-nextjs-rouge.vercel.app/chef/chef-1.png",
        "description": "Expert in crafting authentic Italian dishes and pastries.",
        "available": true
    },
    {
        "name": "Jorina Begum",
        "position": "Sous Chef",
        "experience": 8,
        "specialty": "Pastry and Desserts",
        "image": "https://sanity-nextjs-rouge.vercel.app/chef/chef-2.png",
        "description": "Specializes in creative pastries and dessert innovations.",
        "available": true
    },
    {
        "name": "M. Mohammad",
        "position": "Grill Master",
        "experience": 10,
        "specialty": "Grilled Dishes",
        "image": "https://sanity-nextjs-rouge.vercel.app/chef/chef-3.png",
        "description": "Renowned for creating perfectly grilled meats and vegetables.",
        "available": true
    },
    {
        "name": "Munna Kathy",
        "position": "Culinary Instructor",
        "experience": 15,
        "specialty": "Asian Fusion",
        "image": "https://sanity-nextjs-rouge.vercel.app/chef/chef-4.png",
        "description": "Pioneer in Asian fusion dishes blending traditional flavors with modern techniques.",
        "available": true
    },
    {
        "name": "Bisnu Devgon",
        "position": "Executive Chef",
        "experience": 20,
        "specialty": "Global Cuisine",
        "image": "https://sanity-nextjs-rouge.vercel.app/chef/chef-5.png",
        "description": "Expert in international cuisines and menu planning.",
        "available": true
    },
    {
        "name": "William Rumi",
```

Fetching Data (Foods) :



DATASET: production
API VERSION: Other
CUSTOM API VERSION: v2025-01-17
PERSPECTIVE: raw
QUERY URL [COPY TO CLIPBOARD]: https://koju5afg.api.sanity.io/v2025-01-17/data/query/pr

QUERY

```
*[_type == "food"][0]{
  name,
    originalPrice,
    category,
    available,
    image,
    description,
    price,
    _id


}
```

PARAMS

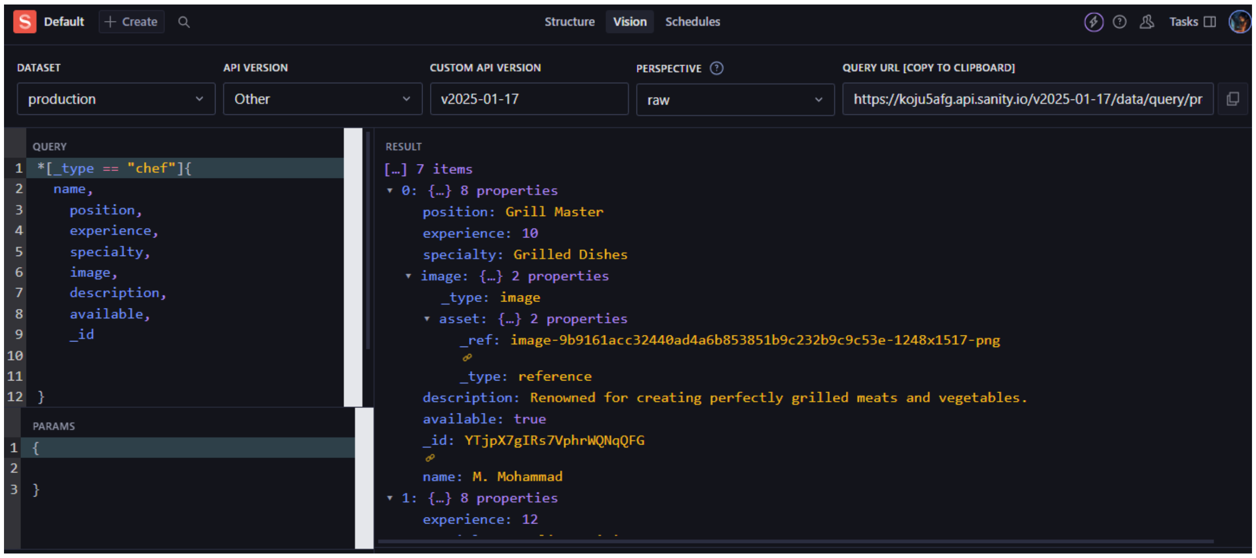
```
{

}
```

RESULT

```
{…} 8 properties
▾ image: {…} 2 properties
  ▾ asset: {…} 2 properties
      _ref: image-e155a50cdbcd0e02ca81649945bd30a2fc111276-1248x1068-png
      _type: reference
    _type: image
  description: Refreshing fresh lime drink made with natural ingredients.
  price: 38
  _id: YTjpX7gIRs7VphrWQNpnzA
  name: Fresh Lime
  originalPrice: 45
  category: Drink
  available: true
```

Execution: 9ms   End-to-end: 277ms   Save result as JSON CSV

Fetching Data (Chefs) :



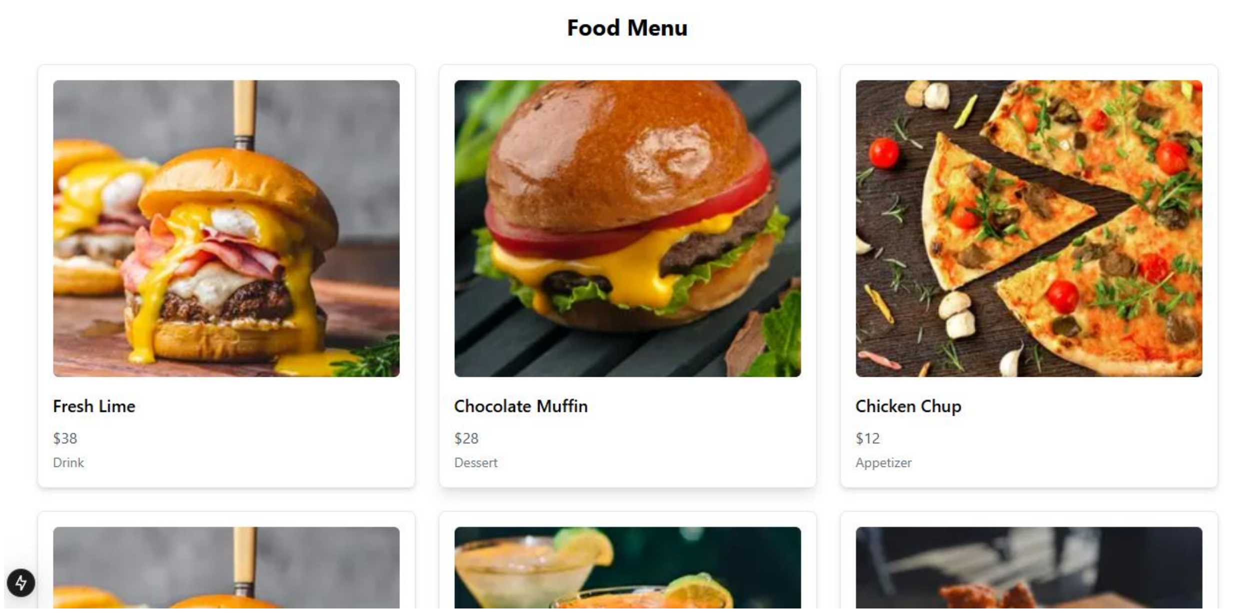
DATASET: production
API VERSION: Other
CUSTOM API VERSION: v2025-01-17
PERSPECTIVE: raw
QUERY URL [COPY TO CLIPBOARD]: https://koju5afg.api.sanity.io/v2025-01-17/data/query/pr

QUERY
```
*[_type == "chef"]{
  name,
    position,
    experience,
    specialty,
    image,
    description,
    available,
    _id


}
```

PARAMS
```
{

}
```

RESULT
```
[…] 7 items
▾ 0: {…} 8 properties
    position: Grill Master
    experience: 10
    specialty: Grilled Dishes
  ▾ image: {…} 2 properties
      _type: image
    ▾ asset: {…} 2 properties
        _ref: image-9b9161acc32440ad4a6b853851b9c232b9c9c53e-1248x1517-png
        _type: reference
    description: Renowned for creating perfectly grilled meats and vegetables.
    available: true
    _id: YTjpX7gIRs7VphrWQNqQFG
    name: M. Mohammad
▾ 1: {…} 8 properties
    experience: 12
```

Data Successfully Displayed in Frontend:

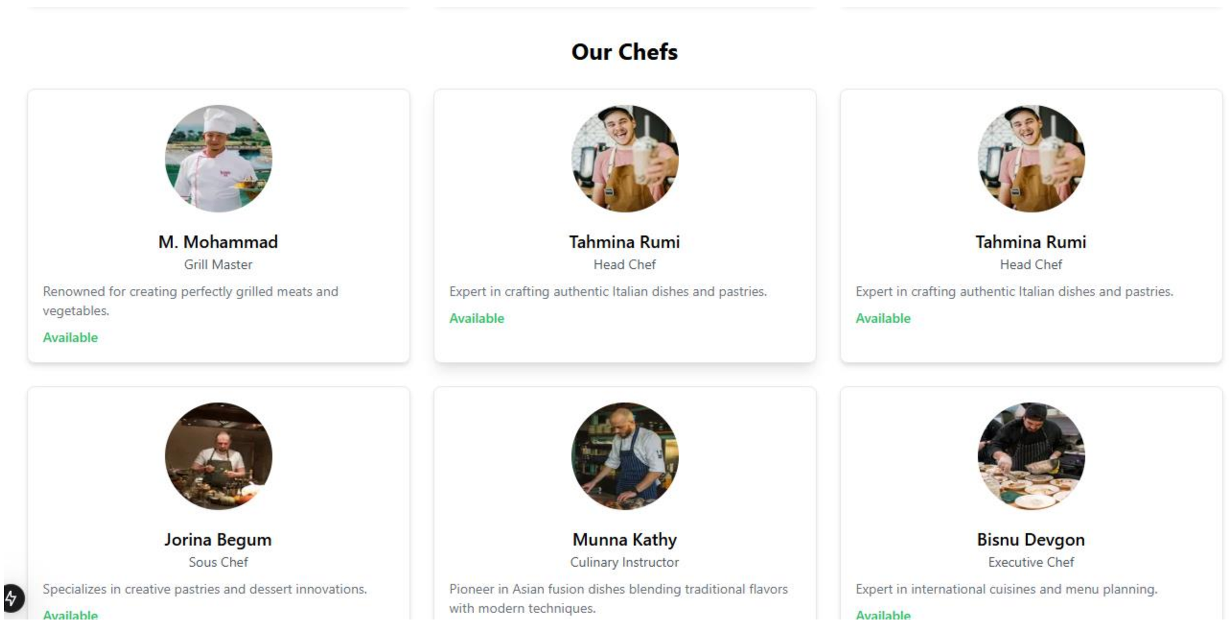


Food Menu

Fresh Lime
$38
Drink

Chocolate Muffin
$28
Dessert

Chicken Chup
$12
Appetizer



Our Chefs

M. Mohammad
Grill Master
Renowned for creating perfectly grilled meats and vegetables.
Available

Tahmina Rumi
Head Chef
Expert in crafting authentic Italian dishes and pastries.
Available

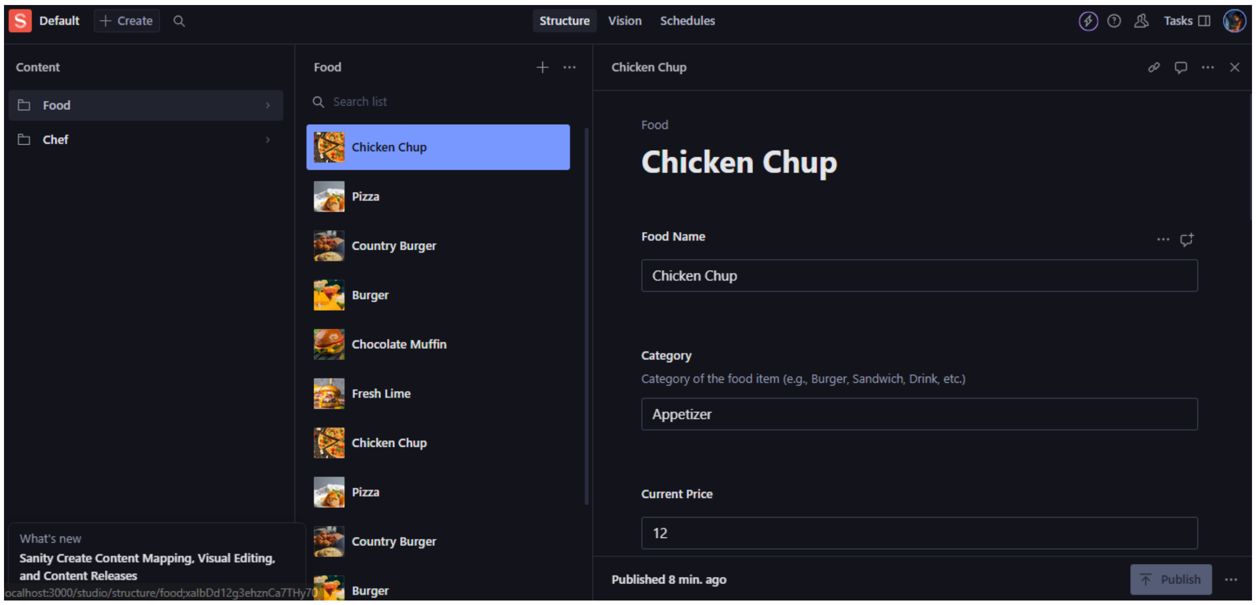
Tahmina Rumi
Head Chef
Expert in crafting authentic Italian dishes and pastries.
Available

Jorina Begum
Sous Chef
Specializes in creative pastries and dessert innovations.
Available

Munna Kathy
Culinary Instructor
Pioneer in Asian fusion dishes blending traditional flavors with modern techniques.

Bisnu Devgon
Executive Chef
Expert in international cuisines and menu planning.
Available

Populated Sanity CMS fields:



Food list:
- Chicken Chup
- Pizza
- Country Burger
- Burger
- Chocolate Muffin
- Fresh Lime
- Chicken Chup
- Pizza
- Country Burger
- Burger

Chicken Chup

Food

Chicken Chup

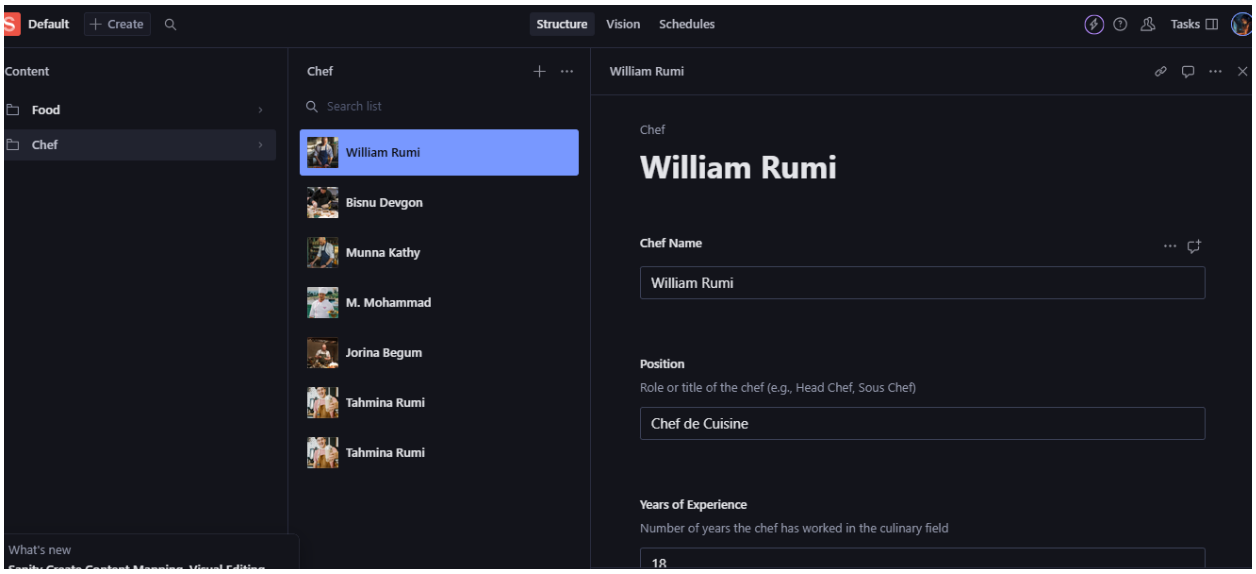
Food Name
Chicken Chup

Category
Category of the food item (e.g., Burger, Sandwich, Drink, etc.)
Appetizer

Current Price
12

Published 8 min. ago

What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases
ocalhost:3000/studio/structure/food;xalbDd12g3ehznCa7THy70



Chef

William Rumi

Chef Name

William Rumi

Position

Role or title of the chef (e.g., Head Chef, Sous Chef)

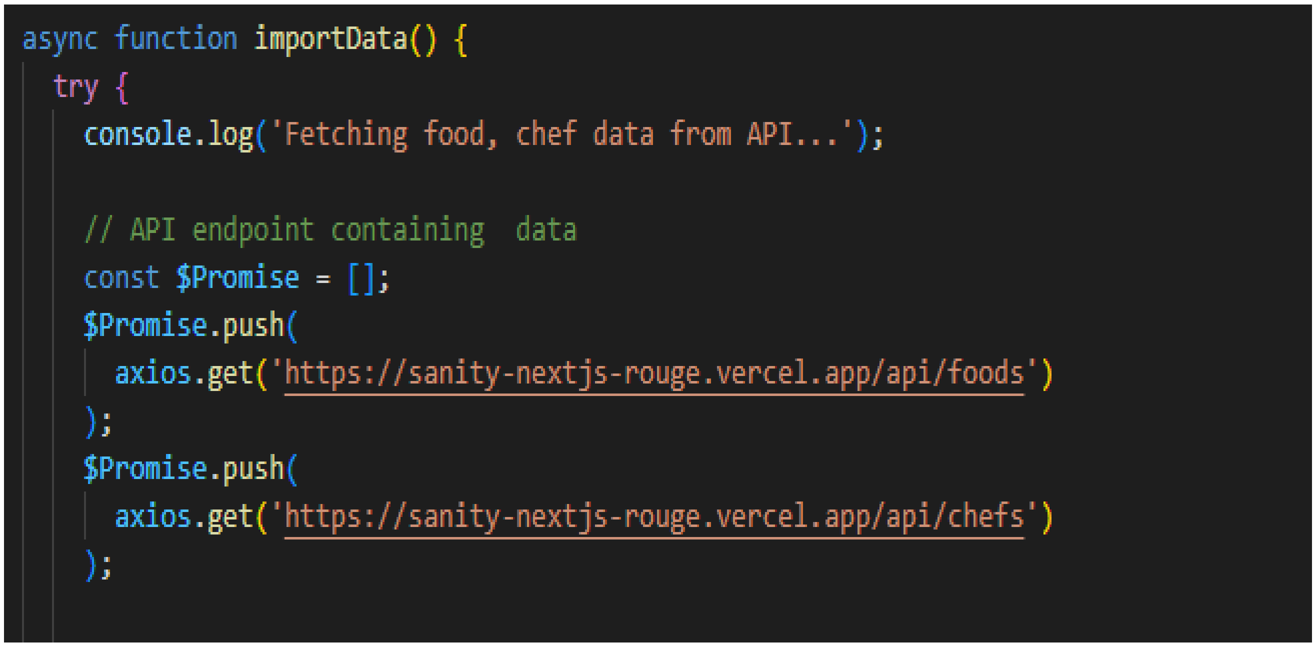
Chef de Cuisine

Years of Experience

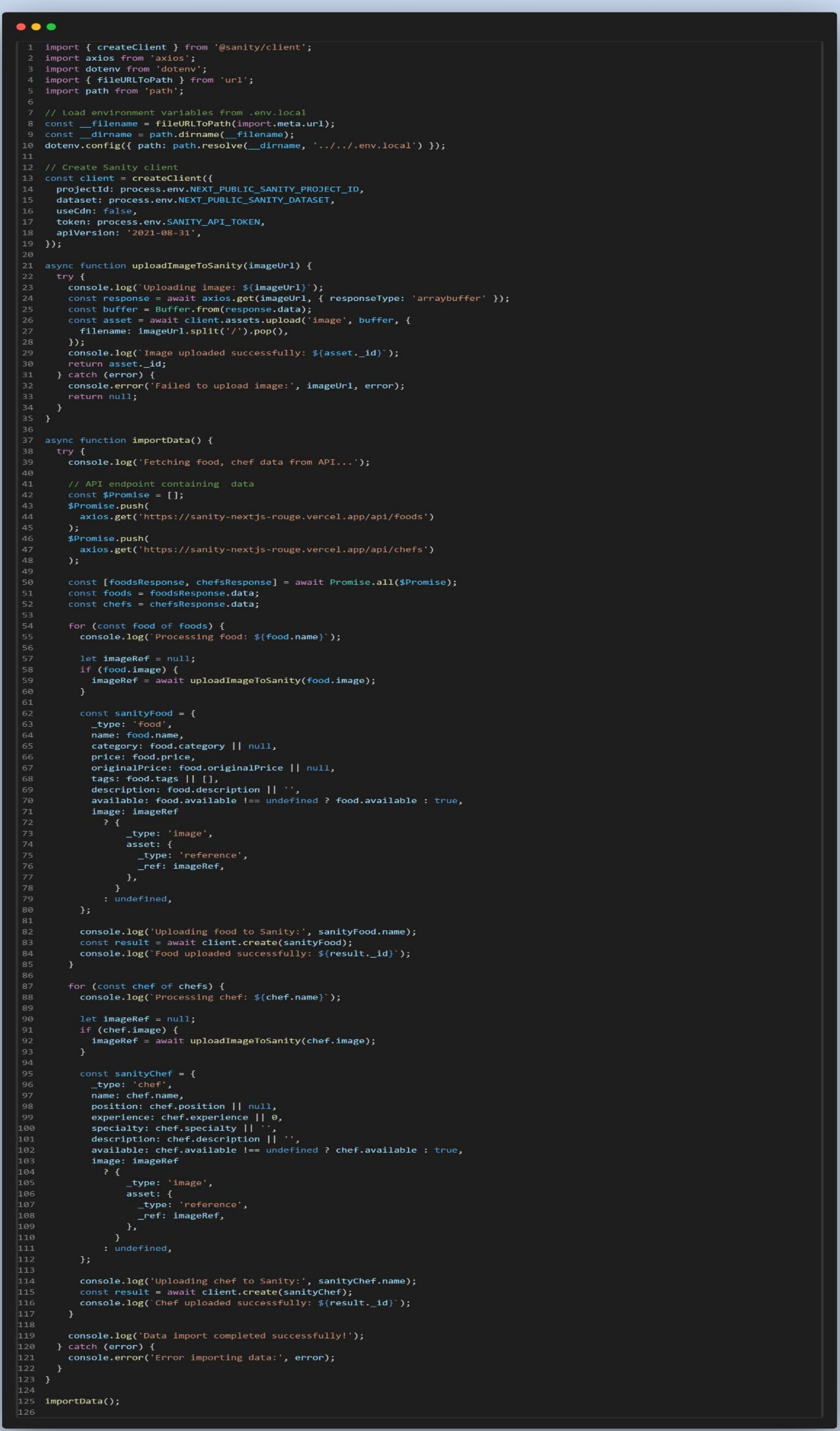
Number of years the chef has worked in the culinary field

18

Code snippets for API integration and migration scripts:



```
async function importData() {
  try {
    console.log('Fetching food, chef data from API...');

    // API endpoint containing  data
    const $Promise = [];
    $Promise.push(
      axios.get('https://sanity-nextjs-rouge.vercel.app/api/foods')
    );
    $Promise.push(
      axios.get('https://sanity-nextjs-rouge.vercel.app/api/chefs')
    );
```



```javascript
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31',
});

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

async function importData() {
  try {
    console.log('Fetching food, chef data from API...');

    // API endpoint containing  data
    const $Promise = [];
    $Promise.push(
      axios.get('https://sanity-nextjs-rouge.vercel.app/api/foods')
    );
    $Promise.push(
      axios.get('https://sanity-nextjs-rouge.vercel.app/api/chefs')
    );

    const [foodsResponse, chefsResponse] = await Promise.all($Promise);
    const foods = foodsResponse.data;
    const chefs = chefsResponse.data;

    for (const food of foods) {
      console.log(`Processing food: ${food.name}`);

      let imageRef = null;
      if (food.image) {
        imageRef = await uploadImageToSanity(food.image);
      }

      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,
      };

      console.log('Uploading food to Sanity:', sanityFood.name);
      const result = await client.create(sanityFood);
      console.log(`Food uploaded successfully: ${result._id}`);
    }

    for (const chef of chefs) {
      console.log(`Processing chef: ${chef.name}`);

      let imageRef = null;
      if (chef.image) {
        imageRef = await uploadImageToSanity(chef.image);
      }

      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,
      };

      console.log('Uploading chef to Sanity:', sanityChef.name);
      const result = await client.create(sanityChef);
      console.log(`Chef uploaded successfully: ${result._id}`);
    }

    console.log('Data import completed successfully!');
  } catch (error) {
    console.error('Error importing data:', error);
  }
}

importData();
```

Self-Validation Checklist:

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

Prepared by: Maria Khan
Date: [18-January-2024]