DAY 3- API INTEGRATION AND DATA MIGRATION

By,
Qirat Saeed,
00160049,
Sunday 9am to 12pm

The document titled "Day 3: API Integration and Data Migration" is part of the Marketplace Builder Hackathon 2025 resources provided by Sir Ameen. In this document I will discuss these topics:

1. **API Integration**: This section probably discusses how to connect and interact with various Application Programming Interfaces (APIs) to enhance the functionality of your marketplace application. It may include best practices for integrating third-party services, handling authentication, and managing data exchange between different systems.
2.
3. **Data Migration**: This part likely focuses on strategies for transferring data from existing systems to your new marketplace platform. It might cover topics such as data mapping, ensuring data integrity during the migration process, and tools or methodologies to facilitate smooth data transitions.

Step no.1 :

Step no.1 is to install and defined the sanity.io by using this command:

```
npm I sanity next-sanity @sanity/vision
```

Before using this command make sure your a Next.JS Configuration is done✓

After this you have create/defined the sanity configuration

Step no.2 :

Create a script file for defined the configuration of how to import data from **third party api / fake api**

With **Sanity.io :**

```
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();
```

Step no.3:

After this run this command:

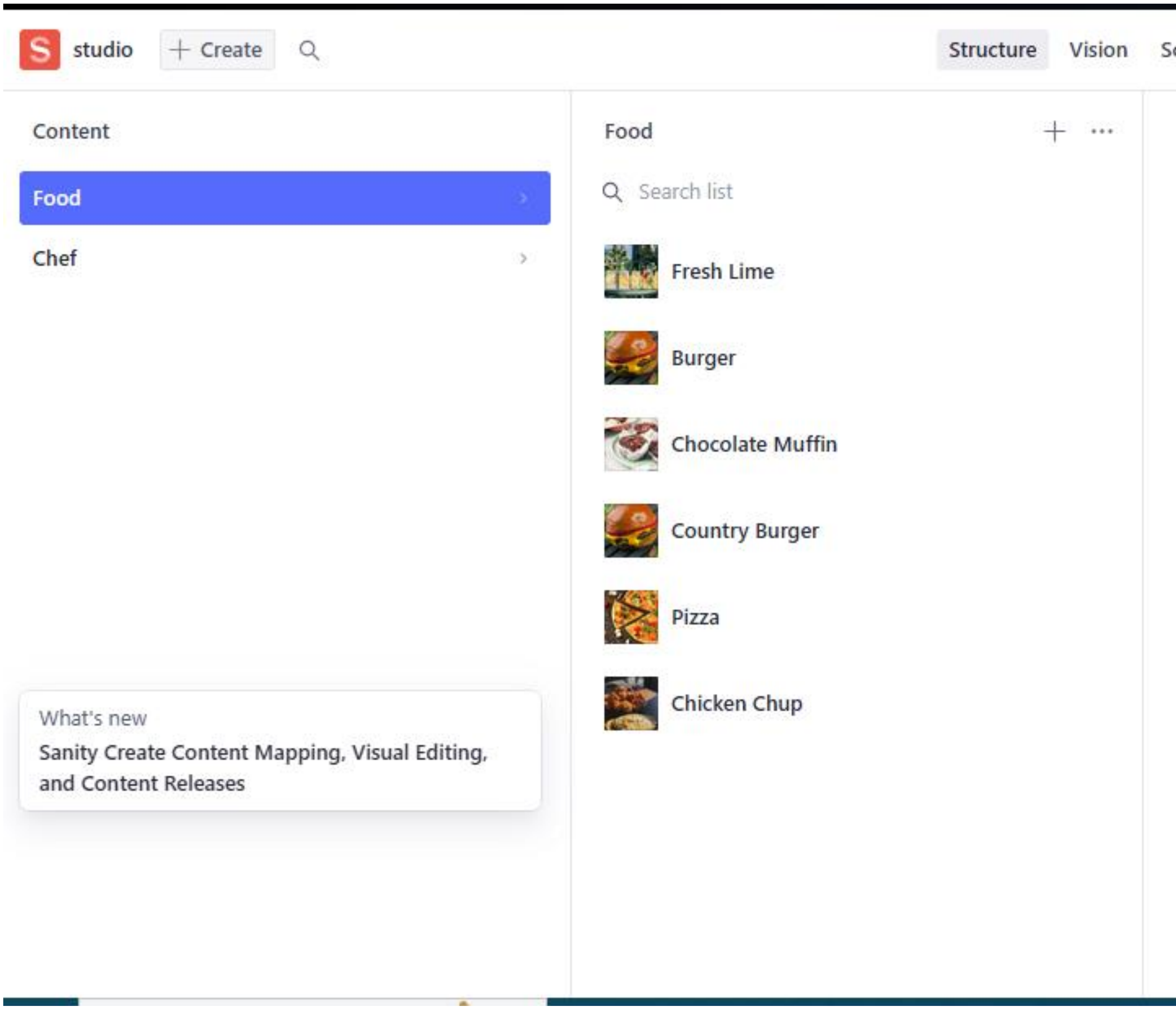
```
npm run import-data
```

Before this make you give the path in package.json

Step no.4:

Now open the Sanity Studio Locally to show the fetched data.

Here is food item data :



Fresh Lime

Burger

Chocolate Muffin

Country Burger

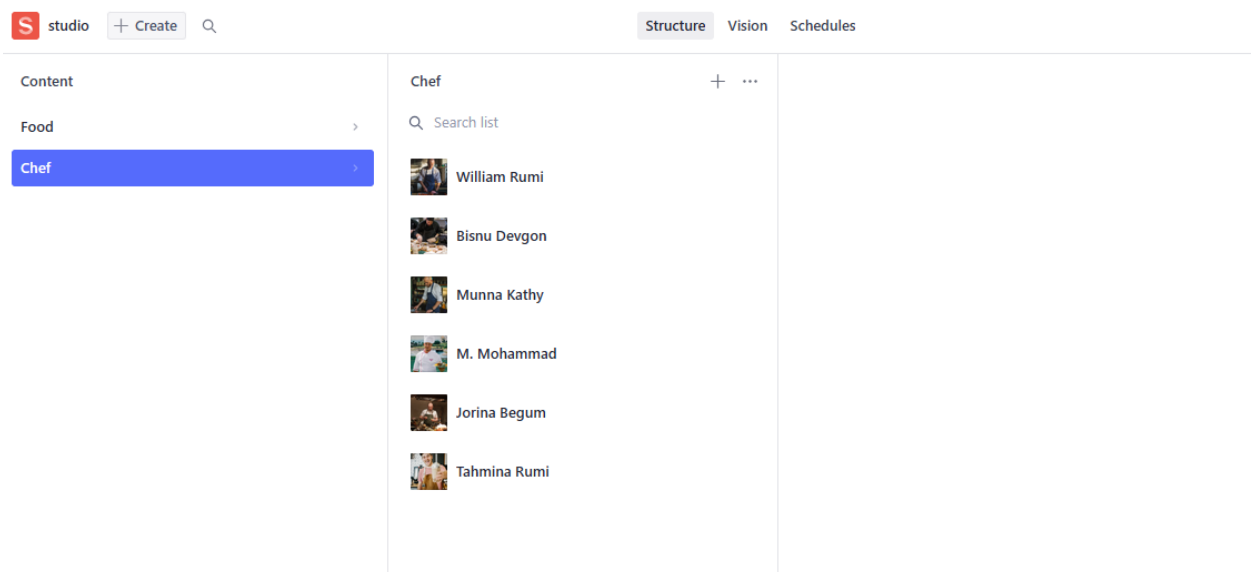
Pizza

Chicken Chup

What's new

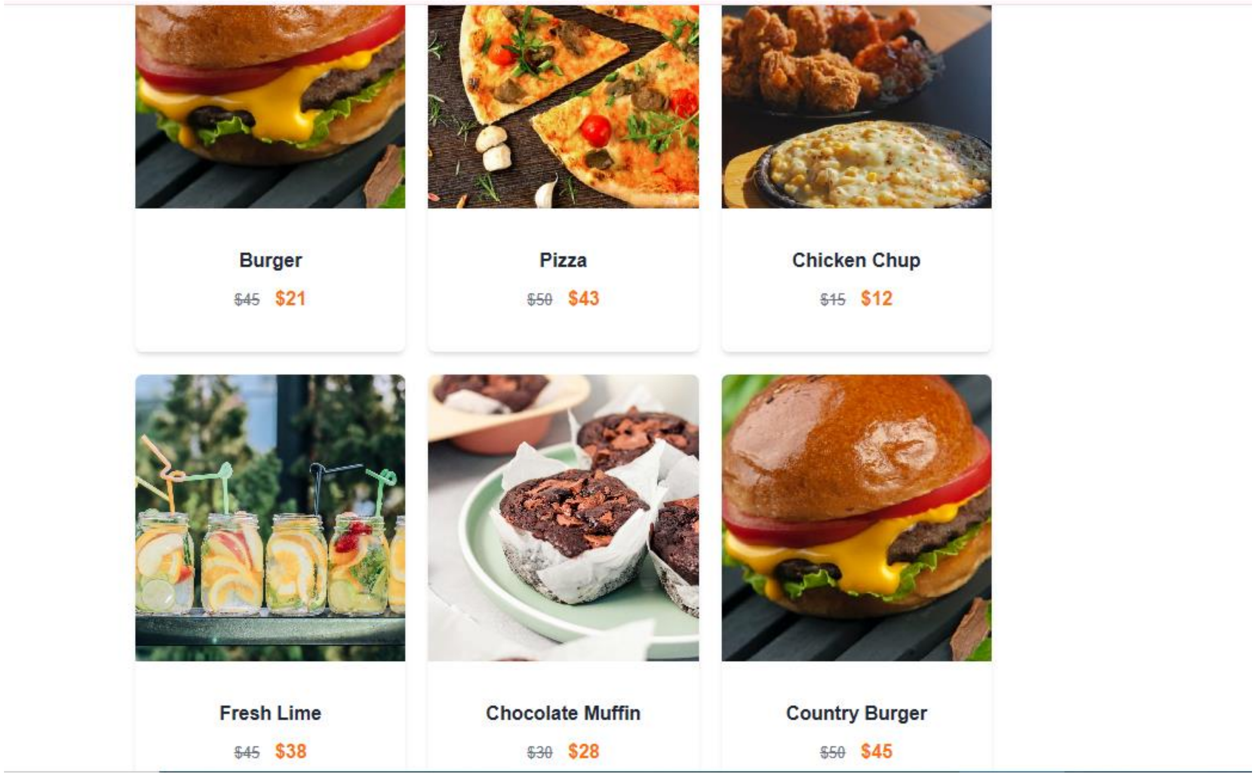
Sanity Create Content Mapping, Visual Editing, and Content Releases

Here is chef data which is fetched from api:



William Rumi
Bisnu Devgon
Munna Kathy
M. Mohammad
Jorina Begum
Tahmina Rumi

Data successfully displayed in the frontend:



Burger
~~$45~~ $21

Pizza
~~$50~~ $43

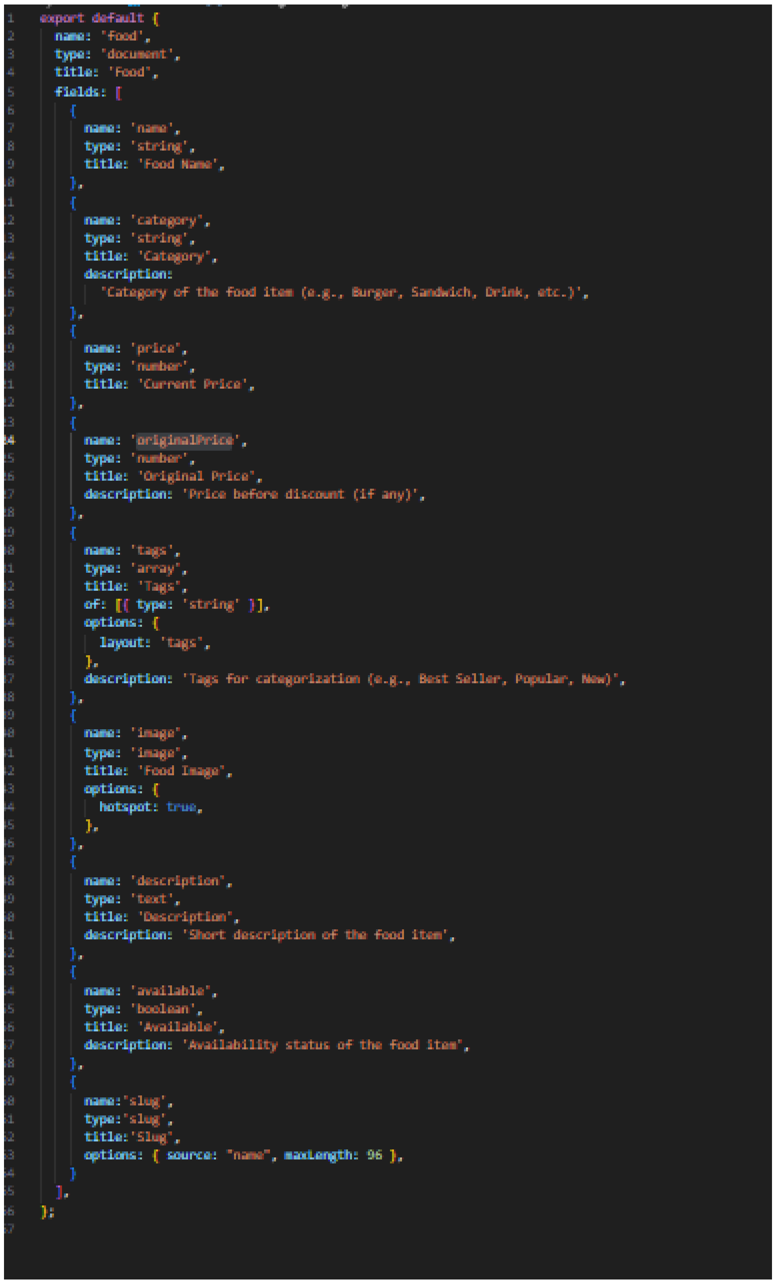
Chicken Chup
~~$15~~ $12

Fresh Lime
~~$45~~ $38

Chocolate Muffin
~~$30~~ $28

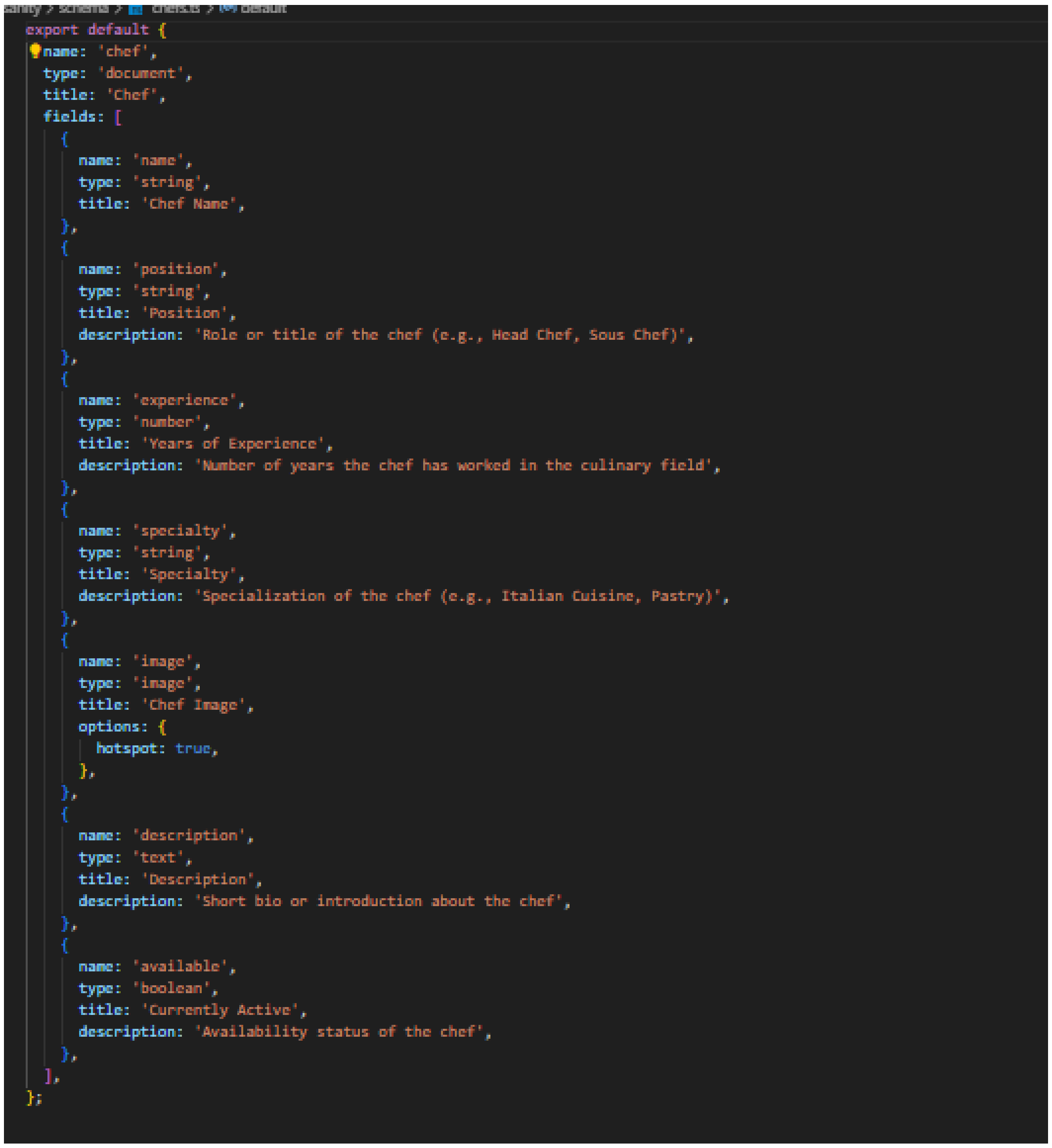
Country Burger
~~$50~~ $45

Schema of food:



```
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',
    },
    {
      name:'slug',
      type:'slug',
      title:'Slug',
      options: { source: "name", maxLength: 96 },
    }
  ],
};
```

Schema of Chefs:



```
anity > schema > chefs.js > default
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',
    },
  ],
};
```

Day 3 Checklist: Self-Validation Checklist: API Understanding: • ✓

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