Ismat Fatima (00373389)

Day 3 - API Integration Report – QuickBite

- API Integration Process:

1. **Sanity Project Setup:**

   - Created a Sanity project for content management.
   - Generated an API token in the project settings for secure access.
   - Configured .env.local file in Next.js project to store the API token and project ID securely.

2. **Data Migration:**

   - Cloned the GitHub repository containing migration scripts and schemas for your API.
   - Modified or validated the schemas (e.g., added images array and stock fields in the foods schema) to align with data structure.
   - Logged in to the Sanity Studio to access and verify schema definitions.
   - Imported API data using the provided migration script in the given repository. Successfully migrated the Food and Chef data to Sanity CMS.

3. **Fetching Data Using GROQ Queries:**

   - Created GROQ query functions to fetch data directly from Sanity. Queries included fetching all categories, categories along with food items, and single food item details.
   - Utilized these queries in the components of my Next.js website.

4. **Frontend Data Integration:**

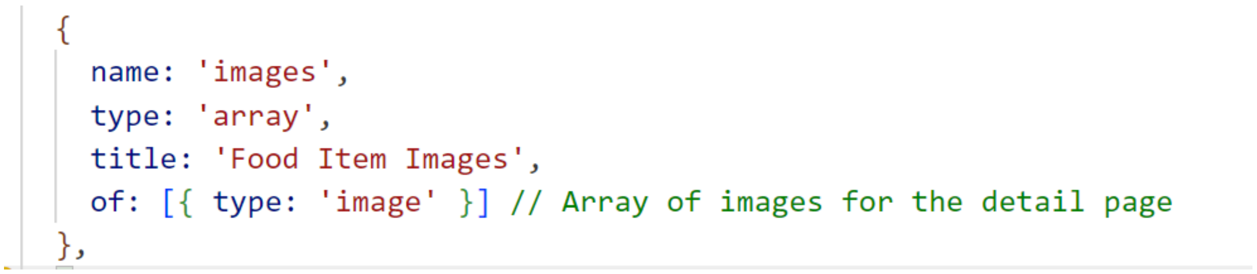
- Integrated the fetched data into the frontend components.
- Used the map function to display the data dynamically in HTML, such as showing lists of food items or categories.
- Ensured the data was rendered correctly by testing API calls and visualizing the data in the browser.

5. **Testing and Validation:**

- Verified the integration by ensuring the frontend displayed the data as intended.
- Populated Sanity CMS fields were cross-checked to confirm the data migration's accuracy.

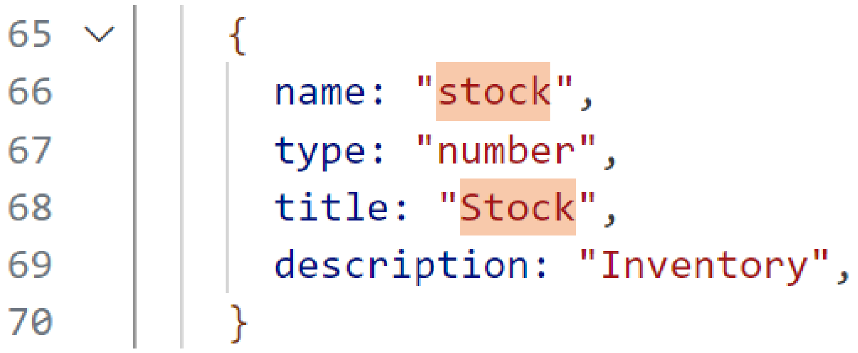
- Adjustment made to schemas:

1) **Added images array field in foods schema:**



```
{
  name: 'images',
  type: 'array',
  title: 'Food Item Images',
  of: [{ type: 'image' }] // Array of images for the detail page
},
```

2) **Added stock field for inventory management in foods schema:**

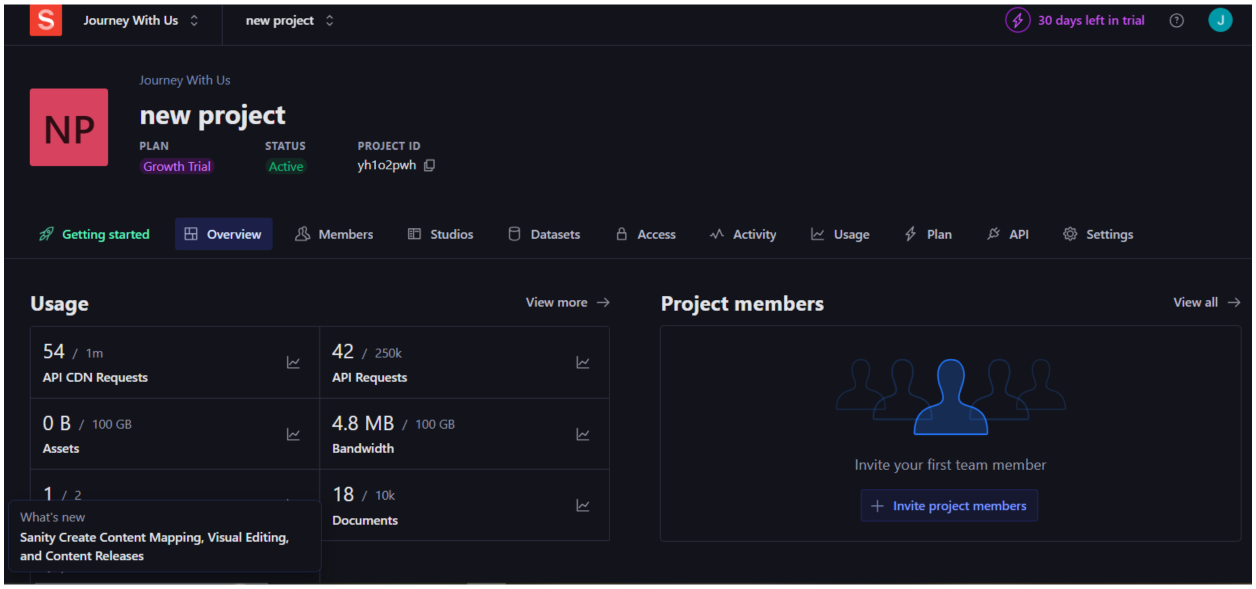


```
{
  name: "stock",
  type: "number",
  title: "Stock",
  description: "Inventory",
}
```

Ismat Fatima (00373389)

Migration steps and tools used:

Setup sanity project in the frontend Next JS website:


Journey With Us | new project | 30 days left in trial

Journey With Us

NP **new project**

PLAN	STATUS	PROJECT ID
Growth Trial	Active	yh1o2pwh

Getting started | Overview | Members | Studios | Datasets | Access | Activity | Usage | Plan | API | Settings

Usage View more →

54 / 1m API CDN Requests	42 / 250k API Requests
0 B / 100 GB Assets	4.8 MB / 100 GB Bandwidth
1 / 2	18 / 10k Documents

What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases

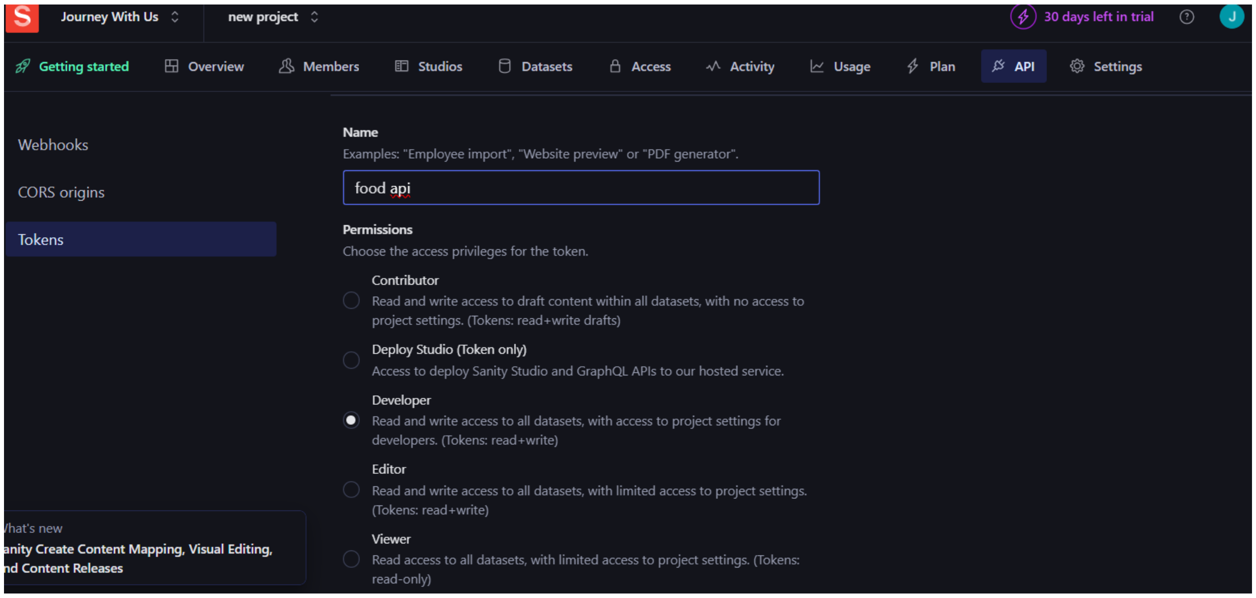
Project members View all →

Invite your first team member

+ Invite project members

Sanity Project created.

Generate editor access API token in project settings:


Journey With Us | new project | 30 days left in trial

Getting started | Overview | Members | Studios | Datasets | Access | Activity | Usage | Plan | API | Settings

Webhooks
CORS origins
Tokens

Name
Examples: "Employee import", "Website preview" or "PDF generator".

food api

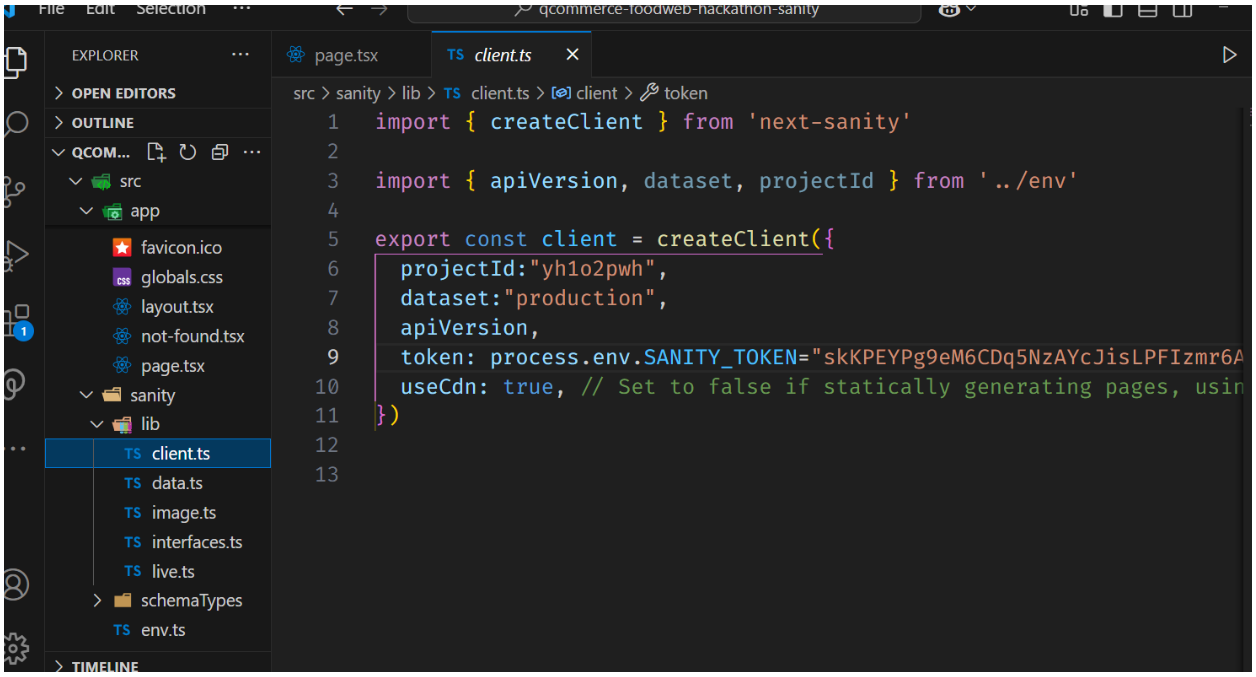
Permissions
Choose the access privileges for the token.

- ○ Contributor — Read and write access to draft content within all datasets, with no access to project settings. (Tokens: read+write drafts)
- ○ Deploy Studio (Token only) — Access to deploy Sanity Studio and GraphQL APIs to our hosted service.
- ● Developer — Read and write access to all datasets, with access to project settings for developers. (Tokens: read+write)
- ○ Editor — Read and write access to all datasets, with limited access to project settings. (Tokens: read+write)
- ○ Viewer — Read access to all datasets, with limited access to project settings. (Tokens: read-only)

hat's new
anity Create Content Mapping, Visual Editing,
nd Content Releases

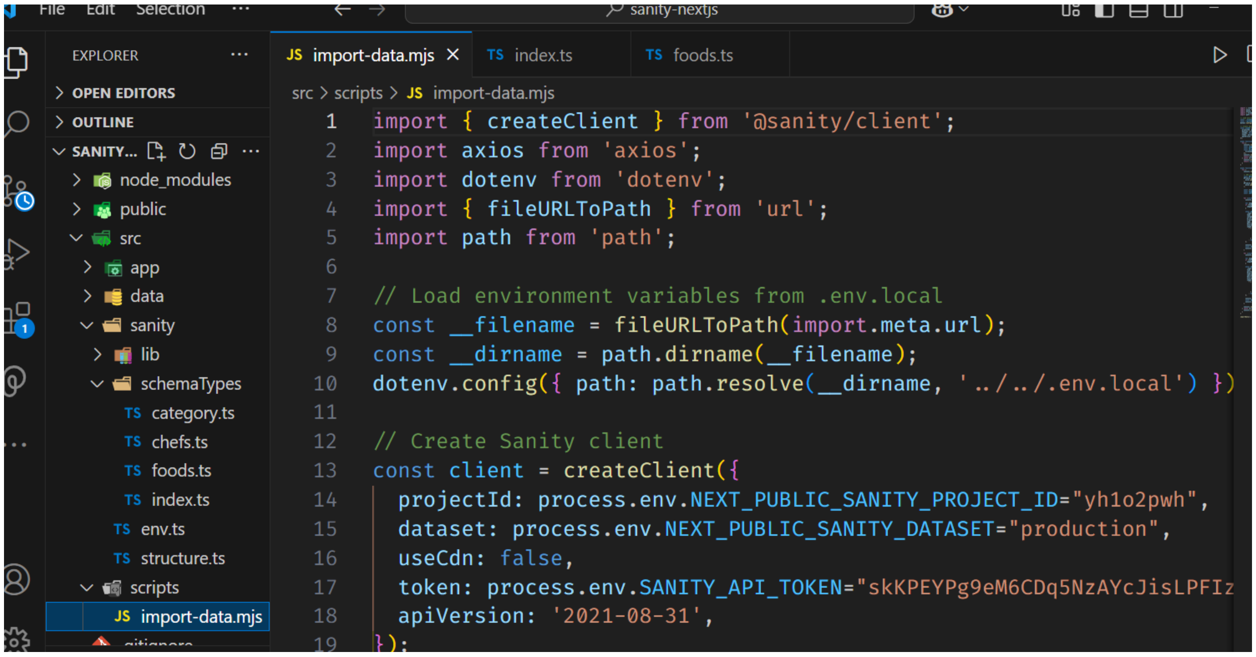
Ismat Fatima (00373389)

Copy and paste the API token in .env.local file and pass that token in client.ts



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

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

export const client = createClient({
  projectId:"yh1o2pwh",
  dataset:"production",
  apiVersion,
  token: process.env.SANITY_TOKEN="skKPEYPg9eM6CDq5NzAYcJisLPFIzmr6A
  useCdn: true, // Set to false if statically generating pages, usin
})
```

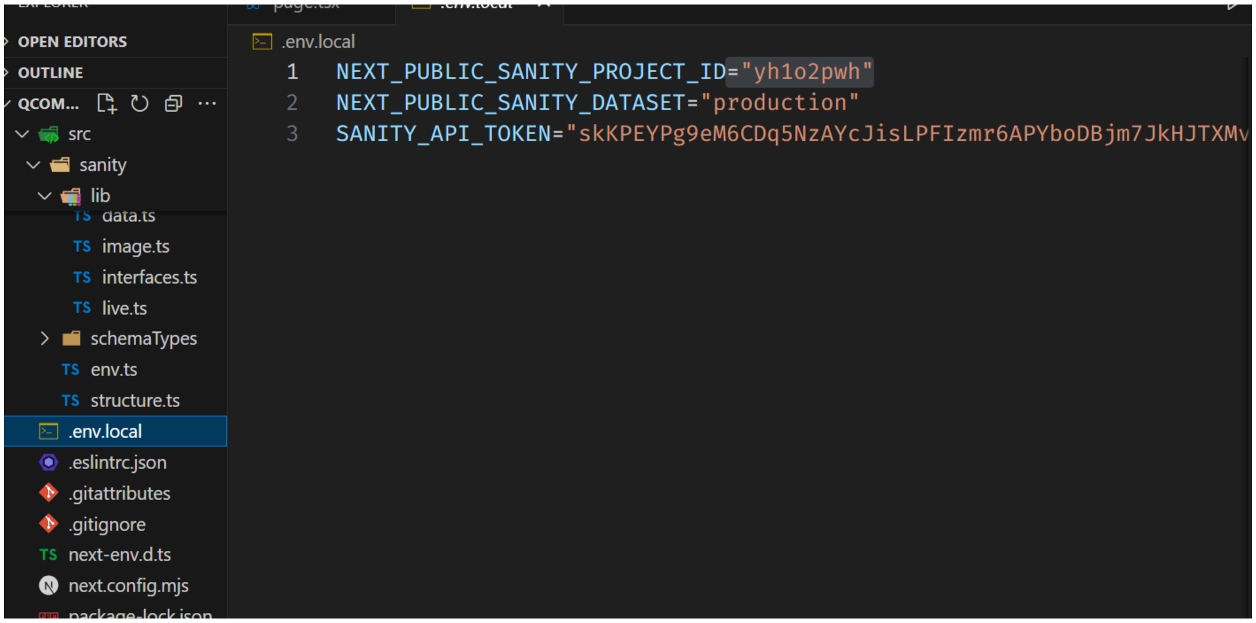
1) **Now, clone the sir Mubashir GitHub repo which contains the migration script for API's and schemas.**



```
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="yh1o2pwh",
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET="production",
  useCdn: false,
  token: process.env.SANITY_API_TOKEN="skKPEYPg9eM6CDq5NzAYcJisLPFIz
  apiVersion: '2021-08-31',
});
```

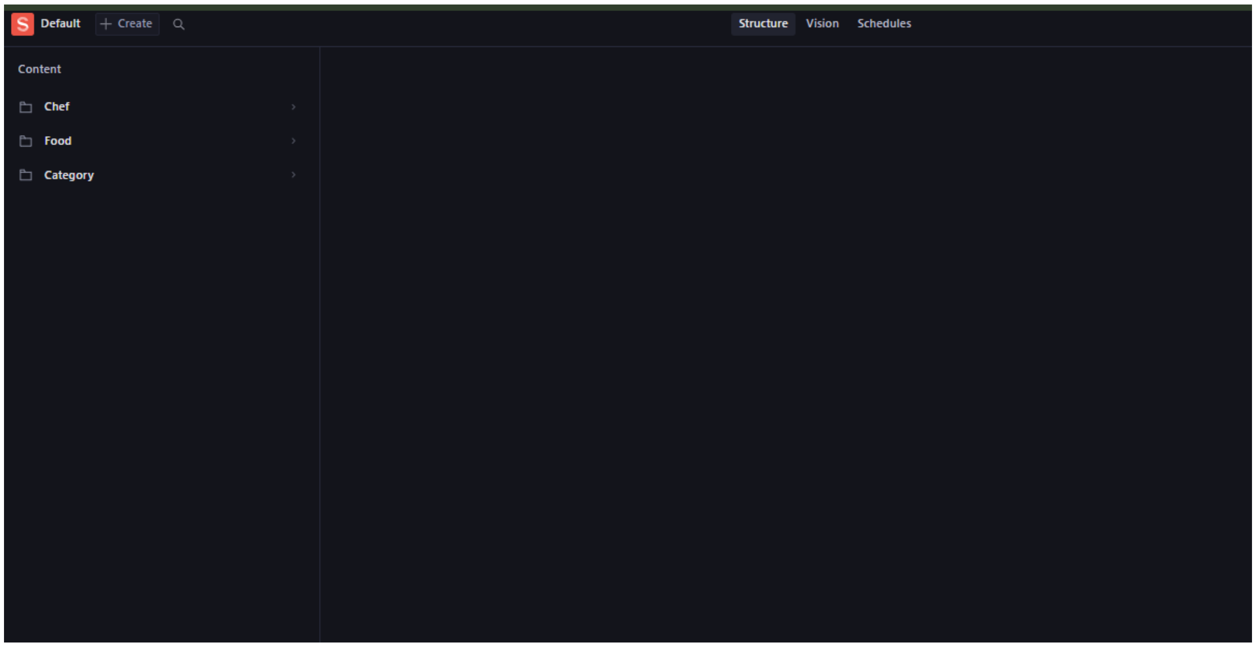
Ismat Fatima (00373389)

Now create .env.local file and set project id and generated api token in .env.local file.



```
NEXT_PUBLIC_SANITY_PROJECT_ID="yh1o2pwh"
NEXT_PUBLIC_SANITY_DATASET="production"
SANITY_API_TOKEN="skKPEYPg9eM6CDq5NzAYcJisLPFIzmr6APYboDBjm7JkHJTXMv
```

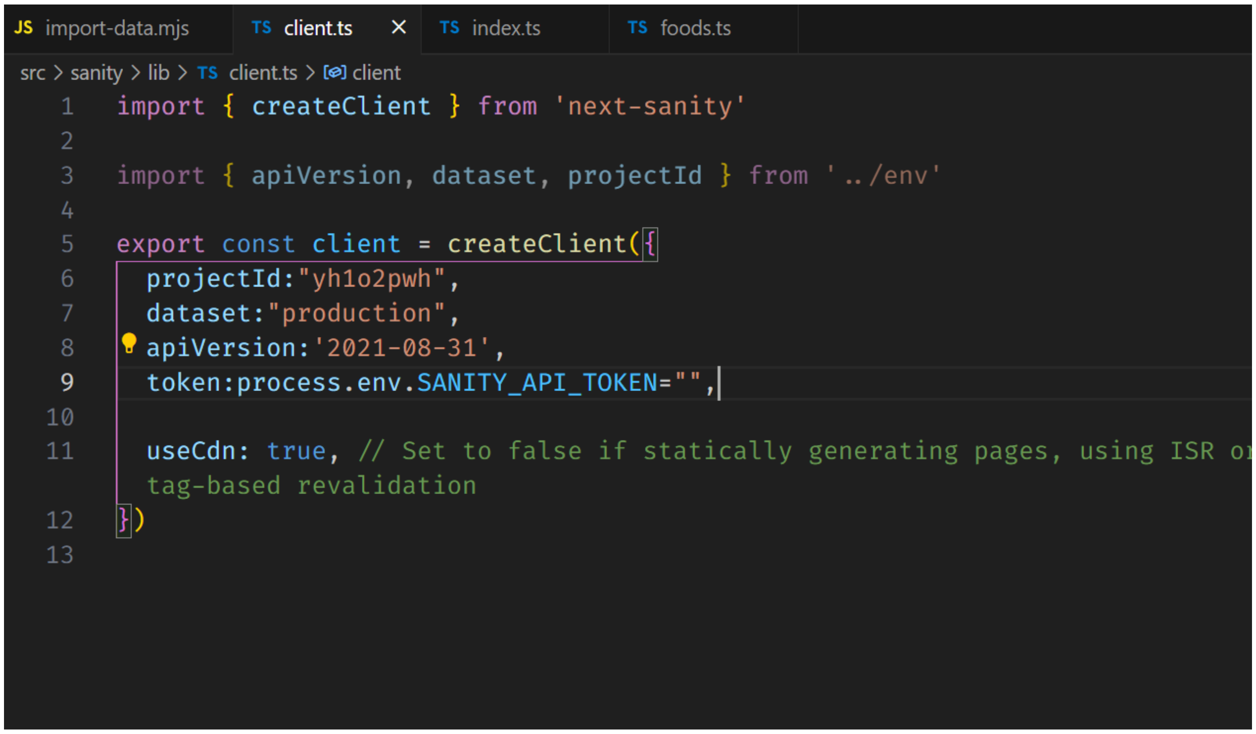
Login to sanity account in /studio:



You can see the defined schemas here.

Ismat Fatima (00373389)

Pass token inside client function in client.ts file.

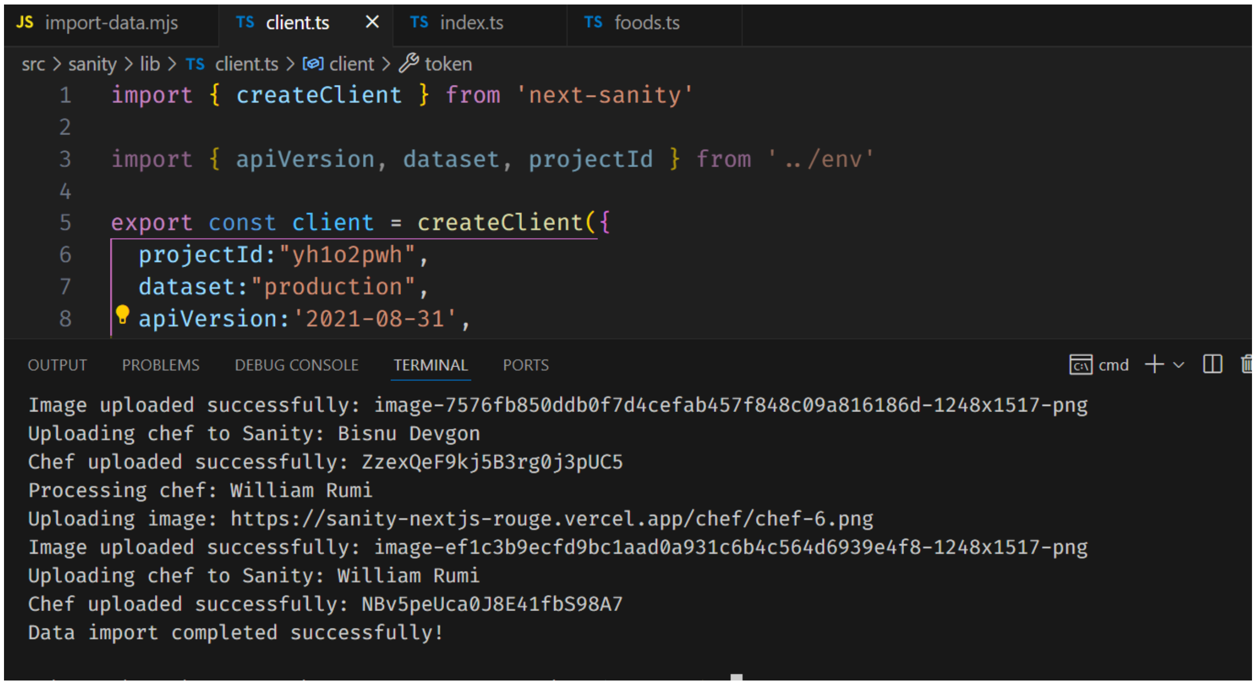


import-data.mjs | client.ts | index.ts | foods.ts

src > sanity > lib > client.ts > client

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

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

export const client = createClient({
  projectId:"yh1o2pwh",
  dataset:"production",
  apiVersion:'2021-08-31',
  token:process.env.SANITY_API_TOKEN="",

  useCdn: true, // Set to false if statically generating pages, using ISR or
  tag-based revalidation
})
```

1) **Now, importing the data using script provided inside script folder.**

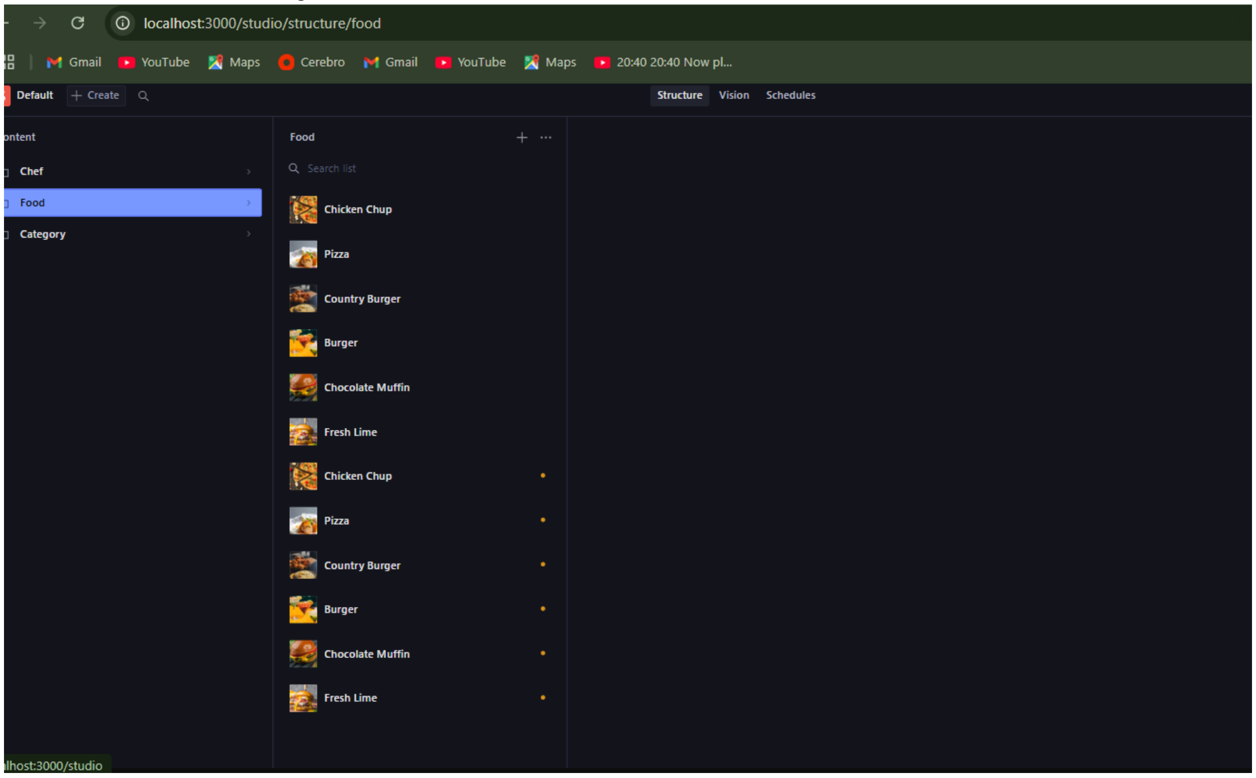


```
import { createClient } from 'next-sanity'

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

export const client = createClient({
  projectId:"yh1o2pwh",
  dataset:"production",
  apiVersion:'2021-08-31',
```

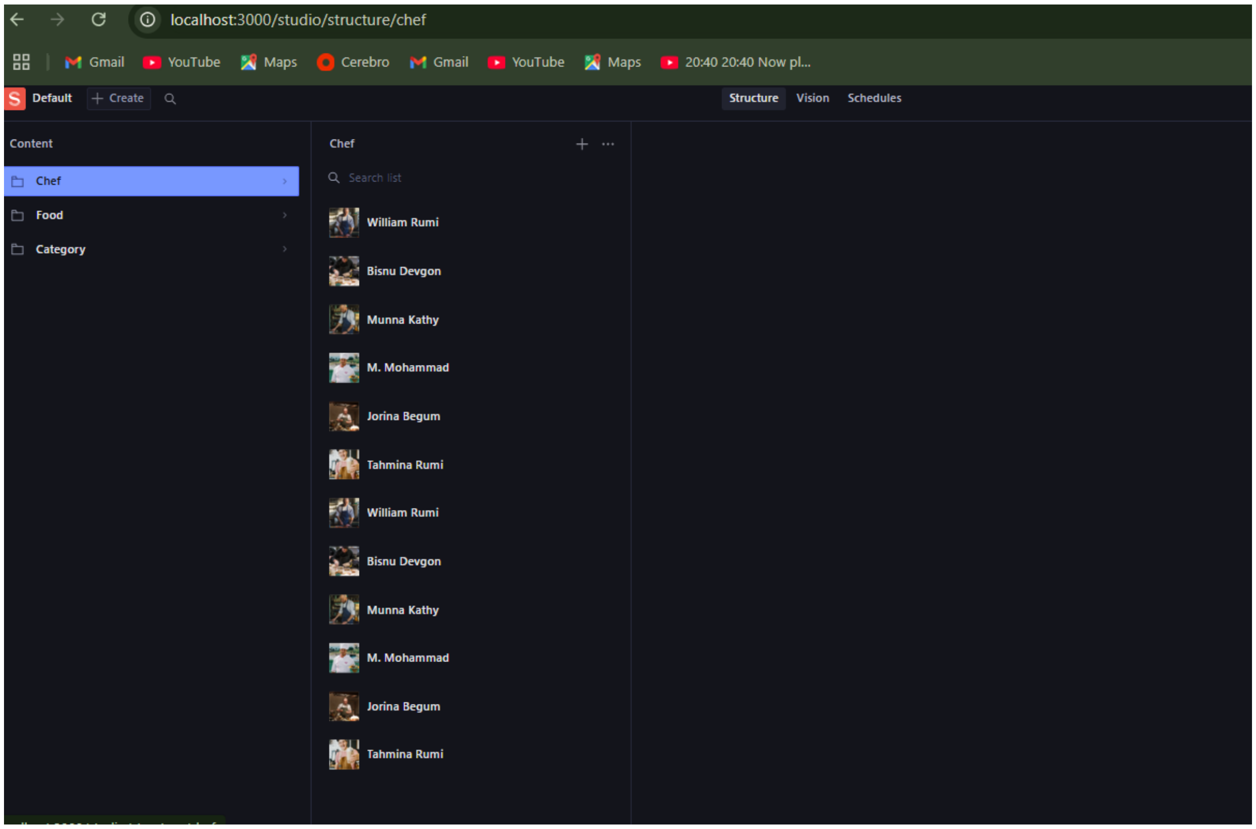
```
Image uploaded successfully: image-7576fb850ddb0f7d4cefab457f848c09a816186d-1248x1517-png
Uploading chef to Sanity: Bisnu Devgon
Chef uploaded successfully: ZzexQeF9kj5B3rg0j3pUC5
Processing chef: William Rumi
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-6.png
Image uploaded successfully: image-ef1c3b9ecfd9bc1aad0a931c6b4c564d6939e4f8-1248x1517-png
Uploading chef to Sanity: William Rumi
Chef uploaded successfully: NBv5peUca0J8E41fbS98A7
Data import completed successfully!
```

Data imported successfully using the script.

Food Data in sanity:



Chefs data in sanity:



Migration Script (import-data.mjs file):

```
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();
```

- API Integration Code Snippets:

Food Category Component API Integration Code (Home page):

```
"use client";
import { Great_Vibes } from "@next/font/google";
import { useEffect, useState } from "react";
import { getAllCategories } from "@/sanity/lib/data";
import { ICategory } from "@/sanity/lib/interfaces";

const greatVibes = Great_Vibes({
  weight: ["400"],
  subsets: ["latin"],
});

const FoodCategory = () => {
  const [categories, setCategories] = useState<ICategory[]>([]);

  useEffect(() => {
    const fetchCategories = async () => {
      try {
        const data: ICategory[] = await getAllCategories();
        setCategories(data);
      } catch (error) {
        console.error("Error fetching categories:", error);
      }
    };

    fetchCategories();
  }, []);

  return (
    <>
      <div className="sec3 px-[20px] py-[60px] sm:px-[60px] text-white max-w-[1320px] relative lg:h-
[600px] mx-auto flex flex-col">
        <div className="flex flex-col items-center">
          <h3
            className={`${greatVibes.className} text-[#FF9F0D] text-[32px] font-bold`}
          >
            Food Category
          </h3>
          <h1
            style={{ fontFamily: "Helvetica, Arial, sans-serif" }}
            className="text-[##FF9F0D] text-[48px] text-center"
          >
            <span>Ch</span>oose Food Item
          </h1>
        </div>

        <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-[40px] justify-items-
center">
          {categories.slice(0, 4).map((category) => (
            <div
              key={category._id}
              className="max-w-[300px]  relative group cursor-pointer"
            >
              <img
                src={category.imageUrl}
                className="w-[100%] h-[100%] rounded-[6px] object-center object-cover"
                alt={category.name}
              />
              <div className="absolute inset-0 flex justify-center items-center bg-black bg-opacity-50 opacity-0
group-hover:opacity-100 transition-opacity duration-300">
                <div className="text-white text-center flex flex-col gap-[5px]">
                  <div className="rounded-[6px] bg-white px-4 py-3 text-[#FF9F0D] w-fit font-bold text-[18px]">
                  Save 30%
                  </div>
                  <div className="rounded-[6px] bg-[#FF9F0D] text-white px-4 py-3 w-[250px] text-[20px] font-
bold">
                    {category.name}
                  </div>
                </div>
              </div>
            </div>
          ))}
        </div>
      </div>
    </>
  );
}
export default FoodCategory;
```

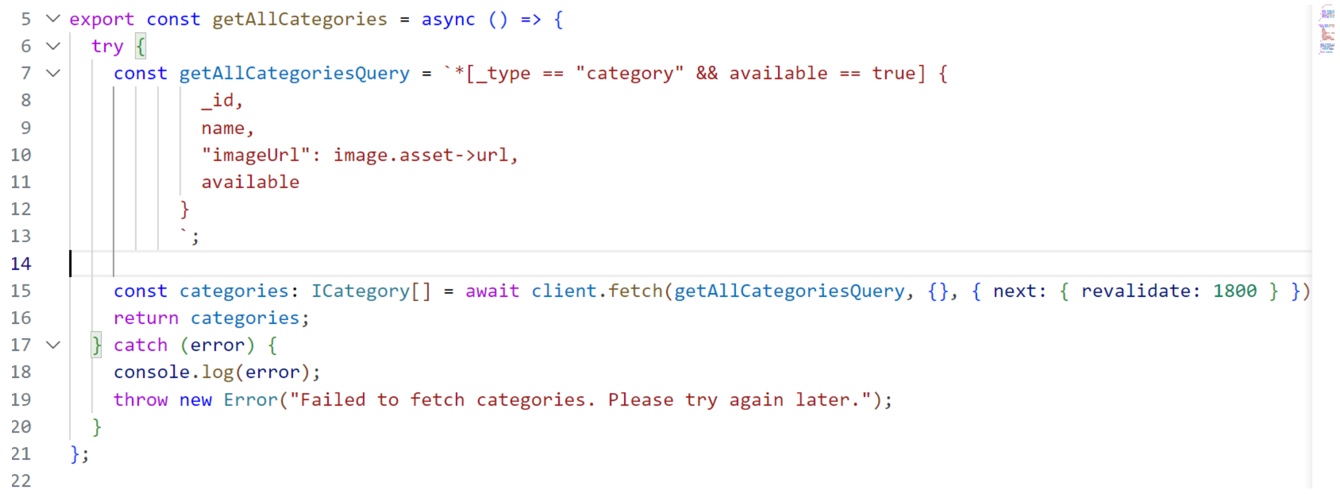
Choose & Pick Component API Integration Code (Home page):

```
"use client";
import { Great_Vibes } from "@next/font/google";
import { useEffect, useState } from "react";
import { getCategoriesWithFoods } from "@/sanity/lib/data";
import { ICategoryWithFoods } from "@/sanity/lib/interfaces";

const greatVibes = Great_Vibes({
  weight: ["400"],
  subsets: ["latin"],
});

const ChooseAndPick = () => {
  const [categories, setCategories] = useState<ICategoryWithFoods[]>([]);
  const [activeTab, setActiveTab] = useState(0);

  useEffect(() => {
    const fetchCategoriesWithFoods = async () => {
      try {
        const data = await getCategoriesWithFoods();
        setCategories(data);
      } catch (error) {
        console.error("Error fetching categories with foods:", error);
      }
    };
    fetchCategoriesWithFoods();
  }, []);

  return (
    <>
      <div className="sec6 px-[20px] sm:px-[60px] py-[60px] max-w-[1320px] lg:h-[800px] mx-auto flex
items-center justify-center">
        <div className="mt-8">
          <div className="flex flex-col items-center">
            <h5
              className={`${greatVibes.className} text-[32px] text-[#FF9F0D] font-normal `}
            >
              Choose & pick
            </h5>
            <h2
              className="text-white text-[48px] font-bold text-center"
              style={{ fontFamily: "Helvetica, Arial, sans-serif" }}
            >
              <span className="text-[#FF9F0D]">Fr</span>om Our Menu
            </h2>
          </div>

          <div className="flex md:flex-row flex-col">
            {categories.map((category, index) => (
              <button
                key={category._id}
                className={`flex-1 py-2 text-center font-medium text-lg ${activeTab === index ? "border-b-2
border-blue-500 text-blue-500" : "text-gray-500 hover:text-blue-500"}`}
                onClick={() => setActiveTab(index)}
              >
                {category.name}
              </button>
            ))}
          </div>

          <div className="p-4 text-gray-700 flex flex-col items-center">
            <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4 items-center">
              <div className="col-span-1 md:col-span-1 w-[100%] lg:max-w-[300px] h-[330px] relative group
cursor-pointer w-full ">
                <img
                  src={categories[activeTab]?.imageUrl}
                  className="w-[100%] h-[100%] rounded-[6px] object-center object-cover"
                  alt=""
                />
                <div className="absolute inset-0 flex justify-center items-center bg-black bg-opacity-50 opacity-0 
group-hover:opacity-100 transition-opacity duration-300">
                  <div className="text-white text-center flex flex-col gap-[5px]">
                    <div className="rounded-[6px] bg-white px-4 py-3 text-[#FF9F0D] w-fit font-bold text-
[18px]">
              Save 30%
                    </div>
                    <div className="rounded-[6px] bg-[#FF9F0D] text-white px-4 py-3 w-[250px] text-[20px] font-
bold">
              {categories[activeTab]?.name}
                    </div>
                  </div>
                </div>
              </div>

              <div className="col-span-2 flex flex-wrap gap-[30px] w-full">
                <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-2 gap-[30px] w-full max-h-[500px] 
overflow-y-auto">
                  {categories[activeTab]?.foods.map((food) => (
                    <a
              href={`/shop/${food._id}`}
              key={food._id}
              className="flex gap-[20px] cursor-pointer"
                    >
              <div className="w-[80px] h-[80px]">
                <img
                  src={food.imageUrl}
                  className="w-[100%] h-[100%] rounded-[6px] object-center object-cover"
                  alt={food.name}
                />
              </div>
              <div className="flex flex-col text-white">
                <h5 className="font-bold text-[20px]">{food.name}</h5>
                <p className="m-0 text-[14px] font-normal">
                  {food.description}
                </p>
                <p className="m-0 font-bold text-[18px] text-[#FF9F0D]">
                  {food.price} $
                </p>
              </div>
                    </a>
                  ))}
                </div>
              </div>
            </div>
```

```
          </div>
        </div>
      </div>
    </>
  );
};

export default ChooseAndPick;
```

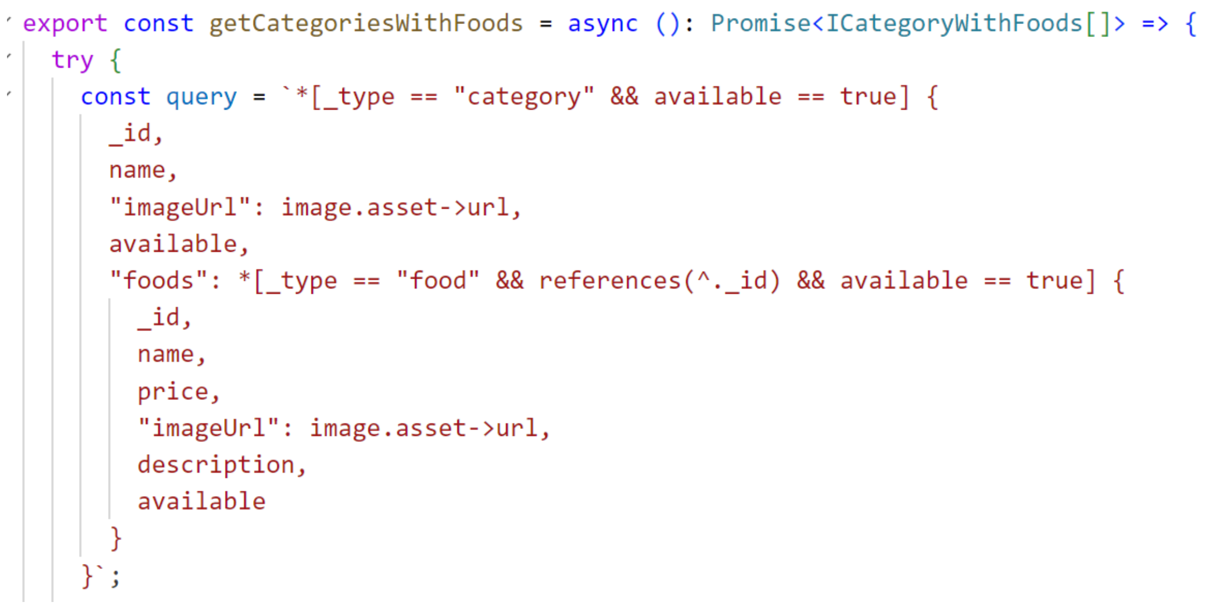
- API Calls (Get data by GROQ query):

1) Get All Categories:



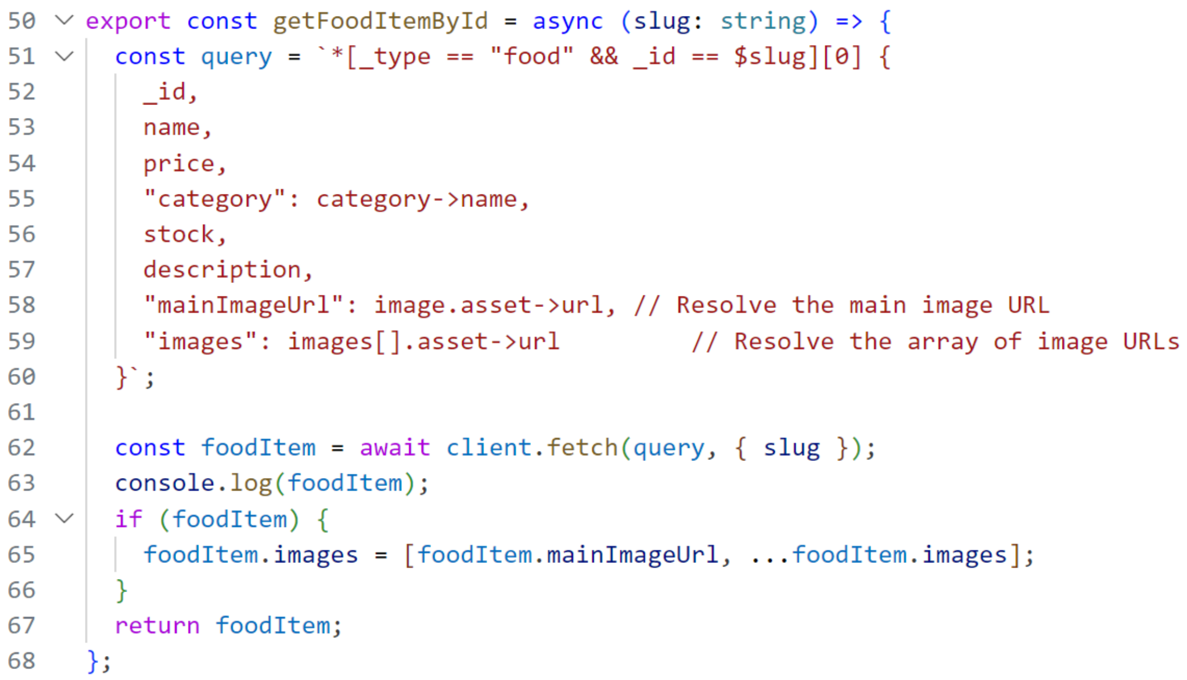
```
export const getAllCategories = async () => {
  try {
    const getAllCategoriesQuery = `*[_type == "category" && available == true] {
            _id,
            name,
            "imageUrl": image.asset->url,
            available
          }
          `;

    const categories: ICategory[] = await client.fetch(getAllCategoriesQuery, {}, { next: { revalidate: 1800 } })
    return categories;
  } catch (error) {
    console.log(error);
    throw new Error("Failed to fetch categories. Please try again later.");
  }
};
```

2) Get all categories data along with food items:



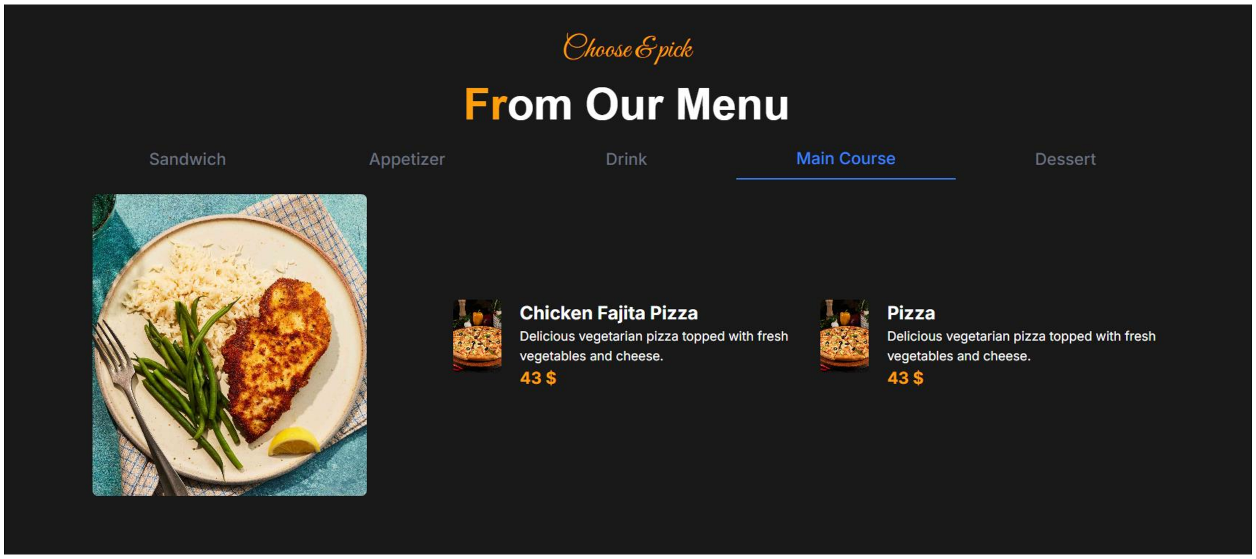
```
export const getCategoriesWithFoods = async (): Promise<ICategoryWithFoods[]> => {
  try {
    const query = `*[_type == "category" && available == true] {
      _id,
      name,
      "imageUrl": image.asset->url,
      available,
      "foods": *[_type == "food" && references(^._id) && available == true] {
        _id,
        name,
        price,
        "imageUrl": image.asset->url,
        description,
        available
      }
    }`;
```

3) Get single food item details:



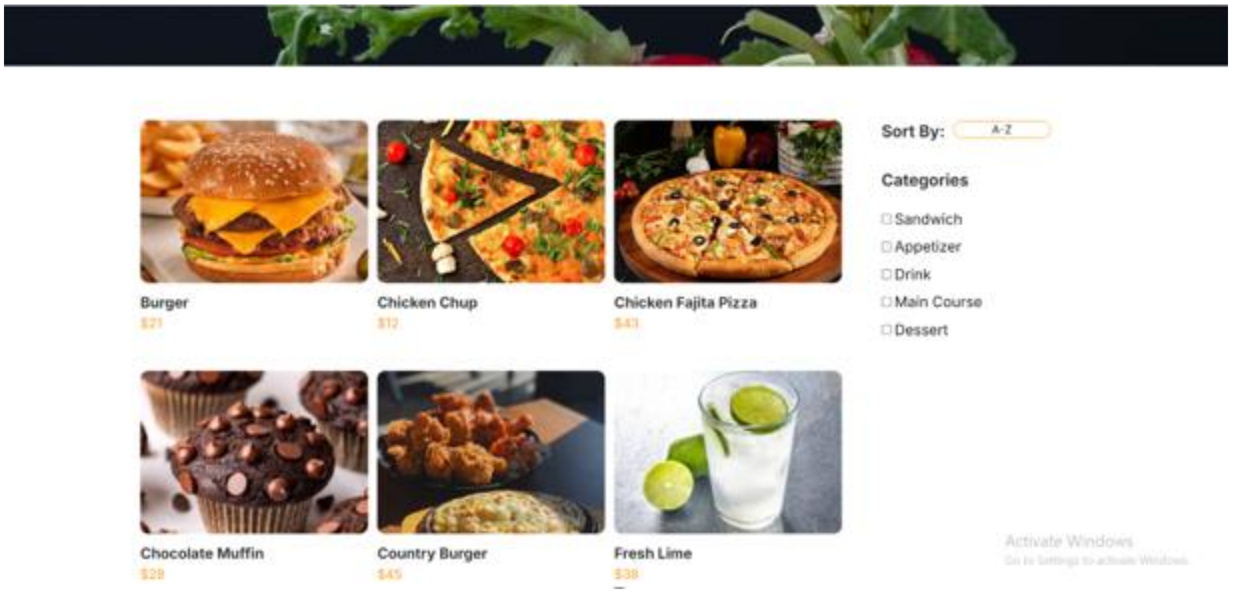
```
export const getFoodItemById = async (slug: string) => {
  const query = `*[_type == "food" && _id == $slug][0] {
    _id,
    name,
    price,
    "category": category->name,
    stock,
    description,
    "mainImageUrl": image.asset->url, // Resolve the main image URL
    "images": images[].asset->url         // Resolve the array of image URLs
  }`;

  const foodItem = await client.fetch(query, { slug });
  console.log(foodItem);
  if (foodItem) {
    foodItem.images = [foodItem.mainImageUrl, ...foodItem.images];
  }
  return foodItem;
};
```

Data Displayed in Frontend Next JS Website:



Chicken Fajita Pizza
Delicious vegetarian pizza topped with fresh vegetables and cheese.
43 $

Pizza
Delicious vegetarian pizza topped with fresh vegetables and cheese.
43 $



Burger
$21

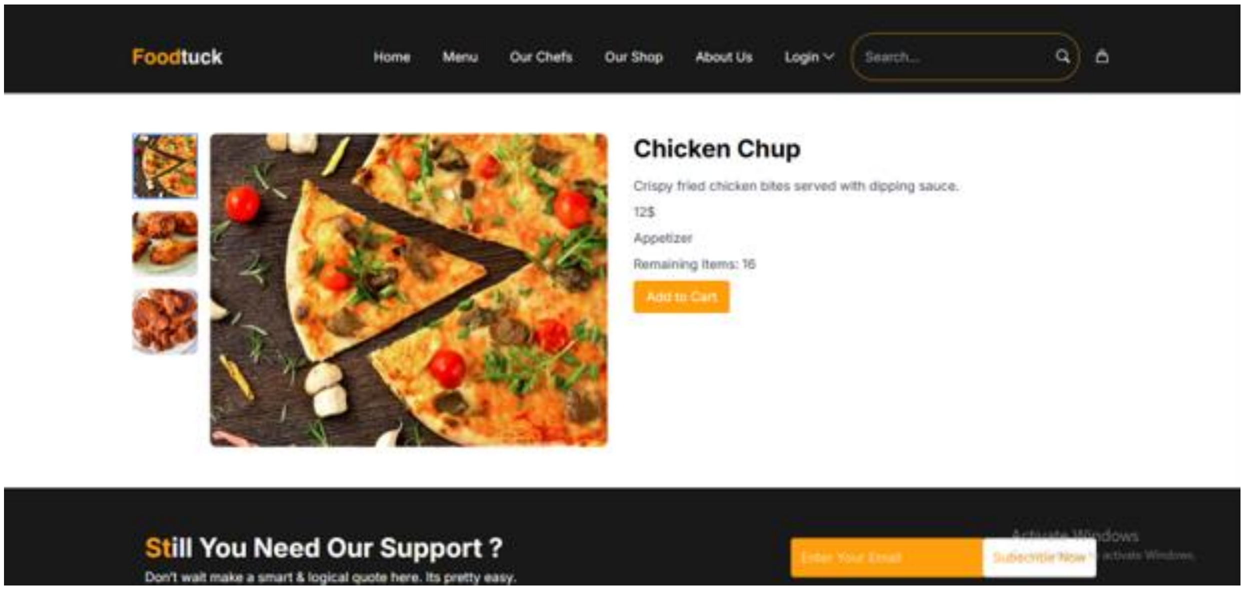
Chicken Chup
$12

Chicken Fajita Pizza
$43

Chocolate Muffin
$28

Country Burger
$45

Fresh Lime
$38



Chicken Chup

Crispy fried chicken bites served with dipping sauce.

12$

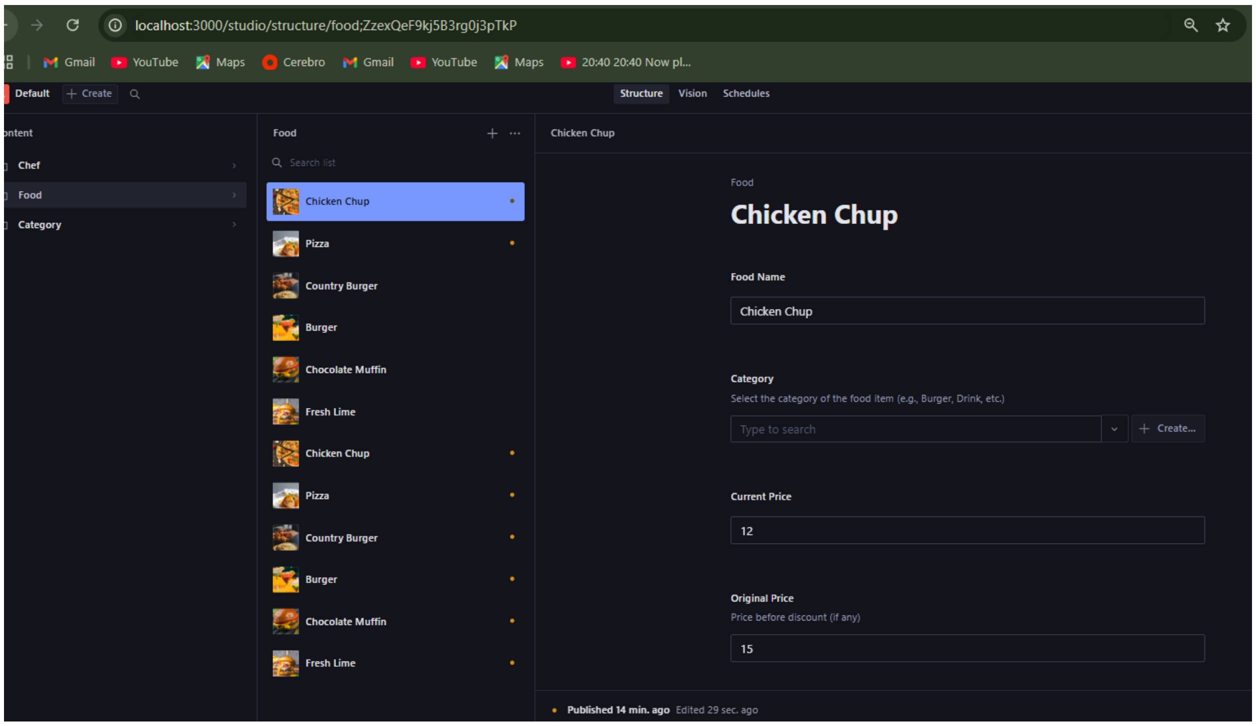
Appetizer

Remaining Items: 16

Still You Need Our Support ?

Don't wait make a smart & logical quote here. Its pretty easy.

Populated Sanity CMS Fields:


localhost:3000/studio/structure/food;ZzexQeF9kj5B3rg0j3pTkP

Gmail YouTube Maps Cerebro Gmail YouTube Maps 20:40 20:40 Now pl...

Default + Create

Structure Vision Schedules

ntent
- Chef
- Food
- Category

Food

Search list

- Chicken Chup
- Pizza
- Country Burger
- Burger
- Chocolate Muffin
- Fresh Lime
- Chicken Chup
- Pizza
- Country Burger
- Burger
- Chocolate Muffin
- Fresh Lime

Chicken Chup

Food

Chicken Chup

Food Name

Chicken Chup

Category

Select the category of the food item (e.g., Burger, Drink, etc.)

Type to search | + Create...

Current Price

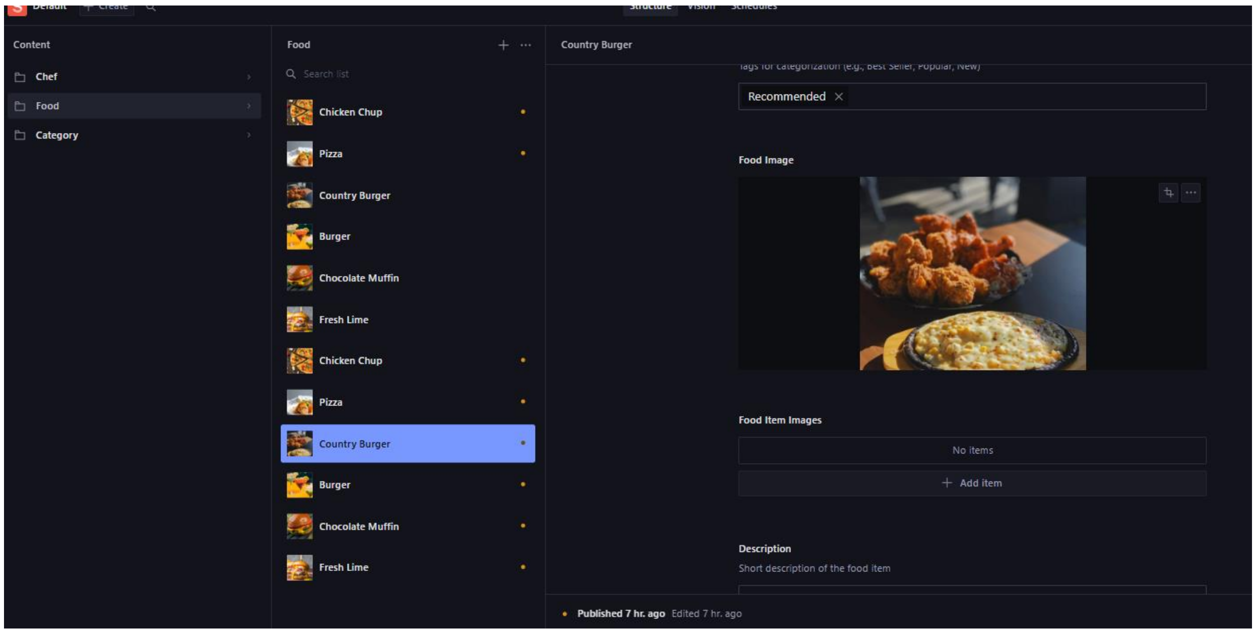
12

Original Price

Price before discount (if any)

15

Published 14 min. ago Edited 29 sec. ago


Default + Create

Structure Vision Schedules

Content
- Chef
- Food
- Category

Food

Search list

- Chicken Chup
- Pizza
- Country Burger
- Burger
- Chocolate Muffin
- Fresh Lime
- Chicken Chup
- Pizza
- Country Burger
- Burger
- Chocolate Muffin
- Fresh Lime

Country Burger

Tags for categorization (e.g., Best Seller, Popular, New)

Recommended ×

Food Image

Food Item Images

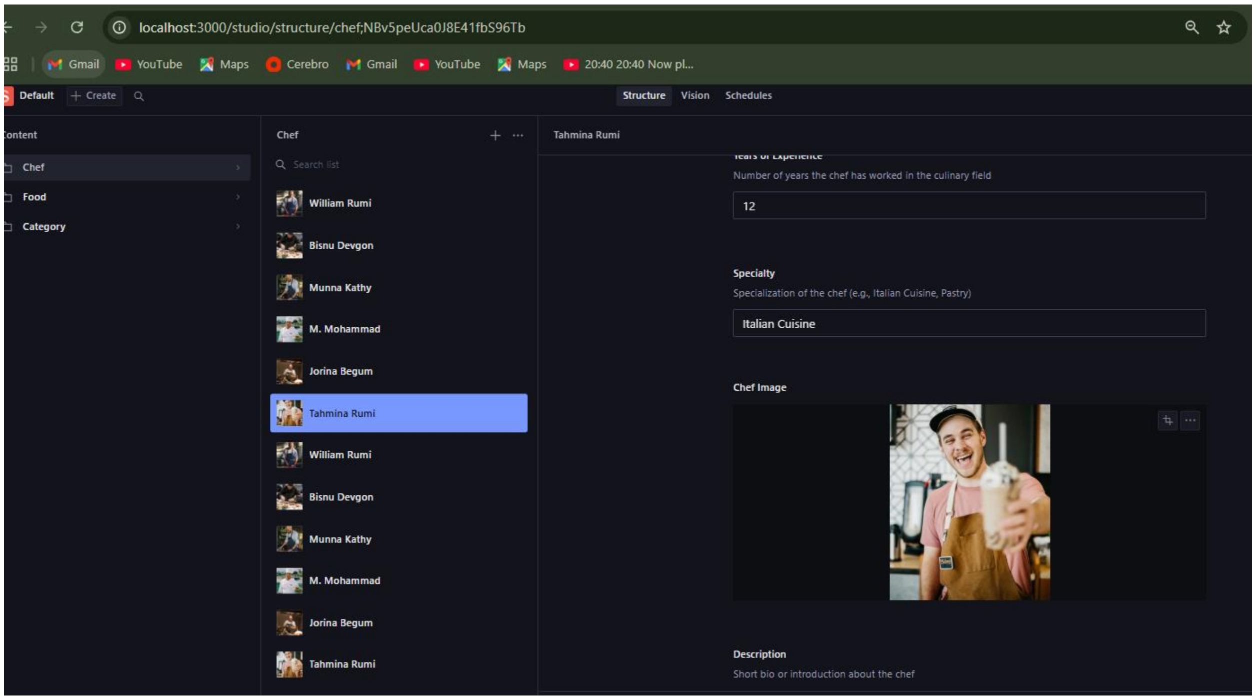
No items

+ Add item

Description

Short description of the food item

Published 7 hr. ago Edited 7 hr. ago



Years of Experience
Number of years the chef has worked in the culinary field
12

Specialty
Specialization of the chef (e.g., Italian Cuisine, Pastry)
Italian Cuisine

Chef Image

Description
Short bio or introduction about the chef

Created by:

Ismat Fatima (00373389)