

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:

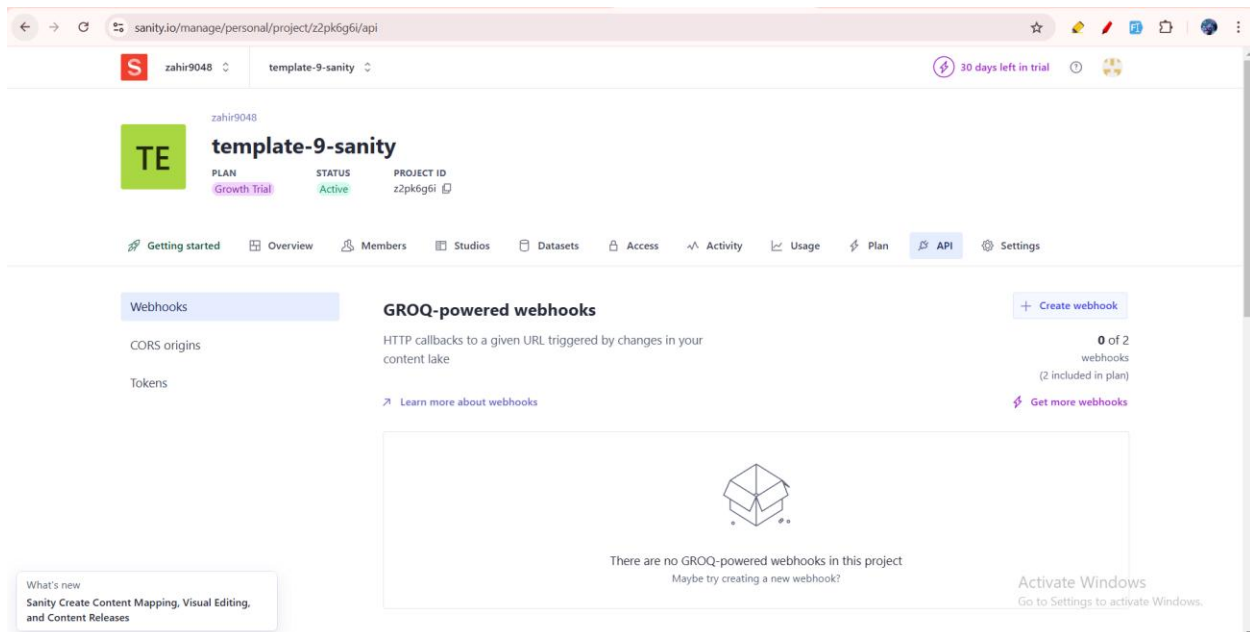
```
{  
  name: 'images',  
  type: 'array',  
  title: 'Food Item Images',  
  of: [{ type: 'image' }] // Array of images for the detail page  
},
```

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

```
65  {  
66    name: "stock",  
67    type: "number",  
68    title: "Stock",  
69    description: "Inventory",  
70  }
```

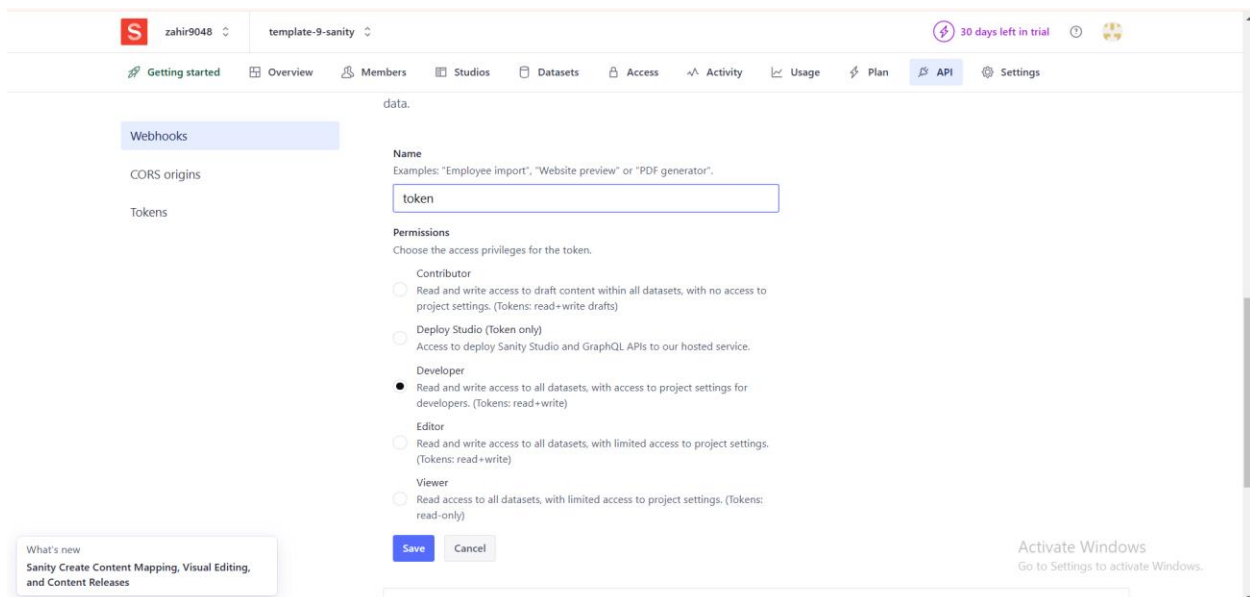
- Migration steps and tools used:

1) Setup sanity project in the frontend Next JS website:

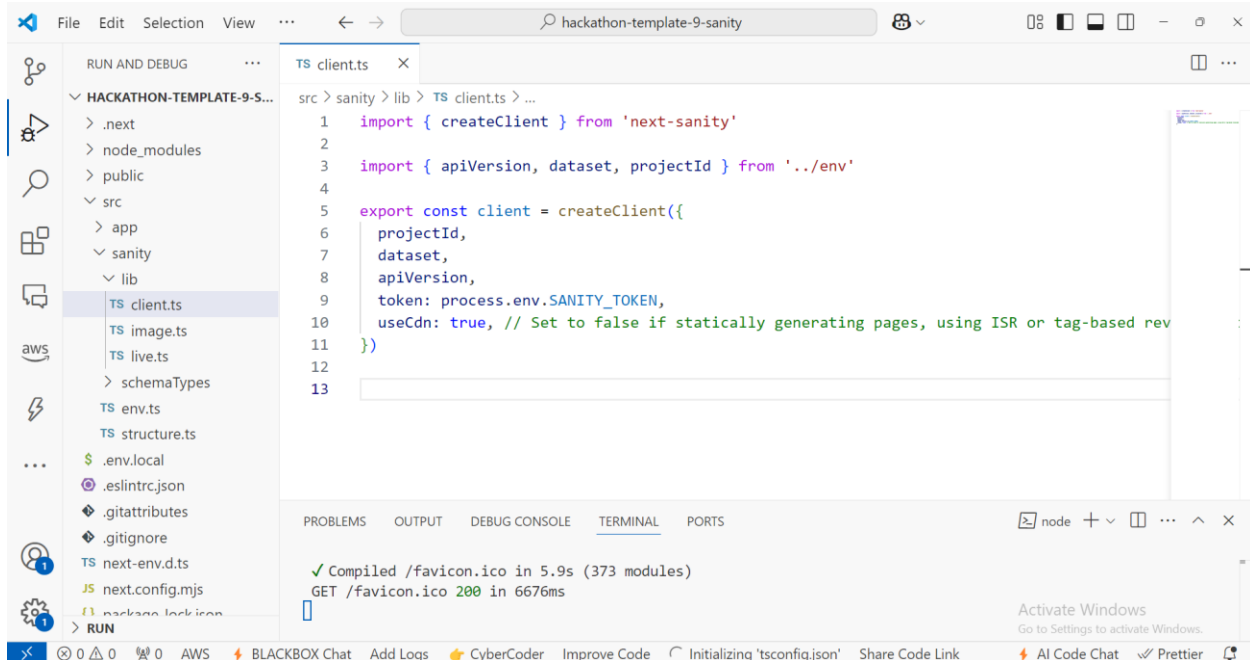


Sanity Project created.

2) Generate editor access API token in project settings:



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



```
src > sanity > lib > TS client.ts > ...
1  import { createClient } from 'next-sanity'
2
3  import { apiVersion, dataset, projectId } from '../env'
4
5  export const client = createClient({
6    projectId,
7    dataset,
8    apiVersion,
9    token: process.env.SANITY_TOKEN,
10   useCdn: true, // Set to false if statically generating pages, using ISR or tag-based rendering
11 })
12
13
```

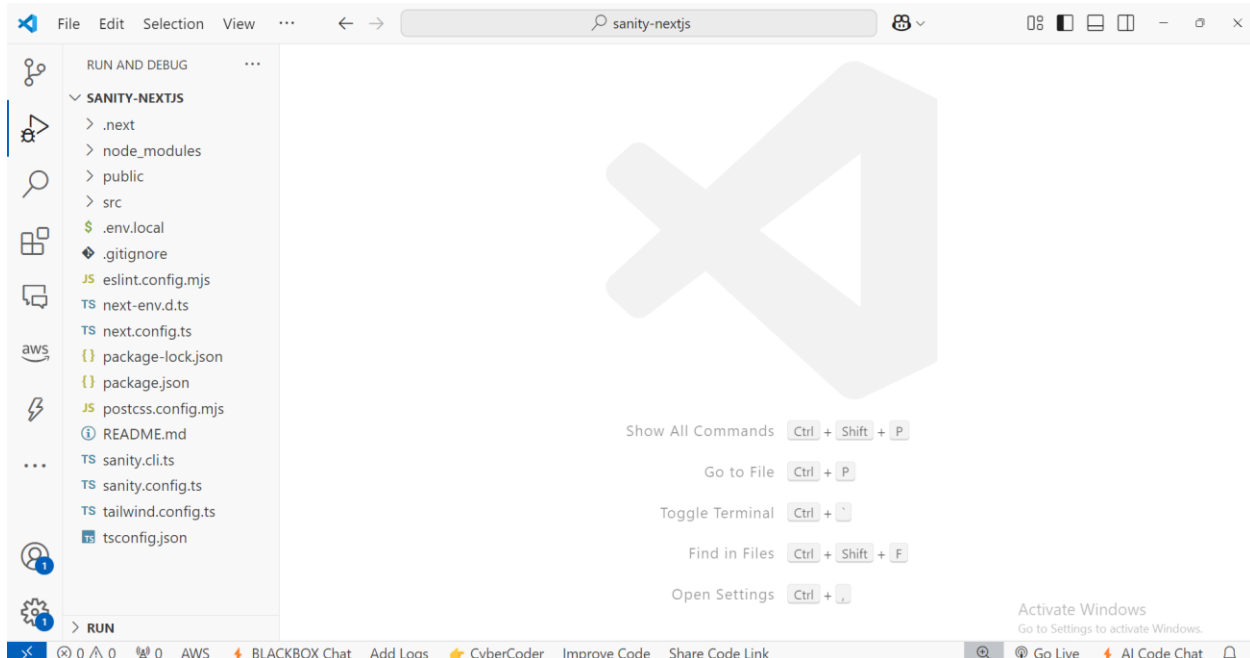
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

node + ... ^ X

✓ Compiled /favicon.ico in 5.9s (373 modules)
GET /favicon.ico 200 in 6676ms

Activate Windows
Go to Settings to activate Windows.

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



```
File Edit Selection View ...
sanity-nextjs
```

RUN AND DEBUG

SANITY-NEXTJS

- > .next
- > node_modules
- > public
- > src
- \$.env.local
- ◆ .gitignore
- JS eslint.config.mjs
- TS next-env.d.ts
- TS next.config.ts
- { } package-lock.json
- { } package.json
- JS postcss.config.mjs
- ① README.md
- TS sanity.cli.ts
- TS sanity.config.ts
- TS tailwind.config.ts
- TS tsconfig.json

RUN

Show All Commands **Ctrl** + **Shift** + **P**

Go to File **Ctrl** + **P**

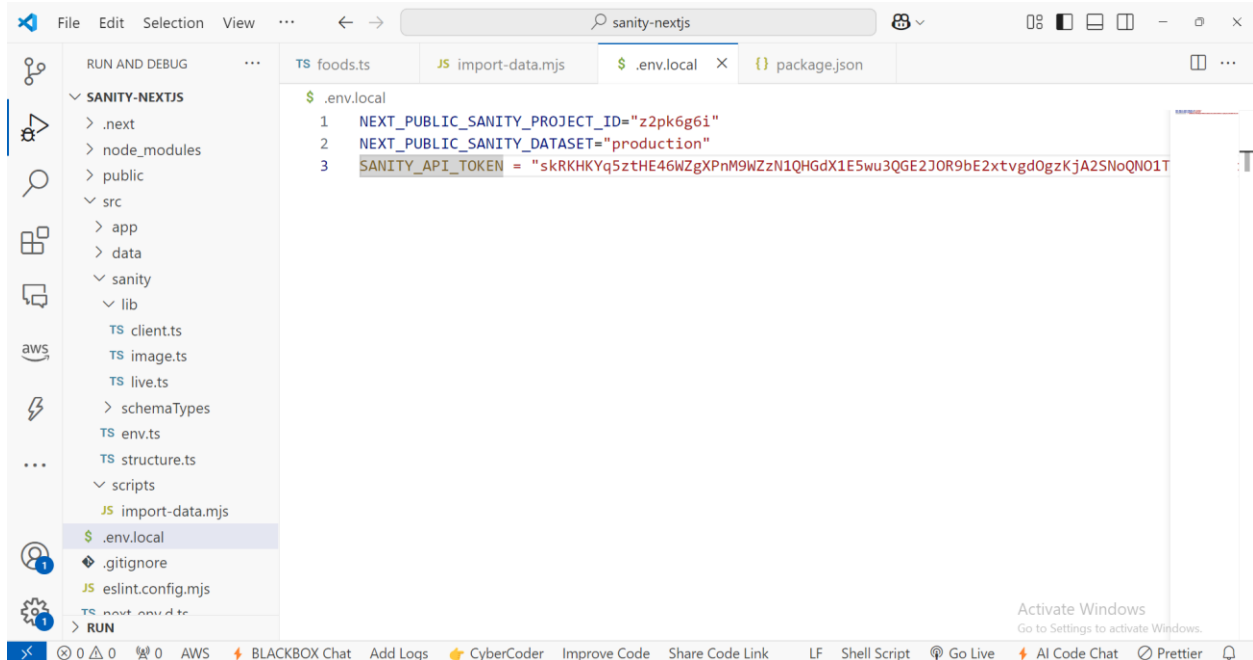
Toggle Terminal **Ctrl** + **`**

Find in Files **Ctrl** + **Shift** + **F**

Open Settings **Ctrl** + **,**

Activate Windows
Go to Settings to activate Windows.

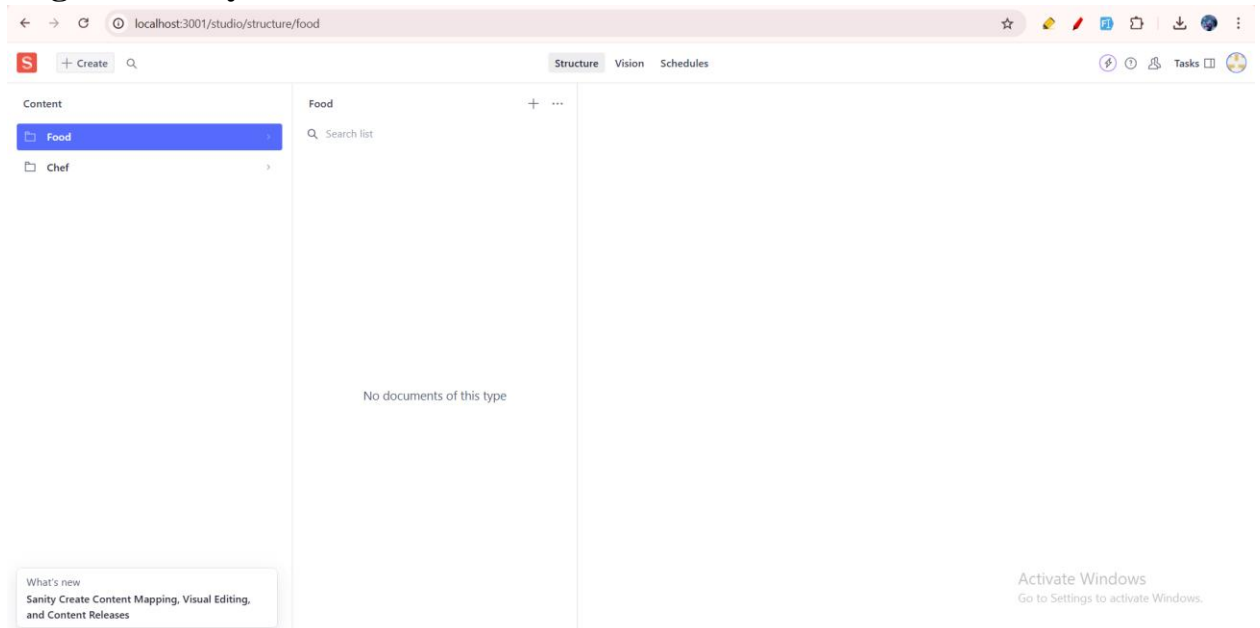
5) Now create .env.local file and set project id and generated api token in .env.local file.



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The file explorer shows the project structure for 'sanity-nextjs'. The file '.env.local' is selected and its contents are displayed in the editor. The contents of the file are:

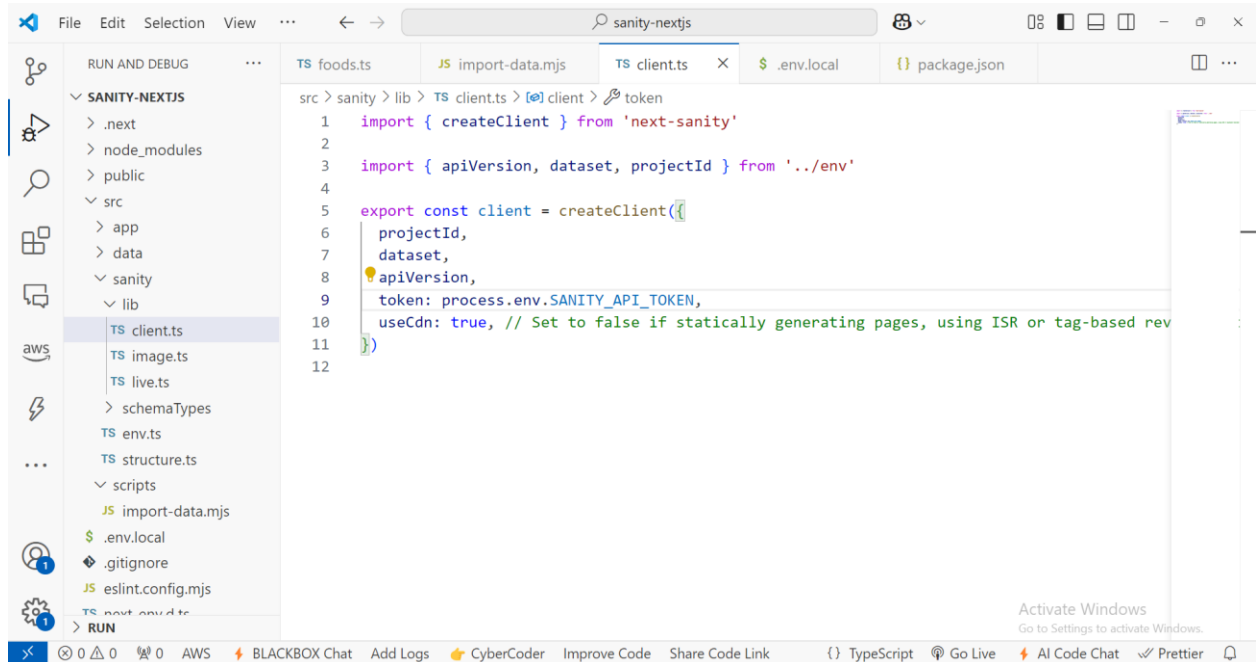
```
1 NEXT_PUBLIC_SANITY_PROJECT_ID="z2pk6g61"  
2 NEXT_PUBLIC_SANITY_DATASET="production"  
3 SANITY_API_TOKEN = "skRKHkYq5ztHE46WZgXPnM9WZzN1QHGDx1E5wu3QGE2J0R9bE2xtvgd0gzKjA2SNoQN01T"
```

6) Login to sanity account in /studio:

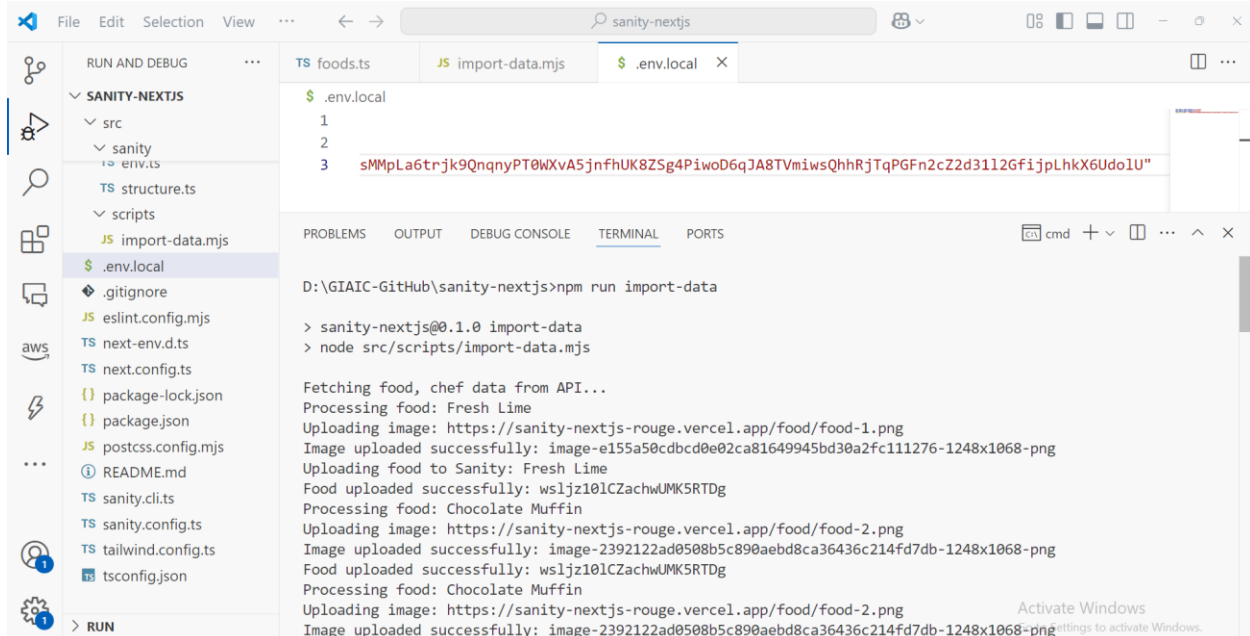


You can see the defined schemas here.

7) Pass token inside client function in client.ts file.



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

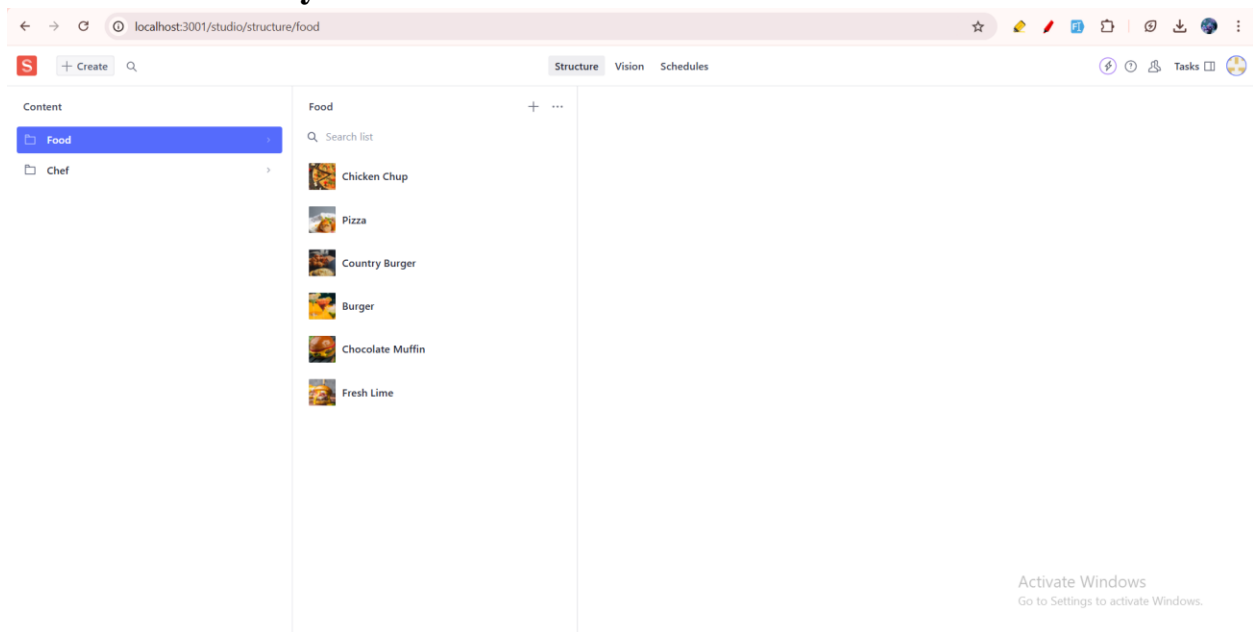


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  [CA] cmd + v |
Image uploaded successfully: image-03eb4eacebd8ca11b707cfc569b87894b4e9bd57-1248x1517-png
Uploading chef to Sanity: Munna Kathy
Chef uploaded successfully: wsljz10lCZachwUMK5RUCy
Processing chef: Bisnu Devgon
Uploading image: https://sanity-nextjs-rouge.vercel.app/chef/chef-5.png
Image uploaded successfully: image-7576fb850ddb0f7d4cefab457f848c09a816186d-1248x1517-png
Uploading chef to Sanity: Bisnu Devgon
Chef uploaded successfully: IZ4ZA8avncRcTDvyubNd2q
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: wsljz10lCZachwUMK5RUPE
Data import completed successfully!

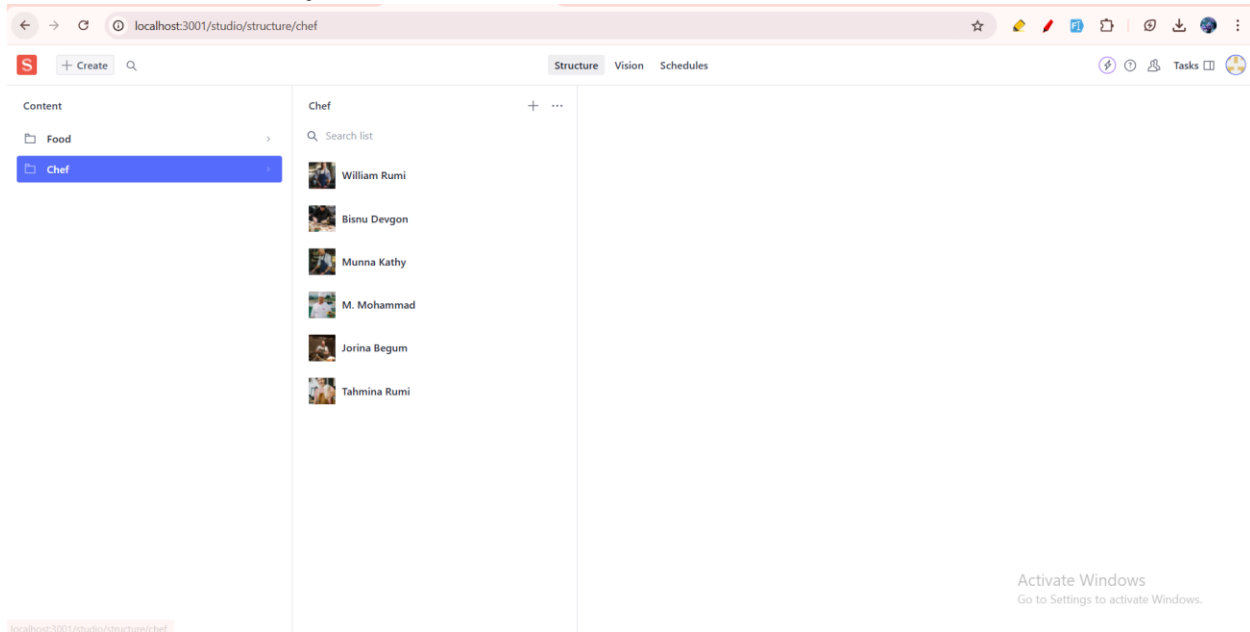
D:\GIAIC-GitHub\sanity-nextjs>
```

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]">From 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>

```

[illegible]

- API Calls (Get data by GROQ query):

1) Get All Categories:

```
5 export const getAllCategories = async () => {
6   try {
7     const getAllCategoriesQuery = `*_type == "category" && available == true` {
8       _id,
9       name,
10      "imageUrl": image.asset->url,
11      available
12    }
13  };
14
15  const categories: ICategory[] = await client.fetch(getAllCategoriesQuery, {}, { next: { revalidate: 1800 } })
16  return categories;
17 } catch (error) {
18   console.log(error);
19   throw new Error("Failed to fetch categories. Please try again later.");
20 }
21 };
22
```

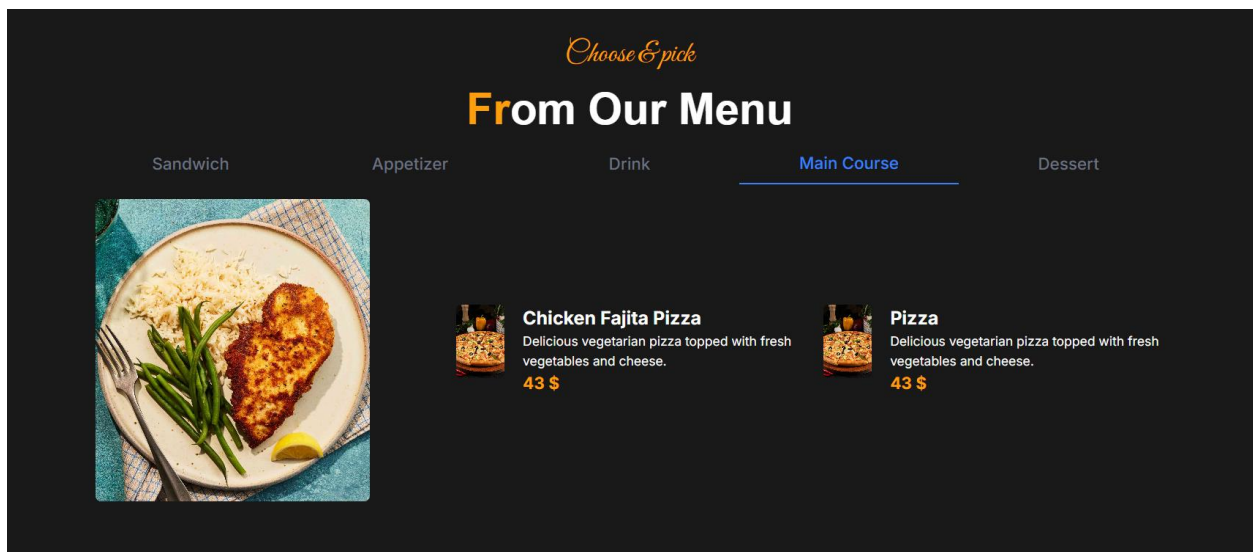
2) Get all categories data along with food items:

```
export const getCategoriesWithFoods = async (): Promise> => {
  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:

```
50 ∨ export const getFoodItemById = async (slug: string) => {
51 ∨   const query = `*[_type == "food" && _id == $slug][0] {
52     _id,
53     name,
54     price,
55     "category": category->name,
56     stock,
57     description,
58     "mainImageUrl": image.asset->url, // Resolve the main image URL
59     "images": images[].asset->url      // Resolve the array of image URLs
60   }`;
61
62   const foodItem = await client.fetch(query, { slug });
63   console.log(foodItem);
64 ∨   if (foodItem) {
65     foodItem.images = [foodItem.mainImageUrl, ...foodItem.images];
66   }
67   return foodItem;
68 };
```

Data Displayed in Frontend Next JS Website:





- Populated Sanity CMS Fields:

