

# DAY 3 - API INTEGRATION AND DATA MIGRATION

## Importing Food and Chef Data from an External API1.

### Setting Up Environment Variables

```
NEXT_PUBLIC_SANITY_PROJECT_ID=your_project_id
NEXT_PUBLIC_SANITY_DATASET=production
SANITY_API_TOKEN=your_sanity_token
```

### 2. Fetching Sanity Project ID and API Token

### 3. Creating the Sanity Schemas

#### Food Schema (food.ts)

```
c > sanity > schemaTypes > TS foods.ts > (e) default > fields
1 export default {
2   name: 'food',
3   type: 'document',
4   title: 'Food',
5   fields: [
6     {name: 'name', type: 'string', title: 'Food Name',
7     },
8     {name: 'category', type: 'string', title: 'Category', description:
9       'Category of the food item (e.g., Burger, Sandwich, Drink, etc.)',
10    },
11    {name: 'price', type: 'number', title: 'Current Price',
12    },
13    {
14      name: 'originalPrice',
15      type: 'number',
16      title: 'Original Price',
17      description: 'Price before discount (if any)',
18    },
19    {name: 'tags', type: 'array', title: 'Tags', of: [{ type: 'string' }],
20      options: {
21        layout: 'tags',
22      },
23      description: 'Tags for categorization (e.g., Best Seller, Popular, New)',
24    },
25    {name: 'image', type: 'image', title: 'Food Image',
26      options: {
27        hotspot: true,
28      },
29    },
30    {name: 'description', type: 'text', title: 'Description', description: 'Short description of the food item',
31    },
32    {name: 'available', type: 'boolean', title: 'Available', description: 'Availability status of the food item',
33    },
34    {name: 'slug', type: 'slug', title: 'Slug', description: 'Unique identifier for the food item, generated from the name', options: { source
35    },
36    },
37  ],
38 }
```

## Chef Schema (`chef.ts`)

```
export default {
  name: 'chef',
  type: 'document',
  title: 'Chef',
  fields: [
    { name: 'name', type: 'string', title: 'Chef Name' },
    { name: 'position', type: 'string', title: 'Position' },
    { name: 'experience', type: 'string', title: 'Experience' },
    { name: 'specialty', type: 'string', title: 'Specialty' },
    { name: 'description', type: 'string', title: 'Description' },
    { name: 'available', type: 'boolean', title: 'Available' },
    { name: 'image', type: 'image', title: 'Image', options: { hotspot: true } }
  ],
};
```

Update your `schemaTypes/index.ts` to include these schemas.

```
c > sanity > schemaTypes > ts index.ts > ...
1  import { type SchemaTypeDefinition } from 'sanity';
2  import chef from './chefs';
3  import food from './foods';
4
5  export const schema: { types: SchemaTypeDefinition[] } = {
6    types: [food, chef],
7  };
```

## 4. Writing the Data Import Script

`scripts/importSanityData.mjs`

```

export default {
  async function uploadImageToSanity(imageId) {
    try {
      console.log('uploading image: ' + imageId);
      const response = await axios.get(imageId, { responseType: 'arraybuffer' });
      const buffer = Buffer.from(response.data);
      const asset = await client.assets.upload('image', buffer, {
        filename: imageId.split('/') + '.png'
      });
      console.log('image uploaded successfully: ' + asset._id);
      return asset._id;
    } catch (error) {
      console.error('failed to upload image', imageId, error);
      return null;
    }
  }

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

      // get restaurant containing data
      const $venues = [];
      $venues.push(
        axios.get('https://sanity-test-jp-rhago.vercel.app/api/foods')
      );
      $venues.push(
        axios.get('https://sanity-test-jp-rhago.vercel.app/api/chefs')
      );

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

      for (const food of foods) {
        console.log('processing food: ' + food.name);

        let imageId = null;
        if (food.image) {
          imageId = 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: imageId
        };

        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 imageId = null;
        if (chef.image) {
          imageId = 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: imageId
        };

        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);
    }
  }
}

```

## 5. Running the Import Script

```
npm install @sanity/client axios dotenv
```

Run the script:

```
npm run import-data
```

**Now ,**

**Fetching Data from Sanity in Next.js for a Food Menu**

## **Step 1: Setting up the Sanity Client**

First, we need to set up the Sanity client to fetch data from our Sanity project. We'll create a new file, `lib/sanity.ts`, to initialize the client.

**lib/sanity.ts**

```
import { createClient } from '@sanity/client';
```

```
export const client = createClient({
  projectId: process.env.SANITY_PROJECT_ID, // from your sanity.json
  dataset: 'production',
  useCdn: true,
  token: process.env.SANITY_API_TOKEN, // Add your token here if
  needed
  apiVersion: '2025-01-17', // Always use the latest version
});
```

## **Step 2: Creating the FetchFood Component**

**Src/components/shopProduct.tsx**

components > @ ShopProducts.tsx > ...

```
"use client";

import React, { useState, useEffect } from "react";
import Link from "next/link";
import Image from "next/image";
import { client } from "@sanity/lib/client"; // Sanity client import

// Define types for the data being fetched
interface Food {
  slug: string;
  name: string;
  category: string;
  description: string;
  price: number;
  originalPrice: number;
  tags: string | string[];
  available: boolean;
  imageUrl: string;
  _createdAt: string;
  _updatedAt: string;
}

const ShopProduct = () => {
  const [foods, setFoods] = useState<Food[]>([]);
  const [filteredFoods, setFilteredFoods] = useState<Food[]>([]);
  const [searchQuery, setSearchQuery] = useState<string>("");
  const [loading, setLoading] = useState<boolean>(true);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const query = `*[ _type == "food" ] {
          name,
          category,
          description,
          price,
          originalPrice,
          available,
          tags,
          "slug": slug.current,
          "imageUrl": image.asset->url
        }`;
        const products = await client.fetch(query);
        setFoods(products);
        setFilteredFoods(products);
      } catch (err: unknown) {
        console.error("Error fetching data:", err);
        setError("An unexpected error occurred");
      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, []);
```

```

const ShopProduct = () => {

  const handleSearch = (event: React.ChangeEvent) => {
    const query = event.target.value.toLowerCase();
    setSearchQuery(query);

    const filtered = foods.filter((food) =>
      food.name.toLowerCase().includes(query) ||
      (Array.isArray(food.tags)
        ? food.tags.some((tag) => tag.toLowerCase().includes(query))
        : food.tags?.toLowerCase().includes(query))
    );

    setFilteredFoods(filtered);
  };

  if (loading) {
    return (
      <div className="flex items-center justify-center min-h-screen bg-gray-50">
        <p className="text-xl font-medium text-gray-700">Loading...</p>
      </div>
    );
  }

  if (error) {
    return (
      <div className="flex items-center justify-center min-h-screen bg-gray-50">
        <p className="text-xl font-medium text-red-600">Error: {error}</p>
      </div>
    );
  }

  return (
    <div className="bg-gray-50 py-10">
      <div className="container mx-auto flex flex-wrap">
        /* Left Section */
        <div className="w-full lg:w-[70%] px-4 mb-10">
          <h1 className="text-3xl font-bold text-center mt-20 text-gray-800 mb-8">
            Food List
          </h1>

          <div className="mb-6">
            <input
              type="text"
              placeholder="Search by name or tags..."
              value={searchQuery}
              onChange={handleSearch}
              className="w-full p-3 border border-gray-300 rounded-lg"
            />
          </div>

          {filteredFoods.length === 0 ? (
            <p className="text-center text-lg font-medium text-gray-500">
              No food items found.
            </p>
          ) : (
            <ul className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6">
              {filteredFoods.map((food) => (
                <li

```

```

      {filteredFoods.map((food) => (
        <li
          key={food.slug}
          className="bg-white border border-gray-200 rounded-lg shadow-md hover:shadow-lg transition-shadow overflow-hidden"
        >
          <Link href={`./ourshops/${food.slug}`}>
            <div className="relative h-[300px] w-full bg-gray-100 overflow-hidden group">
              <Image
                src={food.imageUrl}
                alt={food.name}
                layout="fill"
                objectFit="cover"
                className="transform transition-transform duration-300 group-hover:scale-110"
              />
              <div className="absolute inset-0 bg-black bg-opacity-0 group-hover:bg-opacity-20 transition-all duration-300"></div>
            </div>
            <div className="p-6">
              <h2 className="text-xl font-semibold text-gray-800 mb-2">
                {food.name}
              </h2>
              <p className="text-sm text-gray-600 mb-4">
                {food.description}
              </p>
              <div className="flex items-center justify-between mb-4">
                <p className="text-lg font-medium text-green-600">
                  ${food.price.toFixed(2)}
                </p>
                <p className="text-sm line-through text-gray-500">
                  ${food.originalPrice.toFixed(2)}
                </p>
              </div>
              <div className="text-sm text-gray-500">
                <p>
                  Category: (" ")
                  <span className="font-medium">{food.category}</span>
                </p>
              </div>
            </div>
          </Link>
        </li>
      ))
    </ul>
  </div>
)
</div>

```

```

/* Right Section */


/* Filter Section */
  <div className="mb-6">
    <h3 className="text-xl font-bold mt-32 text-gray-700 mb-4">Category</h3>
    <div className="space-y-4">
      {[
        "Lime", "Orange", "Mango", "Burger", "Pizza", "Cake", "Wrap", "Soup", "Salat", "Biryani", "Sandwich"
      ].map(
        (category) => (
          <div key={category} className="flex items-center space-x-2">
            <input type="checkbox" id={category} />
            <label htmlFor={category} className="text-gray-700">
              {category}
            </label>
          </div>
        )
      )
    </div>
  </div>

  /* Latest Products */
  <div className="mb-6">
    <h3 className="text-xl font-bold text-gray-700 mb-4">Latest Products</h3>
    <div className="space-y-4">
      {[
        { name: "Chicken Tikka", price: 34, image: "/new.png" },
        { name: "custurd", price: 40, image: "/h17.png" },
        { name: "Muffin", price: 40, image: "/post2.png" },
        { name: "Salat", price: 40, image: "/s3.png" }
      ].map((product) => (
        <div key={product.name} className="flex items-center space-x-4">
          <Image
            src={product.image}
            alt={product.name}
            width={71}
            height={65}
            className="object-cover"
          />
          <div>
            <p className="text-sm text-gray-700">{product.name}</p>
            <p className="text-sm text-[#FF9900]">${product.price}.00</p>
          </div>
        </div>
      ))
    </div>
  </div>
</div>
</div>


```

ort default ShopProduct;

Snip & Sketch

Snip saved to clipboard

## Step 4: Creating the Food Component

## Src/app/ourshops.tsx

```
> app > ourshops > page.tsx > [e] default
1 import React from 'react'
2 // import HeroSection from '../Components/HeroSection'
3 import ShopProduct from '../components/ShopProduct'
4 import CartProvider from '@components/Cart/CartProvider'
5
6 const Page = () => {
7   return (
8     <div>
9       | /* <HeroSection title='Our Shop' homeLink='/' currentPage='Shop ' backgroundImage='/starbg.png' /> */
10      <CartProvider>
11        | <ShopProduct/>
12        | </CartProvider>
13      </div>
14    )
15  }
16
17 export default Page
```

## Step 6: Running the Application

Data display successfully





