

Day 3 - API Integration Report – Food Tuck

Introduction: What is API Integration?

the process of connecting two or more applications or systems by using APIs (Application Programming Interfaces) to exchange data and perform actions.

End Points:

GET/foods

- Contain list of foods
- Purpose: to display list of food on UI
- Example:

```
{
  "name": "Fresh Lime",
  "category": "Drink",
  "price": 38.0,
  "originalPrice": 45.0,
  "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
},
```

GET/chefs

- Contain list of Chefs
- Purpose: display list of Chefs on UI
- Example:

```
{
  "name": "William Rumi",
  "position": "Chef de Cuisine",
  "experience": 18,
  "specialty": "Seafood Specialties",
  "image": "https://sanity-nextjs-rouge.vercel.app/chef/chef-6.png",
  "description": "Master of crafting exquisite seafood dishes with unique flavors.",
  "available": true
}
```

Day 3 - API Integration Report – Food Tuck

Steps of API Integration:

1. Setting up utility functions:

- To fetch data from these end points we created such a reusable Functions which will handle API calls
- Example

```
import { defineQuery } from "next-sanity";

export const Foods = defineQuery(
  `*[_type == "food"]{
    _id,
    name,
    description,
    price,
    category,
    originalPrice,
    tags,
    "imageUrl":image.asset -> url,
    available
  }`
)
```

```
export const ChefsData = defineQuery(
  `*[_type == "chef"]{
    _id,
    name,
    description,
    position,
    available ,
    experience,
    specialty,
    "imageUrl":image.asset -> url,
  }`
)
```

Day 3 - API Integration Report – Food Tuck

2. Rendering Data in Components:

- Now after setting up functions we will fetch it in our main component code to display it on screen
- Just Like

```
import { sanityFetch } from "@sanity/lib/live";
import { Foods15 } from '@sanity/lib/queries';
type Food = {
  _id: string;
  name: string;
  category: string;
  price: number;
  originalPrice: number;
  tags: string[];
  imageUrl: string;
  description: string;
  available: true
}

async function shop() {
  const response = (await sanityFetch({ query:Foods15})).data;
  const products: Food[] = response;
  return (
    <div className='grid grid-cols-3 gap-[32px] '>
      {products.map((food)=>(
        <div key={food._id} className='w-auto flex flex-col items-start justify-between gap-[8px] '>
          <div>
            <img src={food.imageUrl} width={312} height={267} alt={food.name}/>
          </div>
          <div className='w-full flex flex-col items-start justify-between gap-[8px] '>
            <h3 className='font-bold text-[18px] '>{food.name}</h3>
            <div className='w-auto flex flex-row items-start justify-between gap-[16px] '>
              <div className='text-[#FF9F0D] '>${food.originalPrice.toFixed(2)}</div>
              <div>{food.price? <div className='text-[#828282] line-through '>${food.price.toFixed(2)}</div> : <div className='hidden '></div>}</div>
            </div>
          </div>
        </div>
      )
    )
  )
}
```

Day 3 - API Integration Report – Food Tuck

```
        <Button variant="primary" size="none" className='p-1 px-2'><Link
href={` /Shop/${food._id}` } className='text-[14px]'>Order Now!</Link></Button>
        </div>
    )}
  </div>
)
}

export default shop
```

3. Testing Endpoints:

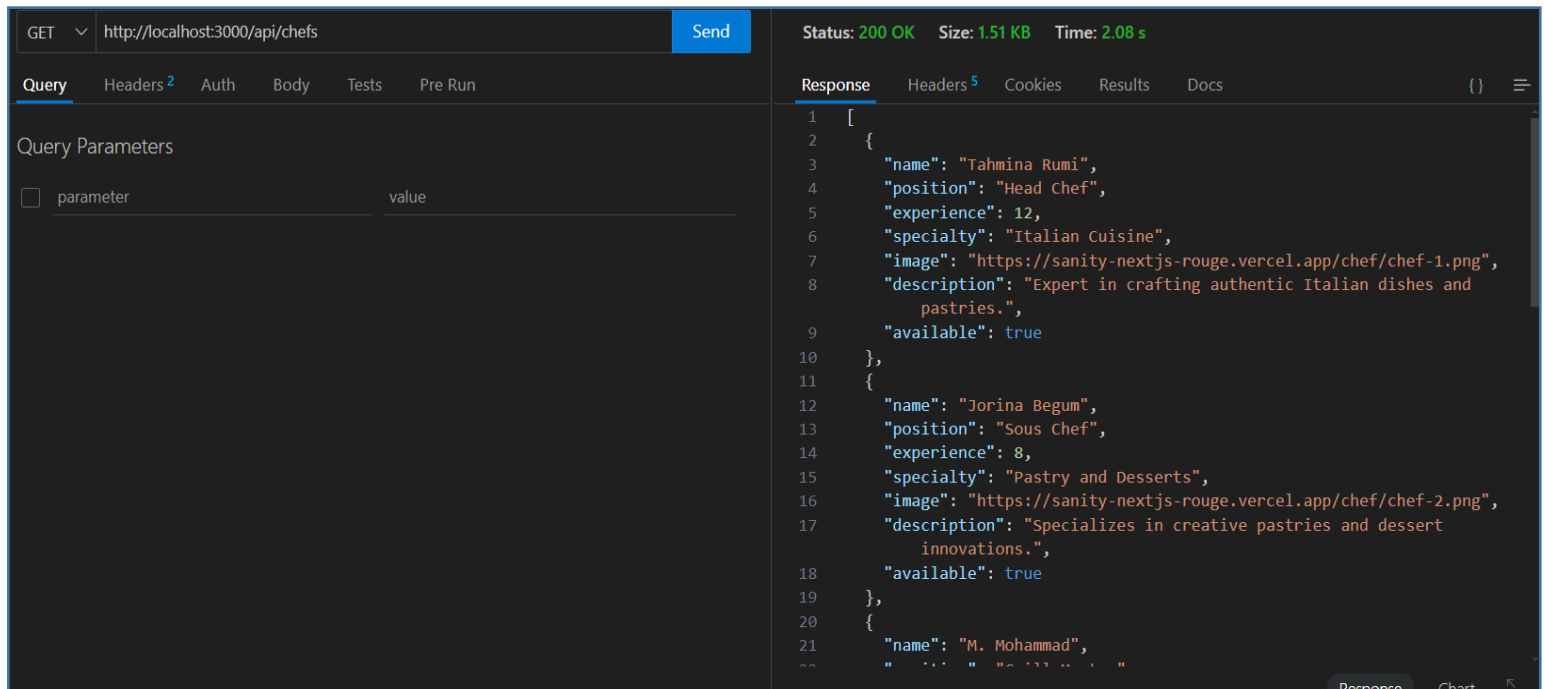
- To test endpoints, we used thunder client

Testimony:

The screenshot shows the Thunder Client interface. The top bar displays the method 'GET', the URL 'http://localhost:3000/api/foods', and a 'Send' button. Below this, the 'Query' tab is active, showing a table for query parameters with columns 'parameter' and 'value'. The 'Response' tab is also visible, showing the JSON response from the API. The response status is '200 OK', the size is '1.5 KB', and the time is '12.43 s'. The JSON response is as follows:

```
[
  {
    "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"
    ]
  }
]
```

Day 3 - API Integration Report – Food Tuck



Challenges I Face:

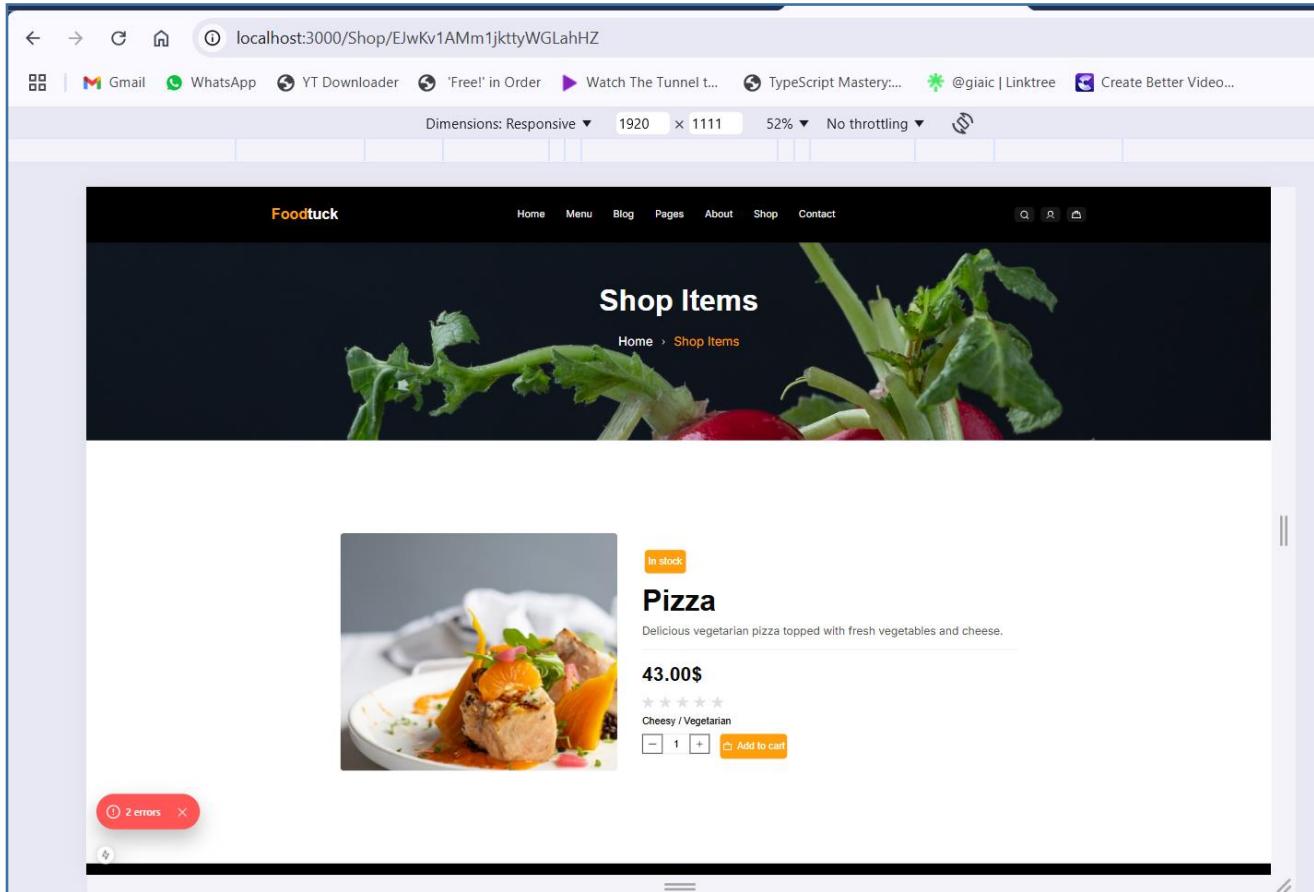
- Making Sure Mapping not get crash
- Making sure that fetching of data is correct
- Inspection of dynamic changes

Outputs:

- Chefs Data: Displayed dynamically on the "Our Chefs" page.
- Foods Data: Rendered as a menu card

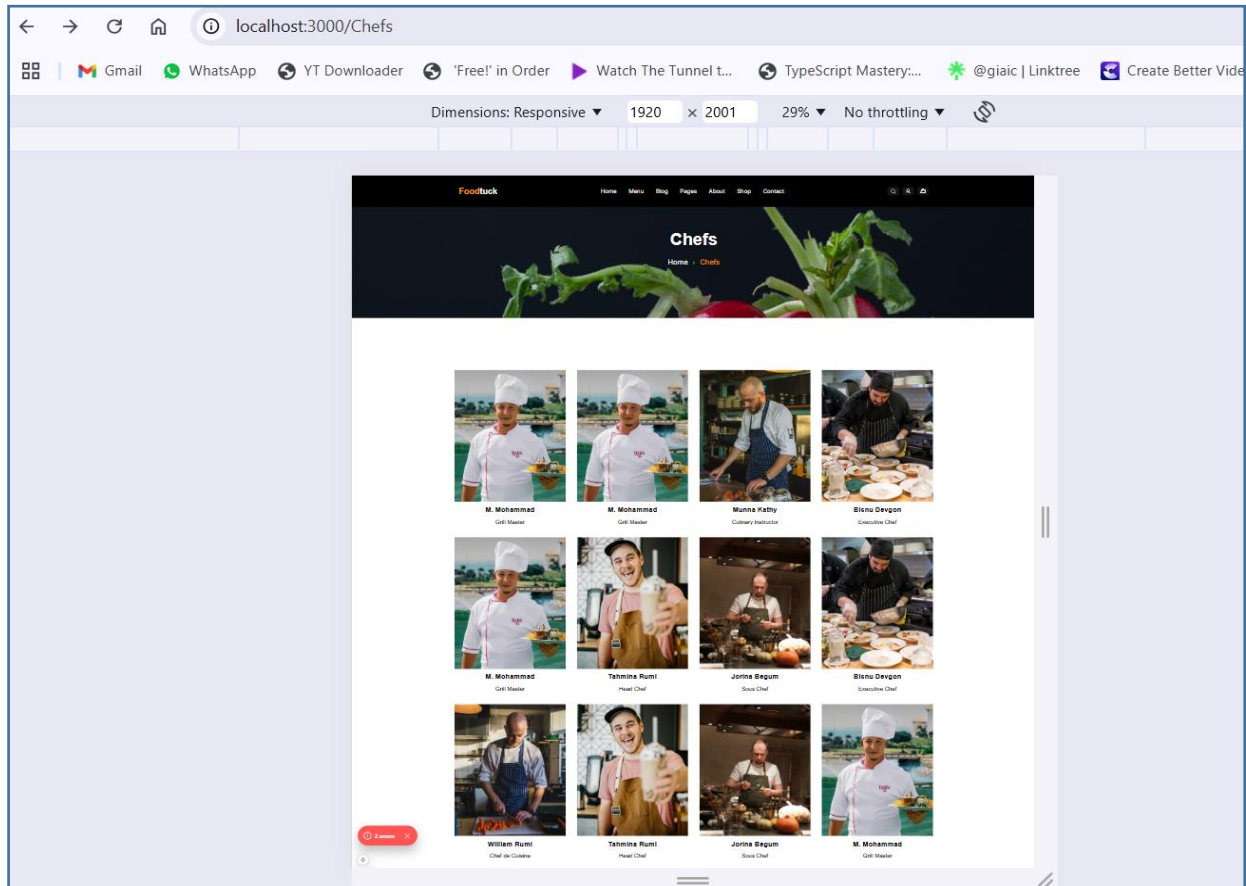
Day 3 - API Integration Report – Food Tuck

OUR SHOP(Dynamic Route):



Day 3 - API Integration Report – Food Tuck

CHEFS:



Day 3 - API Integration Report – Food Tuck

OUR SHOP

