Securin Project Documentation

Submitted By: Jagadeesh Chandra Duggirala
Jagadeesh.Duggirala22@st.niituniversity.in

1. Objective

The application of the Web App and APIs developed in this assignment is to parse a list of recipes stored in a .Json file format and store the data in a database. This data in the database would then be fetched using APIs and then displaying them on a webpage. The Web App and APIs are developed using JavaScript.

2. Task Overview

- Parsing JSON data
- Storing recipes in a database
- Creating API endpoints
- Implementing pagination, sorting, filtering
- Building a frontend to display recipe data

3. Provided Data


```
{
    "Contient": "North America",
     "Country_State": "US",
     "cuisine": "Southern Recipes",
     "title": "Sweet Potato Pie",
     "URL": "https://www.allrecipes.com/recipe/12142/sweet-potato-pie-i/",
     "rating": 4.8,
     "total_time": 115,
     "prep_time": 15,
     "cook_time": 100,
     "description": "Shared from a Southern recipe, this homemade sweet potato pie is easy to make with boiled sweet potato. Try it, it may just be the best you've ever tasted!",
     "ingredients": [
        "1 (1 pound) sweet potato, with skin",
        "0.5 cup butter, softened",
        "1 cup white sugar",
        "0.5 cup milk",
        "2 large eggs",
        "0.5 teaspoon ground nutmeg",
        "0.5 teaspoon ground cinnamon",
        "1 teaspoon vanilla extract",
        "1 (9 inch) unbaked pie crust"
     ],
     "instructions": [
        "Place whole sweet potato in pot and cover with water; bring to a boil. Boil until tender when pierced with a fork, 40 to 50 minutes.",
```

```
            "Preheat the oven to 350 degrees F (175 degrees C).",
            "Remove sweet potato from the pot and run under cold water. Remove and discard skin.",
            "Break sweet potato flesh apart and place in a bowl. Add butter and mix with an electric mixer until well combined. Add sugar, milk, eggs, nutmeg, cinnamon, and vanilla; beat on medium speed until mixture is smooth. Pour filling into unbaked pie crust.",
            "Bake in the preheated oven until a knife inserted in the center comes out clean, 55 to 60 minutes.",
            "Remove from the oven and let cool before serving."
        ],
        "nutrients": {
            "calories": "389 kcal",
            "carbohydrateContent": "48 g",
            "cholesterolContent": "78 mg",
            "fiberContent": "3 g",
            "proteinContent": "5 g",
            "saturatedFatContent": "10 g",
            "sodiumContent": "254 mg",
            "sugarContent": "28 g",
            "fatContent": "21 g",
            "unsaturatedFatContent": "0 g"
        },
        "serves": "8 servings"
    }
```


4. **Database Design**

   The Database chosen was neonDB which is a serverless postgres provider on the internet and database adapter for this is an ORM named Prisma.

5. **Handling NaN Values**

The json was iterated for each item and the data was extracted into each variable required. The numeric variables were then checked from NaN values using the isNaN() function in javascripts along with the if statements. The if condition when returned true would then change the value of the variable to “NULL”.

```
for (let i in jsonData) {
    console.log(i)
    let { cuisine, title, rating, total_time, prep_time, cook_time, description, nutrients, serves} = jsonData[i];
    if (!cuisine||!title||!description||!nutrients||!serves){
       continue;
    }
    if (isNaN(rating)) {
        rating = "NULL"
    }
    if (isNaN(total_time)) {
        total_time = "NULL"
    }
    if (isNaN(prep_time)) {
        prep_time = "NULL"
    }
    if (isNaN(cook_time)) {
        cook_time = "NULL"
    }
```

6. **API Development**

A total of 2 APIs were developed as requested in the problem statement PDF.

API Endpoint 1: Get All Recipes (Paginated and Sorted by Rating)

URL: /api/recipe/

- Method: GET
- Query Parameters:
- page: Page number for pagination (default is 1).
- limit: Number of recipes per page (default is 10).
- Response: A list of recipes sorted by rating in descending order.

Example Query:

GET /api/recipe?page=1&limit=10

Example Response:


```
{
  data: [
    {
      id: 1676,
      cuisine: 'Iced Tea',
      title: 'Copycat Sweet Iced Tea',
      rating: null,
      prepTime: 5,
      cookTime: null,
      totalTime: 15,
      description: 'Just like the sweet tea available at Chick-fil-A®, this copycat version is ultra refreshing and easy to make at home.',
```

```
    nutrients: [Object],
    serves: '4 servings',
    calories: null
  }
],
totalPages: 8244
        }
```


API Endpoint 2: Search Recipes

URL: /api/recipe/search

- Method: GET
- Query Parameters:
  - calories: Filter by calories (greater than, less than, or equal to a specific value).
  - title: Search by recipe title (partial match).
  - cuisine: Filter by cuisine.
  - total_time: Filter by total time (greater than, less than, or equal to a specific value).
  - rating: Filter by rating (greater than, less than, or equal to a specific value).

Example Query:

GET /api/searchRecipe?page=1&limit=1&q=&cuisine=Iced+Tea&calories=&totalTime=&rating=

Example Response:


```
{
  data: [
    {
      id: 1676,
      cuisine: 'Iced Tea',
      title: 'Copycat Sweet Iced Tea',
      rating: null,
      prepTime: 5,
      cookTime: null,
      totalTime: 15,
      description: 'Just like the sweet tea available at Chick-fil-A®, this copycat version is ultra refreshing and easy to make at home.',
      nutrients: [Object],
      serves: '4 servings',
      calories: null
    }
  ],
  totalPages: 8244
}
```

7. Files being submitted

These are the files that are being submitted along with this API documentation.

- ZIP File including code (Frontend & Backend)
- Screenshots of Frontend and API Testing on Postman
- API Document
- Document with steps on how to setup both the frontend and backend

8. Frontend (UI) Implementation

The frontend is developed using the Next.JS framework and the components are imported from shadcn's react components.

8.1 Table View

These are the columns that are being displayed on the frontend homepage.

- Title
- Cuisine
- Rating
- Total Time
- No of servings

8.2 Detail Drawer

The detail drawer is triggered upon clicking the specific row of an item. This detail drawer is imported from the shadcn repository. The details drawer displays the following details:

- Title
- Cuisine
- Rating
- Total Time (along with a tooltip which on hovering shows the cook time and prep time.
- Nutrition Table (Calories, Fat Content, Serving Size, Fiber content etc.)

8.3 Field Level Filters

There are multiple filters implemented in the api and the frontend. Those filters include:

- Title
- Cuisine
- Calories
- Total Time
- Rating

The UI has a input field of each of the filter where for the numeric fields the symbols must be used (e.g. > < >= <= =).

When there is a valid input in any of the filter fields the API Url changes to the /search one.

8.4 Pagination

The pagination for this assignment is implemented on the backend using Prisma's built-in pagination and for the UI the pagination buttons are imported from shadcn's repository.