Full-Stack JS engineer test assessment - the Recipe book

Overview

This documentation provides detailed instructions for completing the test assessment, which involves building two small applications to provide information about recipes. The application includes a Backend (BE) built with Node.js (Nest or Express) and a Frontend (FE) built with React (Next.js is a plus).

Project Overview

Backend:

Tech Stack:

- Node.js (Nest.js or Express.js)
- Typescript

Tasks:

1. **Endpoint: Get Available Recipes**
    a. Create an API endpoint, using Recipe API
    b. This endpoint should support prescription filtering:
        i. **List of All Available Recipes:**
        `https://www.themealdb.com/api/json/v1/1/search.php?s=`
        ii. **List of Recipes Filtered By Ingredient:**
        `www.themealdb.com/api/json/v1/1/filter.php?i=chicken_breast`
        iii. **List of Recipes Filtered By Country:**
        `www.themealdb.com/api/json/v1/1/filter.php?a=Canadian`
        iv. **List of Recipes Filtered By Category:**
        `www.themealdb.com/api/json/v1/1/filter.php?c=Seafood`
2. **Endpoint: Get Recipe Info**
    a. Create an API endpoint to retrieve detailed information about a specific recipe
    `https://www.themealdb.com/api/json/v1/1/lookup.php?i=52772`

Front-end:

Tech Stack:

- React.js
- Typescript
- Next.js (preferred but not mandatory)

Tasks:

1. **Recipe List Page**
    a. Display title based on the applied filter
    b. Display a list of recipes fetched from the endpoint (all recipes by default)
    c. Each recipes item should be clickable and navigate the user to the Recipes Info Page

2. **Recipe Info Page**
    a. Display detailed information about the selected recipe, including:
        i. **Recipe image:** Displayed prominently at the top-left
        ii. **Recipe name:** Displayed at the center
        iii. **Recipe country:** Displayed under the name, should be clickable and navigate the user to list of recipes (**Recipe List Page)** filtered by country
        iv. **Recipe Instructions:** Displayed below at the center
        v. **Recipe Ingredients:** Displays a list of ingredients, each of which is clickable and navigate the user to a list of recipes (**Recipe List Page)** filtered by ingredient
    b. Right sidebar:
        i. Display the list of recipes of the current recipe category
        ii. Should be clickable and navigate the user to list of recipes filtered by category

Additional Requirements

1. **Styling**:
    a. You can use any CSS framework or custom styles to design the components.
    b. Ensure that the UI is responsive and user-friendly.
2. **Environment Variables**:
    a. Create a `.env` file to store sensitive data such as API keys and base URLs.
    b. Ensure that environment variables are loaded and used securely in the application.
    c. Add `.env` to the repository.
3. **Code Quality**:
    a. Set up ESLint and Prettier to ensure consistent code formatting and quality.
    b. Ensure that all files are properly linted and formatted before submission.

4. **Documentation**:
    a. Include a `README.md` file that provides instructions on how to install, run, and test the application.
    b. Include any necessary setup steps, such as installing dependencies or configuring environment variables.

API Documentation

- **Recipe API:** Free Recipe API Support

Additional Instructions

We will be testing your application locally. Please ensure the following:

- **Separate Folders**: Place the frontend (FE) and backend (BE) code in separate folders within the root directory. Do not create a monorepo structure.
- **Parallel Execution**: Ensure that both the frontend and backend can be run simultaneously on different ports. The frontend should be able to communicate with the backend without any issues.
- **Instructions for Running**: Provide clear instructions in the README.md file on how to start both the frontend and backend servers, including any necessary environment variables or configurations.

By following these instructions, we will be able to test your application smoothly and verify that both parts work together as expected.