Recipe CRUD app

In this project, you will build a recipe-tracking app. The app will allow a user to add new recipes, display a list of recipes and delete a recipe from the list by clicking a **Delete** button. You will also be asked to add styling as provided in the mockup.

1:1

This project has starter code you can use for the application. You will need to add handlers and other logic to make this code work.
Note: *If downloading the assessment files to your local machine, make sure you're running Node v18 before you run* `npm install`.

1. *Check which version you are running:* `node -v`
2. *If needed, change the version to v18:* `nvm use v18`

For additional help, review the "Learn your tools: Visual Studio Code" lesson in the "Welcome" module.

Specific instructions

Find the `TODO` comments in the code and create the necessary functionality. Below is a list of specific items you will need to complete for this lesson.

- The app will display a recipe's name, cuisine, photo, ingredients, preparation instructions and action buttons (**Delete** and **Create**).
- You should create at least one additional component that is used by the `RecipeList` component.
- The app does not need to match the exact appearance of the mockup, but should be similar and pass the styling tests.
- You should add handlers and other attributes to the starter code as needed.

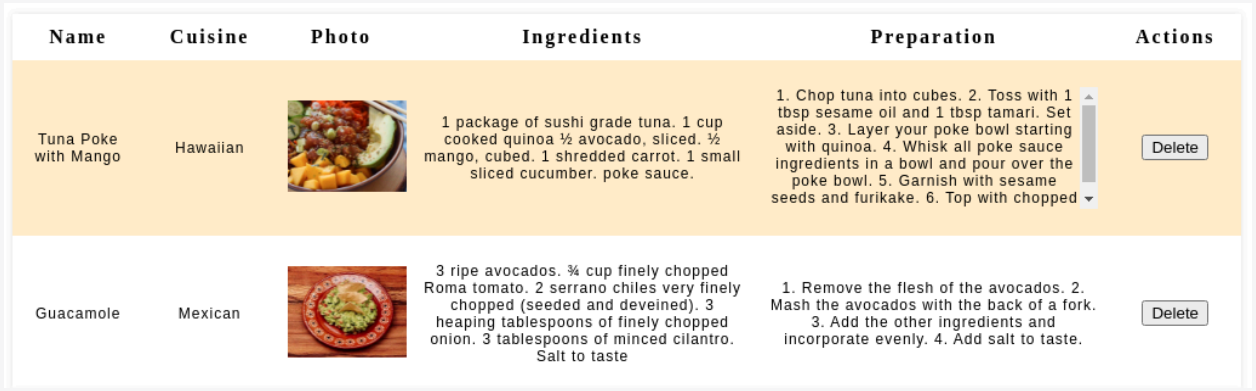
To create a recipe entry, your app will need to have a form with input fields for the name of the dish, the cuisine it belongs to, and an URL that points to a picture of the dish. Use `<textarea>` for the ingredients and preparation. For the tests to pass, use the following names for your inputs: `<input name="name">`, `<input name="cuisine">`, `<input name="photo">`, `<textarea name="ingredients">` and `<textarea name="preparation">`.

The new recipe must be added to the *end* of the list of recipes.

1:1

To read and display the list of recipes use the table structure that is provided in the starter code. Each recipe should display the name, cuisine, photo, ingredients, preparation and a **Delete** button as shown below:

1:1



Clicking the **Delete** button should remove the entire row/recipe from the list. For the tests to pass, make sure that the **Delete** button has `delete` as a name value (`name="delete"`). For example, `<button name="delete" onClick={deleteRecipe}>Delete</button>`.

Styling instructions

The `Delicious Food Recipes` text surrounded by an `h1` tag should use the `'Playfair Display SC'` font that has already been imported in `App.css`. It should also be `centered` and have a size of `64px`.

Read the documentation for nth-child. Use `nth-child` to set the width of the columns. It is suggested that you set the `width` for the *preparation* and *ingredients* columns to `30%`. For the rest of the columns, set the width to `10%`.

Use `nth-child(odd)` to set the table's *zebra striping* color pattern for the rows in `tbody`. The color in the mockup is `#fff0c7` but feel free to use a color of your preference that suits the design.

The *preparation* and *ingredient* columns should display a scrollbar if there is too much text. Use the predefined`content_td` class and `p` tag to wrap the text so that it uses a scrollbar if the text is too long (such as`<td className="content_td"><p>{(recipe.ingredients)}</p></td>`).

Use the object-fit property to `scale-down` the images and set the image `width` and `height` to `100%`.

Success criteria

- **Functionality:**
  - User can create a recipe entry.
  - User can read the list of recipes. The new recipe must be added to the *end* of the list of recipes.
  - User can delete a recipe.
  - CSS is used to make the app look like the mockup.
- **React code organization:**
  - Uses multiple components that play well together
  - Recipe data is contained in the state.
- **General code organization:**
  - Minimal code duplication