Angular Employee Management App

The main goal of this application is to load the contents of the employees.json file and have it displayed it an organized manner, while also developing basic functionalities like: **Search**, **Delete** and **Edit**.

Sizes, spaces, font sizes or other UI details are irrelevant. **Feel free to implement your own personal design** as long as it has a **pleasing look**, and all **technical requirements are met**.

Besides your main task explained above, you must also respect the following methodology requirements described below:

* **Use Routing** (e.g.: for Edit Functionality).
* Use a **Shared Module** (e.g.: for Dialog Component).
* Use an **Angular Service** to load *employees.json*.
* **Embed and use** one of the custom fonts (fond in *assets/fonts* folder)
* Use **SCSS** or other **CSS** preprocessors.
* **Do not use Boostrap, Angular Material** or other similar frameworks.
* The application should be **responsive** (work on different screens sizes).

STEP 1 Main Objectives:

Create a **New Angular** project (we recommend using Angular CLI, ng new command but is up to you to choose the best way to do it)

* Load the content of employees.json file, and map its contents to an interface.
* Display all employees in a list with control features: **Delete** end **Edit**.
* Disable **Edit** button if an employee has Blocked property set to *true*. (e.g.: **Reef Gamble**)
* Create separate components for employees list and its entries (e.g.: EmployeesList Component, and Employee Component).
* Use @Input and @Output to pass data between components.
* Use the images you find in *assets/images* for icons: **User**, **Delete** and **Edit** buttons.
* Sort employees list alphabetically, by FirstName property.
* Use Roboto font (found in *asses/fonts*)
* Add a simple transition animation for **Employee Components**.

STEP 2 Main Objectives:

Create a **Search** functionality that will filter results either by FirstName, SecondName or Position.

Use a Pipe to implement the Search functionality.

STEP 3 Main Objectives:



• Part of a Shared Module, create a ConfirmDialog Component that will ask for user confirmation when he tries to delete an employee.

• Use localStorage API to save the changes over initial employees list so on a page refresh the deleted employees will not be displayed.

• After the first usage of localStorage API, the app should load data from local storage. If the local storage is empty (all employees have been deleted), then the app should load data back from the JSON file.

STEP 4 Main Objectives:



• Create Edit functionality, on a different route. You should be able to edit employee’s Position property.

• Use localStorage API to save the edit changes over the initial employees list, so on a page refresh, the changes will be kept.

• Enable the Save button only if the text has changed and is valid (not empty).

• Display employee’s Details property in HTML format (it must take in consideration the <strong> or <em> tags).