MANAGEMENT APP CREATION FOR ANGULAR

- Specifications and Guidelines -

w w w . a c r e l e c . c o m

Contents

1 DOCUMENT SCOPE

The purpose of this document is to show you how to build an **employee management app using Angular 7**.

There are no specifications about sizes, spaces, font sizes or other UI details. The design we propose in this document serves as a **general guideline**.

Note: *We recommend you implement **your own design** as long as you keep a pleasant aesthetic and implement all the technical requirements.*

2 PROJECT PREREQUISITES

To create an employee management application using Angular, you need the following key requirements:

- An **Angular 7** boilerplate project.
- The `src/assets/employees.json` file, the `src/assets/fonts` folder, and its content. For more details about that, see **here**.
- Internet access

Note: *Remember to use the* `npm install` *command to install all node modules.*

Although access to internet is granted to you, you should still be able to **explain in detail** the general architecture and the concepts used in developing this test app.

IMPORTANT!!!
Total time allowed for this test: 3h.

3 PROJECT MAIN OBJECTIVE

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**.

3.1 Other objectives

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

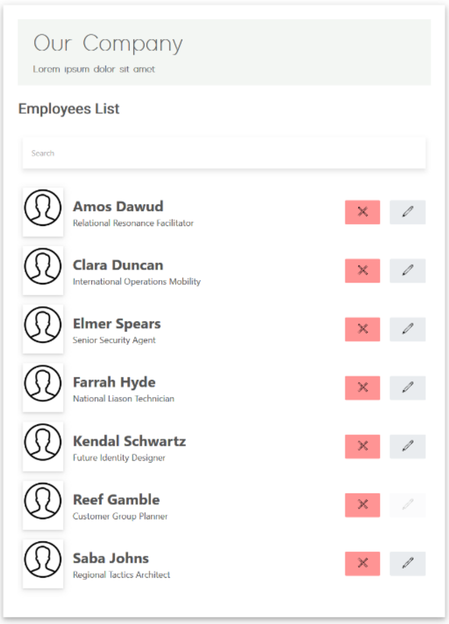
- **Use Routing** (e.g.: for Edit Functionality).
- Use a Lazy **Loaded Module** (e.g.: for Edit Functionality) and a **Shared Module** (e.g.: for Dialog Component or Highlight Directive).
- Use an **Angular Service** to load employees.json.
- **Embed and use** at least one of the custom fonts (fond here: src/assets/fonts)
- Use **SCSS** or other **CSS** preprocessors.
- **Do not use Boostrap, Angular Materials** or other similar frameworks.

- The application should be **responsive** (work on different screens sizes).

4 APPLICATION STATE 1: FULL LIST OF EMPLOYEES

4.1 Main Objectives

- 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**.
- Implement a disabled **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 src/assets/images for icons: **User**, **Delete** and **Edit** buttons.



4.2 Optional Objectives

- Sort employees list alphabetically, by FirstName property.
- Use `Acrelec` or `Roboto` fonts for the application's title (e.g.: **Our Company**) or other texts.
- Add a simple transition animation for **Employee Components**.

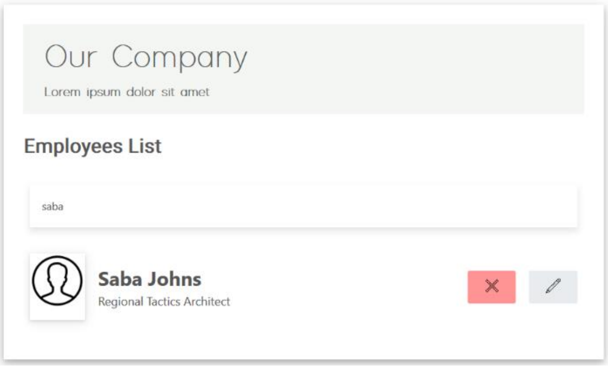
5 APPLICATION STATE 2: SEARCH FUNCTIONALITY

5.1 Main Objectives:

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

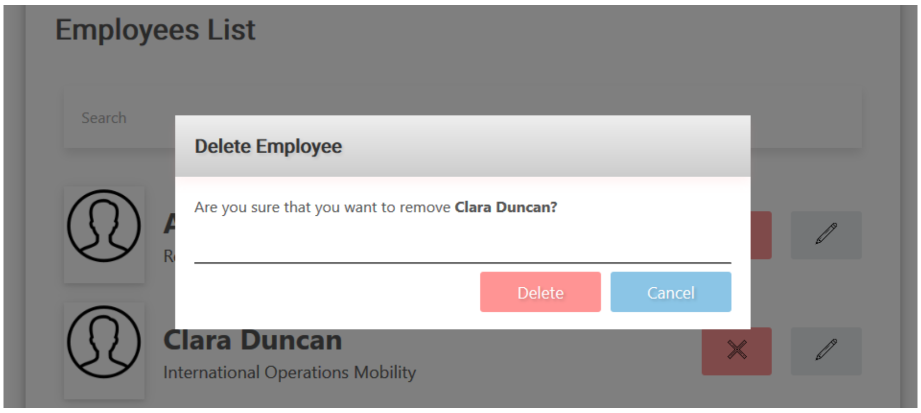
5.2 Optional Objective

Use a **Pipe** to implement the Search functionality.



6 APPLICATION STATE 3: DELETE FUNCTIONALITY

6.1 Main Objectives



Are you sure that you want to remove **Clara Duncan?**

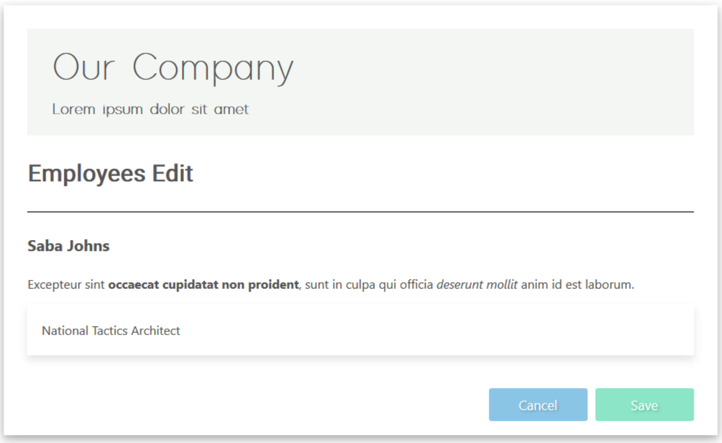
- Part of a Shared Module, create a ConfirmDialog Component that will **ask for user confirmation** when he tries to delete an employee.
- Use `localStorage API` to save the changes over initial employees list so on a page refresh the deleted employees will not be displayed.
- After the first usage of `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.

6.2 Optional Objectives

Add `ConfirmDialog` dynamically (e.g.: using `ComponentFactory`).

7 APPLICATION STATE 4: EDIT FUNCTIONALITY

7.1 Main Objectives


Our Company
Lorem ipsum dolor sit amet

Employees Edit

Saba Johns

Excepteur sint **occaecat cupidatat non proident**, sunt in culpa qui officia *deserunt mollit* anim id est laborum.

National Tactics Architect

- Create **Edit** functionality, on a different route, encapsulated in a Lazy Loaded Module: 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.

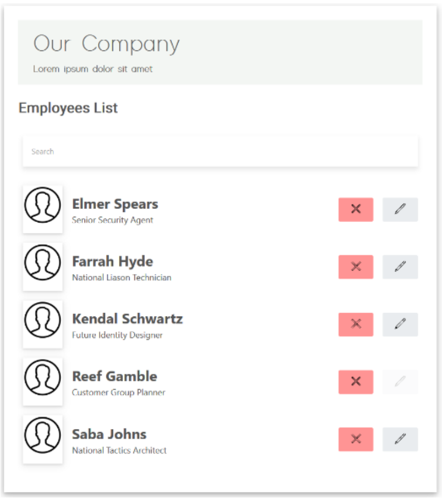
7.2 Optional Objectives

- 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).

8 APPLICATION STATE 5: FINAL LIST AFTER DELETE AND EDIT

8.1 Main Objectives

- Display the final list of employees after **Edit** and **Deletion** operations (see right).


Our Company
Lorem ipsum dolor sit amet

Employees List

Elmer Spears
Senior Security Agent

Farrah Hyde
National Liason Technician

Kendal Schwartz
Future Identity Designer

Reef Gamble
Customer Group Planner

Saba Johns
National Tactics Architect