**Problem Statement:**

Create an CRUD application for maintaining the employee details. Following are the requirements:

**Back-end:**

**1. There should be user registration feature on the website.  
  
- user should be able to register and login into the website  
- change and password features  
  
2. Create required models for storing the employee details  
  
 a. Each employee should have these basic fields  
 - Employee id , designation , date of joining   
 - name , address , phone number , email  
 b. The Model should not contain ANY duplicate data.  
 c. The Database should be populated with the data post setup of project (use fixtures)   
  
3. Create RESTful API for the same, should contain GET , POST , UPDATE , DELETE methods on the Model.**  
  
**4. Write a python script to generate the fixtures , the number of fixtures should be based on user’s input.**  
  
**5. Create a Periodic Task to update the employee details in the database from fixtures.**

**6. Write unit test cases for the models and views using the Django testing suite or pytest**  
  
7. Should be able to sort users based on user preference (by name , date of joining etc)  
  
8. Django administrator panel with search and bulk edit functionality on the Model Fields should be Implemented

Note:  
Improvement would be to create this entire functionality using ag-grid   
  
------- --------- --------- --------- -------- ------ ------ -------- -------- ------------- -------------- -------- ------ ------- ------- -

Front-end:

1. Create the following components:

a. Header (Navbar)

b. Dashboard

c. Create Record

Each component should have the following features:

a. Header Component:

Create a service file which will be used to fetch the available records from the JSON file and store it in a variable which will be used through out.

This component displays the following details:

i. Title as "Employee Management System"

ii. Total number of employee records.

iii. Two menu items namely "Dashboard" and "Create Record" which will navigate to Dashboard and Create component respectively. Implement routing for the same. By default Dashboard component should be loaded.

This data transfer between components should be done via BehaviourSubject.

b. Dashboard Component:

i. This component displays the available records in a table on load by getting the data which is passed using BehaviourSubject. Sort the data based on the Name in ascending order. Also differentiate the Cab Required column values (Yes - Green Color, No – Red Color) using angular directives.

Note: Don’t use any packages to render the table. Build the table from scratch.

ii. Table also should contain a column named “Delete” where each row in the table will have a “Delete” button so that the user can delete the record. Deleting a record should remove the record from the variable in the header component and the table should be updated and the count in the header should be updated.

iii. Develop a CUSTOM PIPE to achieve this functionality.

The table should also contain the search input above it through which the user can search the available records based on the ID only and the table should display only the filtered records. On clearing the search input, all the records should be shown again.

c. Create Record Component:

I. The user should be able to create a record by entering the above fields and also perform the validations as mentioned above. On click of “Add” button, the record should be added in the variable in the Header Component using BehaviourSubject and the count should be updated.