Assignment Project:

Name: Employee Manager

Type: UI/UX Development (**RESTful and JavaScript**)

Technology required: HTML, CSS, Javascript

Functional Prototype/ Wireframes: http://vmbbho.axshare.com

Interface Features:

- Graceful Degradation
- Responsive Design
    - Large, Tablets and Smartphones
- Browser Support:
    - IE9 +
    - FF, Chrome, Safari Latest
- **Theme:** Please use your design instinct to style the interface.

Use Cases:

Employee Management Screen

Employee Table/Grid:

1. The Grid list all the employees and the operations user can do on this listed employee information are –
    a. View More Employee Information.
    b. Edit Employee Information.
    c. Delete Employee Information

    These three operations can be performed from the icons on "Action" column in the grid.

2. There should be a visual cue for the grid rows where the employee state column is "inactive" (e.g. - Font style - italic and Font color - gray)

3. Clicking on header cell will sort the table using given particular cell data. Sorting will work A-Z and Z-A.

Employee Search:

1. There is a global search field to search any specific information from the Employee grid. The search filter should be for all the columns.

Add New Employee:

1. On clicking the "Add New" button on the top-right of the grid will open a modal pop-up where user can enter all the employee information and save.
2. All the input fields in the "Add New Employee" pop up need proper inline validation.
3. The profile picture can be dragged and dropped as well for upload.
4. On click "Save" button it will first validate the fields and then save the data. After that it should close the modal. The newly added Employee should appear in the grid in the top and it should also highlight the newly added Employee row for a short interval as a visual cue.

View More Employee Information:

1. The fist icon under the "Actions" column in the Employee grid is to view additional Employee information. On clicking the "View" icon, a flyout should slide open from right with some animation (Slide in Animation) and it shows additional user information.
2. The flyout panel should come on top of the main content without pushing the main content.
3. The "Slide out/Close" icon in the flyout should close it with some animation (Slide In Animation)
4. The "Pin" icon in the flyout should pin the flyout panel and push the main content. This "Pin" icon should work as a Toggler to pin /unpin the flyout panel. The pin state should persist even on browser refresh.

Edit Employee information:

1. The second icon under the "Actions" column in the Employee grid is to edit the Employee information. On clicking the "Edit" icon, the same modal should open with existing Employee information prepopulated. All the information can be edited / updated except the one or those which is/are the primary field(s) for the employee information record set.
2. Similar operations need to be carried out for "Save" and "Cancel" buttons.

Delete Employee information:

1. The third icon under the "Actions" column in the Employee grid is to delete the Employee information. On clicking the "delete" icon, it should ask for a confirmation with "yes" and "no" options in a pop up. "Yes" will remove the record from the grid and "No" will abort the operation.

Note:

1. The CUD (Create, Update and Delete) operations should show proper message after the operation being performed wherever necessary.
2. All the scenarios for content overflow needs to be handled properly.
3. Keyboard Navigation needs to be supported throughout

Development Guidelines:

General guidelines

- Assume, this is a huge project, "Employee Management" is just one module. You have another 40+ modules. Manage the project architecture and structure accordingly.
- Utilize build tools like Node JS, NPM, Grunt, Bower.
- Host your codes into GitHub or Azure
- Use Web Storage /JSON for data storage

JavaScript

- Use native JavaScript, OOP. No external/third party plugin
- Use of design pattern is recommended, use any pattern that you feel is most suitable for this project.
- Use of following JS libraries are allowed:
  - Modular Code - RequireJS, Common js for modularity
  - Data Binding - Underscore, Handlebars, or any template library
- DRY Principles
- DOM Manipulation e.g. jQuery, zepto is not recommended. Mostly write using plain JS.
- Think of High performance application, highly efficient in memory management, gc etc.
- Proper comments
  - Eg.

```
/* @name ModuleName
 * @description multiline
 * @params A What a does here
```

```
 * @returns Object
 */
```

CSS

- No use of CSS Framework. Every CSS should be written. Use of CSS Pre-processor (SASS/LESS) is recommended.
- Modularity
- Less specificity
- Highly optimized selectors and rules
- DRY Principles
- Very loosely coupled
- Proper comments

Non-Functional Requirements (NFR) Guidelines

- Page Load had to be managed.
- Asynchronous CSS, JS loading is recommended
- No FOUC,
- Symantec HTML
- Valid HTML5 (Exceptions can be accepted)
- Valid WCAG 2.0 AA standard
- Valid Section 508
- Prefer usability/accessibility rather pixel perfection.