OfficeRnD assignment

Summary

You need to create an application using Angular (whichever version you decide) which pulls data from a predefined set of endpoints and displays that data into a grid. Adding, removing, sorting and filtering of data should also be available in the application.

In detail

You need to follow these wireframes and apply the design to your application. The following functionality should be available

- **Fetch and display** - fetch information from the following APIs and display it according to the design above
- **Add member** - the user should be able to add members by clicking the Add member button on the top right. A dialog should appear (as shown in the wireframes) and the user should be able to fill the fields and save the data. Once done the member should appear in the grid
    - **Validation** - there should be a few fields in the dialog that are validated upon saving. The name, team and location are required, the start date should be after 01/01/2019
- **Delete members** - rows in the grid should be selectable. Once at least one row is selected the Delete button on the top right should be enabled and when clicked a confirmation for deletion should appear (styling of the confirmation dialog is up to you, it would be great if the design you chose complies with the one provided). When confirmed, the selected members should be deleted
- **Sorting** - the user should be able to sort the information in the grid by name, createdAt and company
- **Filtering** - the user should be able to filter the information in the grid by name, location and company. Additionally clicking the tabs on the top left should filter the members by status. Tab names represent valid status field values.
- **Virtualization** - the grid should support client-side virtualization and load only 20 items initially. When scrolling down it should load more items.

You don't need to deploy the application. Send us a link to a git repo where we can download and run locally. Please have a good structure for your code, files and folders. Keep your html, css and js neatly organised.
Except Angular there are no other frameworks that are required for this task. Use whatever helper tools/libraries you decide.

APIs and authentication

The following API endpoints should be used to retrieve data:

- **GET|POST /members** - an endpoint returning a set of member entities
- **GET /teams** - an endpoint returning a set of team entities
- **GET /offices** - an endpoint returning a set of office entities

The root url for each endpoint is:
https://staging.officernd.com/api/v1/organizations/assignment-demo

In order to get access to the afore-mentioned endpoint you need to add an '***Authorization'*** header with value ***'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjVkODM5MzJiZTU5YmU1MjcwNGMzMGZhOSIsImlhdCI6MTU2ODkwMzk3OSwiZXhwIjoxNjAwNDM5OTc5fQ.H4VdCyUQ4om4lhKHRqrcG09UCwV1jlpKEy_ixfuBRx4'***

Member entity

Each member entity has the following properties:

- **name** - the name of the member
- **email** - the email of the member
- **image** - url to an image that the member uploaded
- **createdAt** - the date/time the member has been created
- **team** - an _id of a team to which this member belongs
- **startDate** - the date the member started with the team
- **office** - the _id of an office(location) the member is located at
- **calculatedStatus** - status of the member, available values: ['active', 'drop_in', 'former', 'contact', 'lead']

Team entity

Each team entity has the following properties:

- **_id** - id of the team used in the member entity
- **name** - name of the team

Office entity

Each office entity (also called location) has the following properties:

- **_id** - id of the office (location) the member is at
- **name** - name of the office