1. MEAN - Angular-NodeJS - Team Tracker

Angular-NodeJS - Team Tracker

A company has separate team members for different technologies. Build an application which helps to maintain a tracker of those team members.

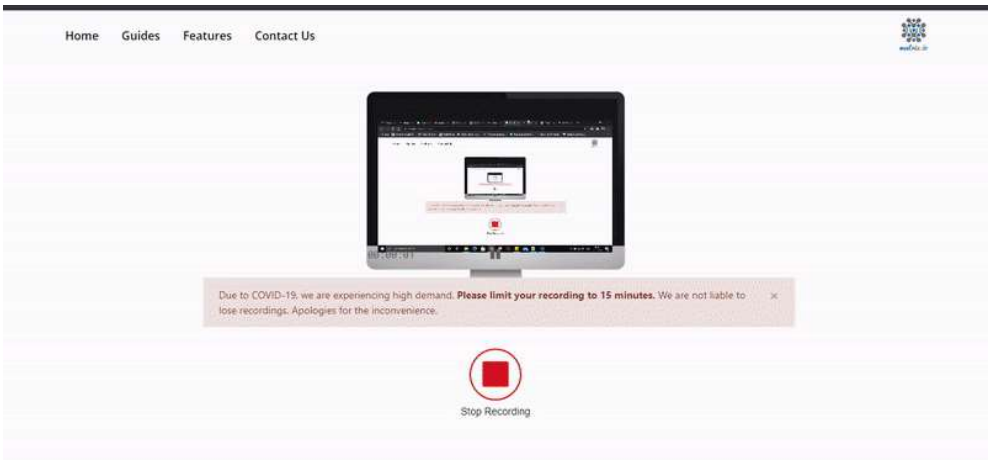
Components

Add member

This component should be used to add a member in a team (**team should be select from the dropdown list of options**) and add or remove a team name from the dropdown options.

Add Member Form [addMemberForm] : Form fields and its criteria
Employee ID : *[required], [min_value_10000], [max_value_300000]*
Employee Name : *[required], [pattern](atleast 3 characters, atmost 20 characters, allow alphabtes and spaces only)*
Experience : *[required], [min_value_0]*
Technology Name : *[required]*

Adding or removing options in dropdown:

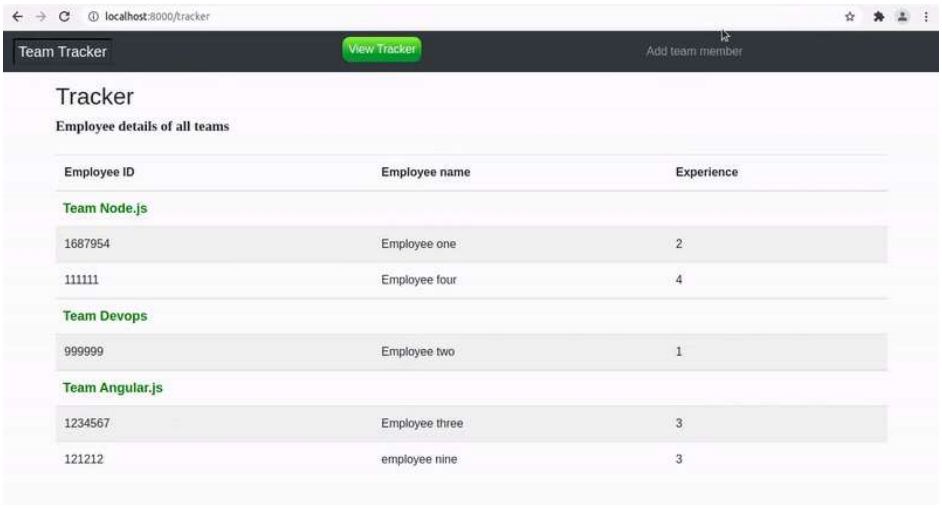


Tracker

This component should display the team and its member details.

- It displays the details of all members in different teams or technologies.
- Display the members separately respective to the team name.

Adding a member and display the member in the tracker:



Service

Tracker Requests Service

- This service is used to make API calls to the back-end NodeJS.
- The request URLs and the request methods to be done are present in the service file ***[angular/src/app/service/tracker-requests.service.ts]*.**
- **NodeJS** and **MongoDB** should be used as backend.

Back-end:

Develop an application that uses **Node.js** and **MongoDB** to manage teams.

Database: team-tracker

Collections: members, teams

Collections

There are two files for collections namely **members.js** and **teams.js** that should reside inside **src\mongoose\models**. The schema for those collections are given below

members				
name	**type**	**validations**	**required**	**default**
_id	ObjectID	Auto-generated	-	-
employee_id	Number	-	TRUE	-
employee_name	String	Should have at least 3 letters and can have only alphabets and spaces	TRUE	-
technology_name	String	-	TRUE	-
experience	Number	minium value = 0	TRUE	-
__v	Number	Auto-generated	-	-

teams					
name	**type**	**validations**	**required**	**default**	**u**
_id	ObjectID	Auto-generated	-	-	-
name	String	Should be unique	TRUE	-	-
__v	Number	Auto-generated	-	-	-

Routers:

There is a single file for routers namely **teams.js** that resides inside **src\routers**. The endpoints that each router will contain is given as follows.

1) ***/tracker/members/add*** **-> POST Method ->** This route should have to verify whether the employee already exists in the same team by using the **employee_id** and **technology_name** that comes with the request. If the data does not already exist, then you should add the data that comes with the request body to the **members** collection. If the **technology_name** doesn't exist, then you should add the data to the **teams** collection

Sample data comes with the request:

```
{
    "employee_id": 1213456,
    "employee_name": "Patlu",
    "technology_name": "Angular.js",
    "experience": 1
}
```

If data is stored successfully in members collection then you should send a response status code of **201**.

If something went wrong and the data have not been saved successfully then you should send a response status code of **400**.

2) ***/tracker/technologies/get*** **-> GET Method ->** This route should have to fetch all the data from the teams collections as the response.

If the data is fetched successfully from the database then you should send a response status code of **200.**

If the fetching was not successful then you should send a response status code of **400**.

Sample response:

```
[
    {
        "_id": "5fbf21afde58413b73e3d2b6",
        "name": "Node.js",
        "__v": 0
    },
    {
        "_id": "5fbf21afde58413b73e3d2b7",
        "name": "React",
        "__v": 0
    }
]
```

3) ***/tracker/technologies/add*** **-> POST Method ->** This route should have to save the technology name that comes with the request body to the teams collection.

Data sent with the request:

```
{
    "technology_name": "Java"
}
```

If the data is saved successfully to the teams collection, then you should send a response status code of **201**.

If something went wrong and the data is not stored successfully in the database then you should send a response status code of **400**.

4) ***/tracker/technologies/remove/:technology_name*** **-> DELETE Method ->** This route should have to remove the technology having a name that equals technology_name that comes with the request URL. While removing the team name(technology), it should also remove all the members associated with the respective team.

Sample Request: /tracker/technologies/remove/java

If the data was deleted successfully from the teams collection then you should send a response status code of **200**.

If something went wrong and data was not deleted successfully then you should send a response status code of **400**.

5) ***/tracker/members/display*** **-> GET Method ->** This route should have to fetch all the data from the members collection.

If the data is fetched successfully from the database then you should send a response status code of **200.**

If the fetching was not successful then you should send a response status code of **400**.

Sample Response:

```
[
    {
        "_id": "5fbf21afde58413b73e3d2b3",
        "employee_name": "Employee two",
        "employee_id": 999999,
        "technology_name": "Devops",
        "experience": 1,
        "__v": 0
    },
    {
        "_id": "5fbf21afde58413b73e3d2b4",
        "employee_name": "Employee three",
        "employee_id": 1234567,
        "technology_name": "Angular.js",
        "experience": 3,
        "__v": 0
    }
]
```

Note:

- ***Follow the instructions given in the code to complete the challenge***
- ***Click Run -> Test to set up your default data in the database.*** *(it will run the test cases as well as store some default data in backend DB)*

Instructions to install and run front end and back end in online IDE :

- To install project dependencies and run angular front-end server, Click **Run -> Install**. (If you are not getting **starting database mongodb** at the end of the process click the Install again)
  (If installation not ends with running the angular server or brings an error, click **Run -> Install** again or give the command in terminal to run angular in its root folder - **npm start**)
- The backend should run in **8001** port which is specified **angular/proxy.config.json.**
- To run the back-end server, click **Run -> Run**
  (For every change in your NodeJS back-end, you need to click **Run -> Run** or give the command in the terminal to run NodeJS in its root folder - **npm start**).
- Refresh the output. The Angular front-end changes will be reflected (re-run is not required).
- To run the test cases, click **Run Tests**.
- After completion, click **Submit Code**.

MongoDB commands:

- You can open the mongo shell by running ***mongo*** from the terminal.
- You can view all the data from the database in MongoDB by running ***show dbs*** from the mongo shell.
- You can select the database by running **use *team_tracker***.
- You can view the names of collections by running ***show collections***.
- You can view the data inside a collection by running ***db.collection_name.find().***
- Enter ***ctrl+c*** to exit.

Git Instructions

Use the following commands to work with this project

run Copy

```
bash run.sh
```

test Copy

```
bash test.sh
```

install Copy

```
bash install.sh; fuser -k 8000/tcp; npm start
```