1. TEAM TRACKER

ReactJS-NodeJS-Team Tracker

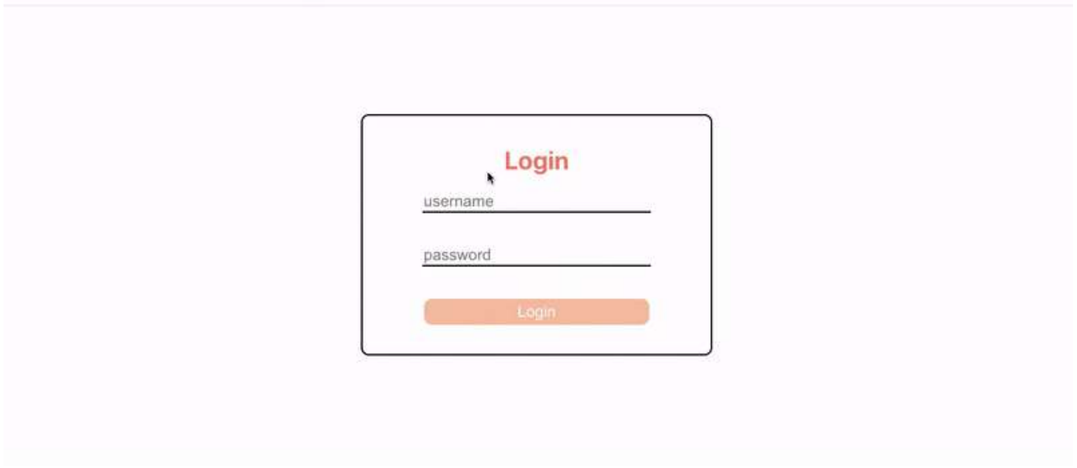
Instructions to Run Project

- Install the prerequisites required for your project by clicking **Run** > **Install**
- You can run your project by clicking **Run** > **Run.**
- For viewing your application, click the **Preview App.**
- **To** test your project, click **Run** >**Test** or use the **npm test** command in the terminal.

Frontend - ReactJS:

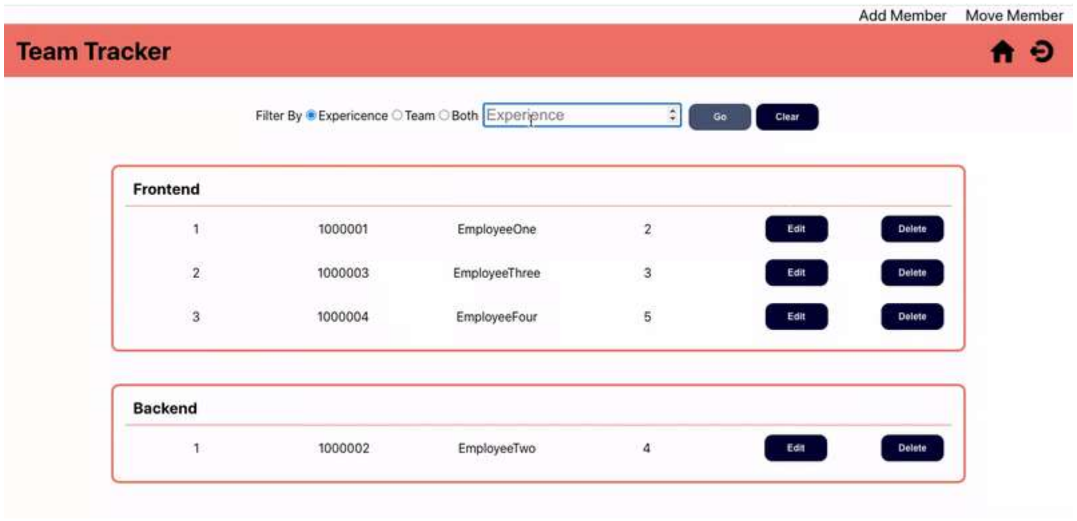
Login Page:

- Write code in src/Pages/Login.js.
- In this component, the user should be able to give inputs for name and password.
- Only when both input fields have valid values Login button should be enabled.
- If the credentials are invalid alert should be displayed with "Invalid Credentials".
- Credentials that to be used is username- **Admin**, password- **Fresco@123**.
- On successful login, the authentication token should be stored in local storage.
- On page loading, the local storage data should be cleared.



Home Page:

- Write code in src/Pages/Home.js
- This page will be the home page of the application.
- If the local storage does not have auth token, then the page should route back to the login page.
- The Home page should have the Header component.
- This page will have options to filter the team members, as shown in the screenshot. Also, it has teams component.
- '**Go**' button should be enabled only when an input field has valid input.
- On clicking the '**Go**' button, the filter should happen.
- On clicking the '**Clear**' button, all data teams should be displayed.



Frontend

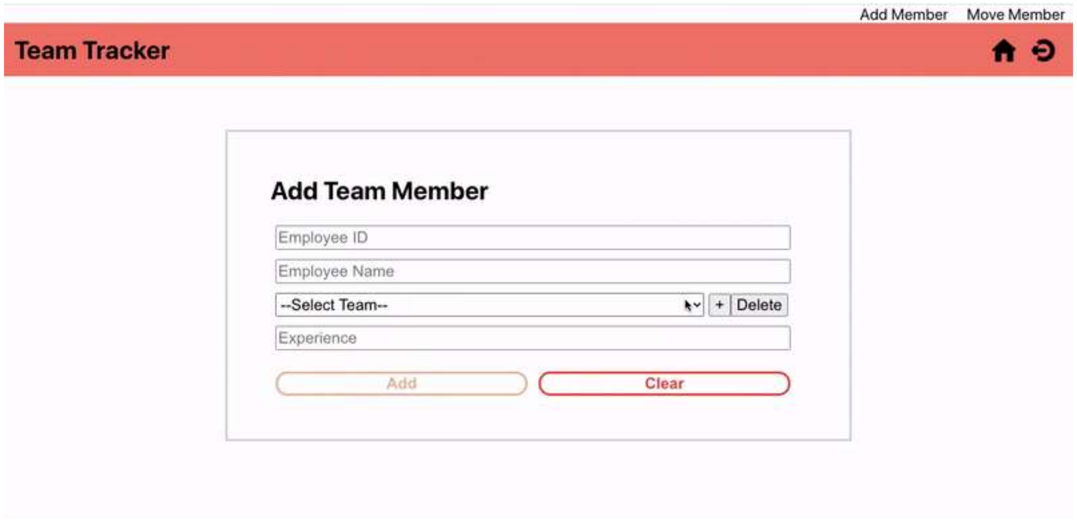
1	1000001	EmployeeOne	2	Edit	Delete
2	1000003	EmployeeThree	3	Edit	Delete
3	1000004	EmployeeFour	5	Edit	Delete

Backend

1	1000002	EmployeeTwo	4	Edit	Delete

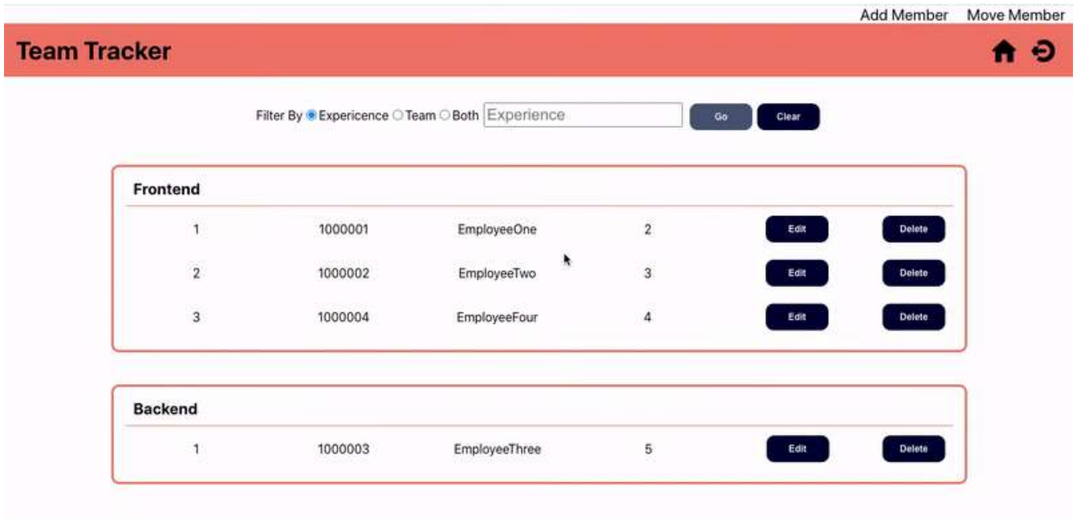
AddMember Page:

- Write code in src/Pages/AddMember.js
- If the local storage does not have auth token, then the page should route back to the login page.
- Using this page user should be able to add member, team technologies and delete the team technologies.
- Only when all input fields have valid values '**Add Member'** button should be enabled.
- If a member already exists in a team, an alert should be displayed with "Member already exist in the team which you are adding".
- Clicking the '+' button should display div that is used to add the team.
    - On clicking the '**save'** button, should save the team tech.
    - On clicking the '**cancel**' button, should hide the div.
    - If the team already exist, an alert should be displayed with "Team already exist".
- Clicking the '**Delete**' button should display the div with the team tech list with an '**x**' button which is used to delete the respective team tech.
    - On clicking the '**x**' button, should delete their respective team tech.
    - On clicking the '**cancel**' button, should hide the div.
    - If the team couldn't be deleted, an alert should be displayed with "Couldn't delete the team".



MoveMember Page:

- Write code in src/Pages/MoveMember.js
- If the local storage does not have auth token, then the page should route back to the login page.
- Using this page user should be able to move a member from one team to another team.
- Only when all input fields have valid values '**Move Member'** button should be enabled.
- On clicking '**Move Member'** the member should be moved.



Frontend

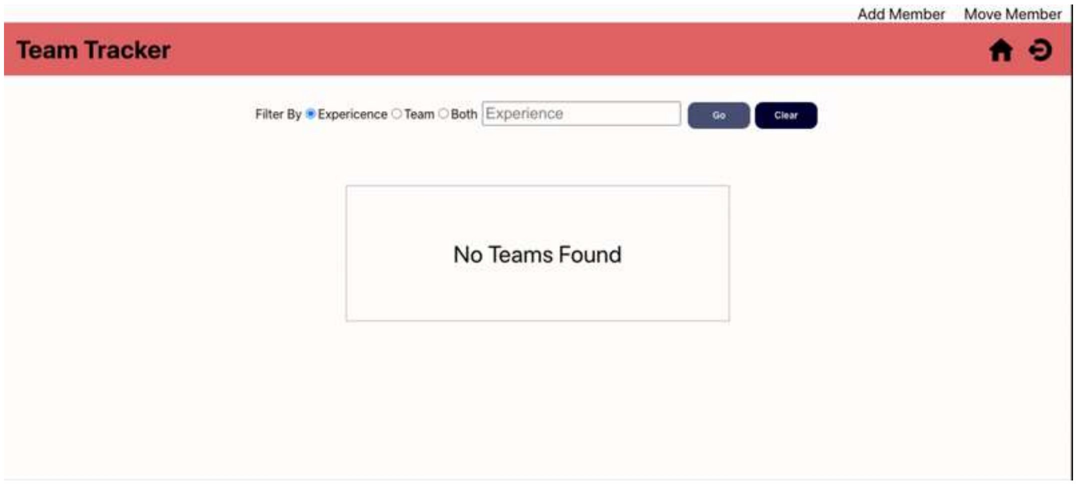
1	1000001	EmployeeOne	2	Edit	Delete
2	1000002	EmployeeTwo	3	Edit	Delete
3	1000004	EmployeeFour	4	Edit	Delete

Backend

1	1000003	EmployeeThree	5	Edit	Delete

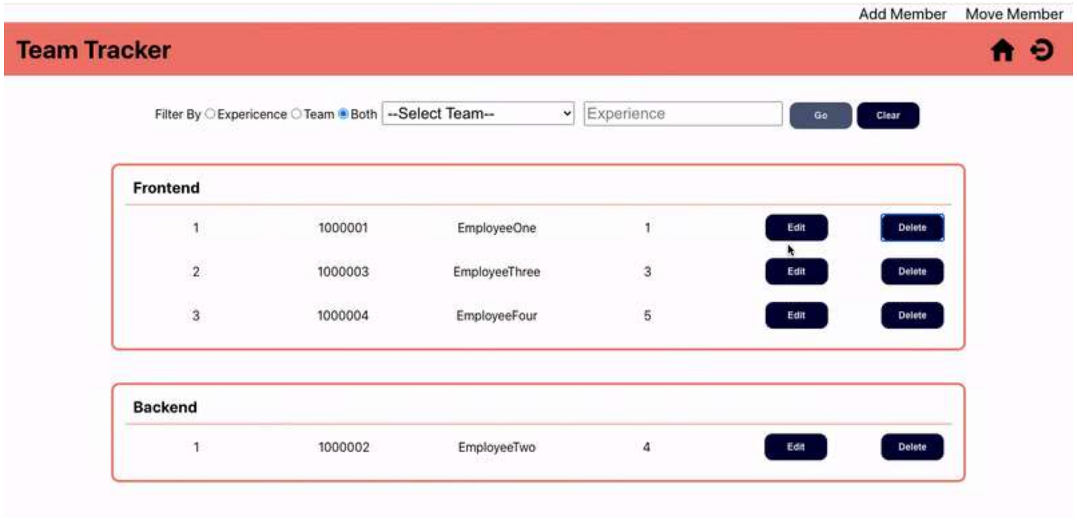
Header Component:

- Write code in src/Components/Header.js
- This component will have AddMember and MoveMember link which navigates to their respective pages.
- Also, this component has a home and a logout icon.
- The home icon should navigate to the home page.
- Clicking Logout icon should remove the token from local storage and navigate login page.



Teams Component:

- Write code in src/Components/teams.js
- This component should display the team members categorised based on teams in the table.
- Each member in the table should have the 'Edit' and 'Delete' button.
- Clicking the 'Delete' button should delete the respective team member.
- Clicking the 'Edit' button should enable the edit option.
- When the edit option is enabled, 'Done' and 'Cancel' button should be displayed, and the respective row should have input fields with their respective value.
- **Done** button should be enabled only when all input fields are valid.
- Clicking '**Done'** button should update the respective row's value if it is changed.
- Clicking the '**Cancel'** button should cancel the edit mode.



Frontend

1	1000001	EmployeeOne	1	Edit	Delete
2	1000003	EmployeeThree	3	Edit	Delete
3	1000004	EmployeeFour	5	Edit	Delete

Backend

1	1000002	EmployeeTwo	4	Edit	Delete

Validation:

The respective error message should be displayed for the appropriate reason.

	Error Message
Employee ID has empty input value or when employee id is not between 100000 and 3000000.	*Please enter a value *Employee ID is expected between 100000 and 3000000
Employee Name has empty input value or if the value is not between 100000 and 3000000 or if it doesn't have a minimum of 3 letters.	*Please enter a value Employee name can have only alphabets and spaces Employee Name should have at least 3 letters
Experience has empty input value.	*Please enter a value

NOTE: Make sure to use **/api** instead of **http://localhost:8001** to hit the backend server.

Backend - NodeJS:

- For back-end purposes, you have to make use of **NodeJS** and **MongoDB**.
- **team-tracker** is the name of the database that you will be using in this application.
- There are three collections, namely **admins**, **teams**, and **members** inside the database team-tracker.

Collections:

There are three files for collections, namely **admin.js**, **members.js**, and **teams.js** that reside inside **Backend/src/mongoose/models**. The schema of those collections is as follows.

admins				
Sl.No.	**field_name**	**type**	**validations**	**required**
1	_id	ObjectID	Auto-generated	-
2	name	String	-	TRUE
3	password	String	-	TRUE
4	tokens	Array of objects	each object should have the following properties -> token(JWT token)	-
5	__v	Number	Auto-generated	-

members				
Sl.No.	**field_name**	**type**	**validations**	**required**
1	_id	ObjectID	Auto-generated	-
2	employee_id	Number	min = 100000, max = 3000000	TRUE
3	employee_name	String	should contain only alphabets and spaces, the length should be greater than 2	TRUE
4	technology_name	String	-	TRUE
5	experience	Number	min = 0	TRUE
6	__v	Number	Auto-generated	-

teams					
Sl. No.	**field_name**	**type**	**validations**	**required**	
1	_id	ObjectID	Auto-generated	-	-
2	name	String	Should be unique	TRUE	-
3	__v	Number	Auto-generated	-	-

You should **set validations** for the fields in the collections, as mentioned in the tables above.

Middlewares:

There will be a single middleware file **auth.js** resides inside **Backend/src/middlewares,** that will verify the token created on logging in of the admin and will be used to provide access to the routes if the authentication is successful.

Routers:

There is a single router file **teams.js**, residing inside **Backend/src/routers** that manage all the app's endpoints. The endpoints and their functionalities are as follows.

You have to set **JWT token authentication** for the endpoints marked in red with **auth.js**. Use **_id** of logging in admin and **secret_token** given inside the helper.js file as the secret token to generate the JWT tokens.

1) */admin/login* -> POST Method -> This route should verify the credentials (name and password) of the admin that comes with the request body, and if valid, a JWT token should be generated and concatenated with the tokens array of the admin.

Data sent with the request:

```
{
  "name": "Admin",
  "password": "password"
}
```

- If the authentication was successful and the admin is valid, then you should send a response code of **200**.

- If the credentials sent with the request were incorrect, then you should send a status code of **400** with the following response.

Sample response:

```
{
  "error": "Username or password is wrong"
}
```

- If something goes wrong in executing the request, then you should send a response code of **400**.

2) ***/tracker/members/add*** **-> POST Method ->** This route should add the member details coming with the request body in the **members** collection. If the technology name that comes with the request body does not exist inside the **teams** collection, then you should add the technology name inside the **teams** collection.

Data sent with the request:

```
{
  "employee_id": 123456,
  "employee_name": "employee_name",
  "technology_name": "ReactJS",
  "experience": 2
}
```

- If the data is stored inside the **members** collection successfully, you should send a response code of **201**.
- If the data with employee_id and technology_name that comes with the request body already present inside the **members** collection, then you should send a response code of **400** with the following response.

```
{
  "error": "Member with same team already exists"
}
```

- If something goes wrong, then you should send a response code **400.**

3) ***/tracker/technologies/get*** **-> Get Method ->** This route should fetch all the data from the **teams** collection.

Sample response:

```
[
  {
    "name": "ReactJS"
  },
  {
    "name": "NodeJS"
  },
  {
    "name": "Angular"
  }
]
```

- If the data is fetched successfully from the **teams** collection, you should send a response code of **200**.
- If something went wrong, then you should send a response code of **400**.

4) ***/tracker/technologies/add*** **-> POST Method ->** This route should add the name of the technology that comes with the request body inside the **teams** collection.

Data sent with the request:

```
{
  "name": "AI"
}
```

- If the data sent with the request is stored successfully inside the **teams** collection, you should send a response code of **201**.
- If something went wrong, then you should send a response code of **400**.

5) ***/tracker/technologies/add*** **-> DELETE Method ->** This route should delete the data having name equals the technology_name that comes with the request body from the **teams** collection.

Data sent with the request:

```
{
  "name": "Java"
}
```

- If the deletion was successful, then you should send a response code of **200**.
- If something went wrong, then you should send a response code of **400**.

6) ***/tracker/members/update/:id*** **-> PATCH Method ->** This route should update the details of the data having _id in the **members** collection equal to id that comes along with the request URL with the data that comes with the request body.

Sample request: /tracker/members/update/5efaeef59ce58936c44d9cd0

Data sent with the request:

```
{
  "employee_id": 546231
}
```

- If the update was successful, then you should send a response code of **200**.
- If something went wrong, then you should send a response code of **400**.

7) ***/tracker/members/display?tech=technology_name&experience=experience*** **-> GET Method ->** This route should fetch all the members from the **members** collection. There are two optional parameters that can be sent with the request, namely tech and experience. If some value is passed with the parameters, then you should fetch the data accordingly (i.e.) If any technology name is sent with the tech param, then you should fetch the members having technology_name equal to the value of tech param. If some value is sent with the experience param, then you should fetch the members having their experience equal to or greater than the value of the experience param.

Sample request 1: /tracker/members/display

Sample response 1:

```
[
  {
    "employee_id": 123456,
    "employee_name": "employee_one",
    "technology_name": "ReactJS",
    "experience": 2
  },
  {
    "employee_id": 542613,
    "employee_name": "employee_two",
    "technology_name": "NodeJS",
    "experience": 1
  },
  {
    "employee_id": 685544,
    "employee_name": "employee_three",
    "technology_name": "Angular",
    "experience": 3
  }
]
```

Sample request 2: /tracker/members/dispaly?tech=NodeJS

Sample response 2:

```
[
  {
    "employee_id": 123456,
    "employee_name": "employee_one",
    "technology_name": "NodeJS",
    "experience": 2
  },
  {
    "employee_id": 542613,
    "employee_name": "employee_two",
    "technology_name": "NodeJS",
    "experience": 1
  }
]
```

Sample request 3: /tracker/members/dispaly?experience=3

Sample response 3:

```
[
  {
    "employee_id": 123456,
    "employee_name": "employee_one",
    "technology_name": "NodeJS",
    "experience": 3
  },
  {
    "employee_id": 542613,
    "employee_name": "employee_two",
    "technology_name": "ReactJS",
    "experience": 4
  }
]
```

Sample request 4: /tracker/members/dispaly?tech=Angular&experience=2

Sample response 4:

```
[
  {
    "employee_id": 123456,
    "employee_name": "employee_one",
    "technology_name": "Angular",
    "experience": 3
  },
  {
    "employee_id": 542613,
    "employee_name": "employee_two",
    "technology_name": "Angular",
    "experience": 2
  }
]
```

- If the data is fetched successfully from the collection, then you should send a response code of **200**.
- If something goes wrong in fetching the data, then you should send a response code of **400**.

8) ***/tracker/members/delete/:id*** **-> DELETE Method ->** This route should delete the data with _id in the **members** collection that equals id that comes with the request URL.

Sample request: /tracker/members/delete/5efaeef59ce58936c44d9cc1

- If the data is deleted successfully from the **members** collection, then you should send a response code of **200**.
- If something goes wrong in deleting the data, then you should send a response code of **400**.

MongoDB Commands:

- You can open the mongo shell by running ***mongo*** from the terminal.
- You can view all the databases 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 the collections by running ***show collections***.
- You can view the data inside a collection by running ***db.collection_name.find()*****.**
- Press ***ctrl+c*** to exit.