**Integrated Assignment**

**Developing a Single Page Application using React**

**Duration: 3Hrs Marks:50**

**Instructions to use the executable jar file provided:**

* Download the **“To\_Trainee”** folder using the **Download** button on your Hands On Exam client and save it in your local system
* Inside **PickAMeetingApp\_BackEnd** folder **“StartBackEnd.bat**”batch file is provided, double click the same to run the back-end web service.
* To ensure that the backend is running successfully, open **Postman** and test for below Data making a post request

**url:** <http://localhost:8765/meeting/schedule/>

**{**

**"schedulerName":"John",**

**"dl":"ETA\_MYS\_JAVA",**

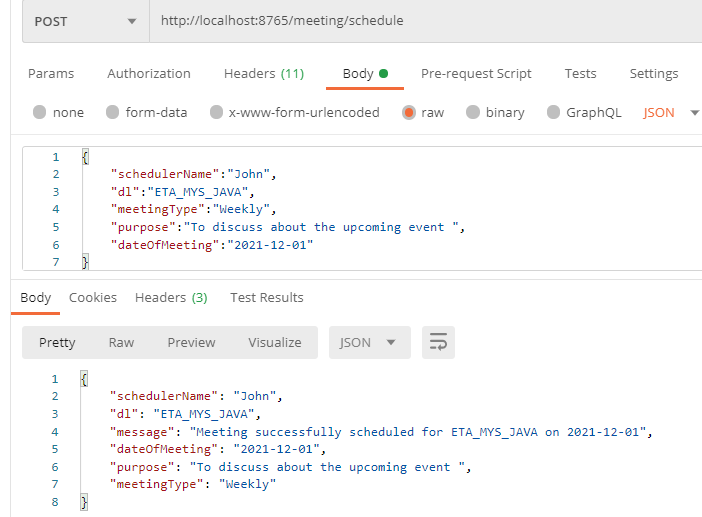
**"meetingType":"Weekly",**

**"purpose":"To discuss about the upcoming event ",**

**"dateOfMeeting":"2021-12-01"**

**}**

**Check below screenshot for same**



**Instructions to use the FrontEnd folder provided:**

* Open **PickAMeetingApp\_FrontEnd in** Visual studio code.
* Execute below **npm** command in to **PickAMeetingApp\_FrontEnd** project folder.
* **npm config set registry https://infyartifactory.ad.infosys.com/artifactory/api/npm/npm/**
* **npm install**

**Backup, Verification and Submission guidelines:**

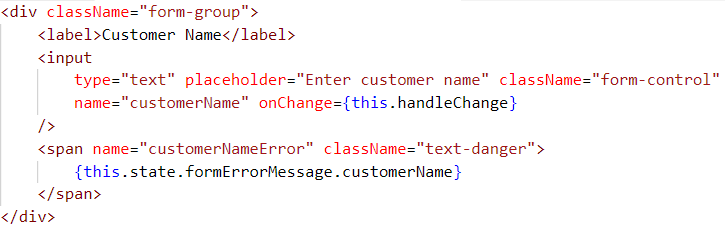
* Inside the **PickAMeetingApp\_FrontEnd** folder “**StartVerification.bat”** batch file is provided, double click the same to verify your project.
* This will execute the jar file and open **localhost:1729** in a web browser.
* Select the “**I Agree**” checkbox and then click on “**Verify”** button to Verify the projectand **“Copy Scripts to Desktop”** button **for back up**
* Enter your **employee id** correctly and click on “**Copy Scripts to Desktop**” button.
* This will create a folder on your desktop with the name as your **[employee id]** andcopies only **src**. Use this employee id folder to take **backup** and **Submission** from hands-on client.

Note: There should not be any space in the workspace folder name.

**FrontEnd Coding Guidelines:**

* **DO NOT** alter the function name or the argument list of the function that is provided to you
* **DO NOT** add any new function.
* **DO NOT** write codes that result in infinite loops.

Example:

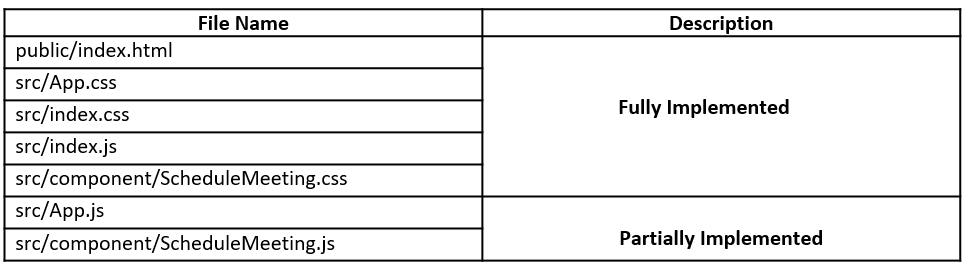


**Problem description**

**PickAMeetingApp** is an online portal for scheduling meeting. **It includes following module:**

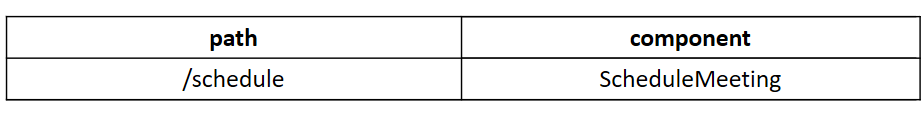
**1.Schedule Meeting.**

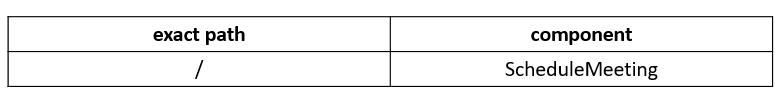
The **client side** need to be developed, for interacting with the server side, which has been already implemented.

****

**src/App.js: (to be implemented)**

* Import all the necessary modules for implementing the routing.
* Configure the Route as shown below

****

****

* When the application loads the following view must be displayed:



**src/component/ScheduleMeeting.js: (to be implemented)**

All the **states** for the **ScheduleMeeting** are already defined as given below:

* **formValue:** It should store the data entered by the user in Schedule Meeting Form
* **formErrorMessage:** It should store error message for each input field when validations fails
* **formValid:** It should store the validity of each input field
* **errorMessage:** It should store error message
* **successMessage:** It should store success message

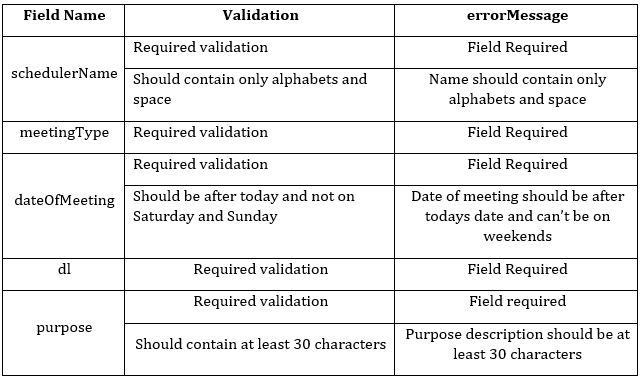
**Note: Do not add any new states or alter the given states**

The **methods** in **ScheduleMeeting** component are as given below:

* **handleChange(): (to be implemented)**
  + This method takes event as a parameter and sets the value entered by the user to the corresponding state of **formValue** object
  + After setting the state value, it should invoke **validateField ()** method to check whether the entered value is valid according to the validations mentioned or not

**Note: Do not return any value from this method**

* **validateField(): (to be implemented)**
  + This method should take name of the field and the corresponding value entered to validate it based on the given requirements
  + If value entered is invalid, it should set appropriate error message to state of corresponding field in **formErrorMessage** object and corresponding state in **formValid** objectshould be set as **false**
  + If valid input is entered, the corresponding state in **formValid** object should be set to **true** and the **errorMessage** (if any) for that field should disappear
  + Fields, validation and their validation messages are given as below:



* + **Note:** The **Schedule** button should be disabled until all the fields have valid input. The **buttonActive** state of **formValid** object should be changed as per requirements to either disable or enable the **Schedule** button
  + While setting formErrorMessage and formValid objects to state, the previous states of successMessage and errorMessage should be cleared

**Note: Do not return any value from this method.**

* **handleSubmit (): (to be implemented)**
  + This method should take event as a parameter.
  + It should prevent the page reload on form submission and then invoke **scheduleMeeting()**

**Note: Do not return any value from this method.**

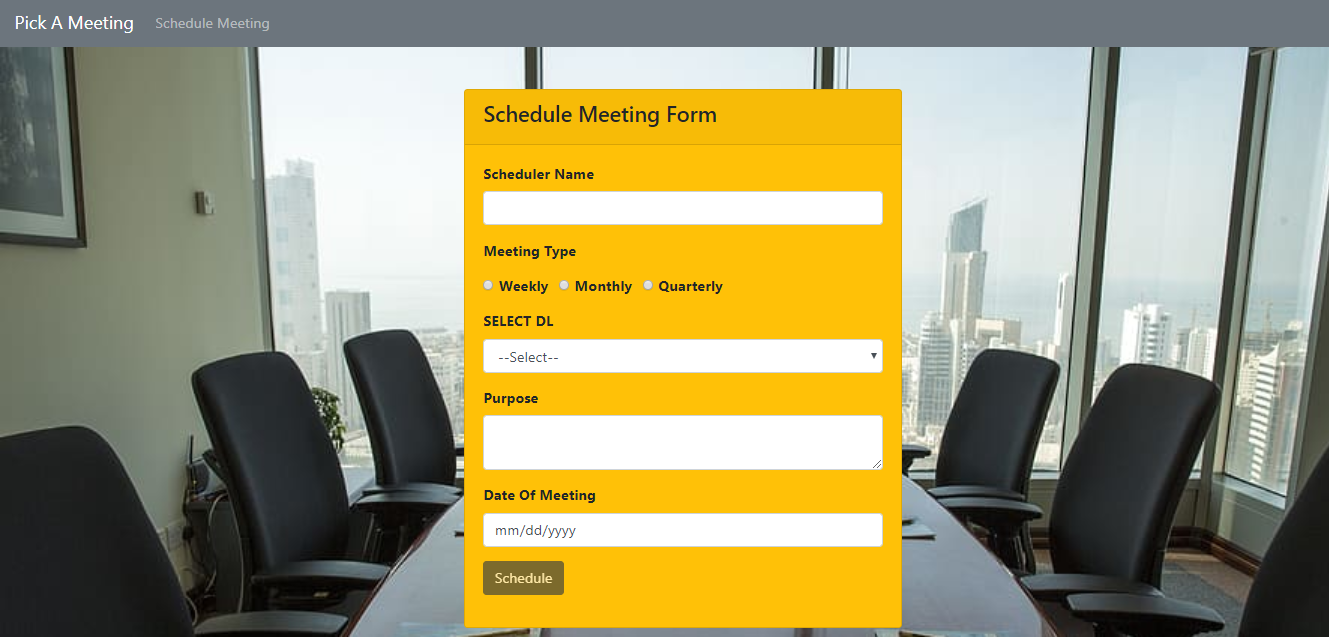
* **scheduleMeeting (): (to be implemented)**
  + It should reset the **errorMessage** and **successMessage** states with appropriate values.
  + It should send an **AXIOS** **Post** request to the constant **scheduleMeetingURL** givenat top of component bypassing **formValue** through body of the request, which in turn returns a message on successful scheduling of meeting.
  + If the request is successful, it should set the **successMessage** state with message of response data.
  + If the request fails, it should check if error response is defined,
    - If error response is defined, it should set **errorMessage** with message of the error response data.
    - Else, it should set the **errorMessage** state as **“Please run the backend”.**

**Note: If the request is successful, the errorMessage (if any) should disappear and VICEVERSA**

**Note: Do not return any value from this method.**

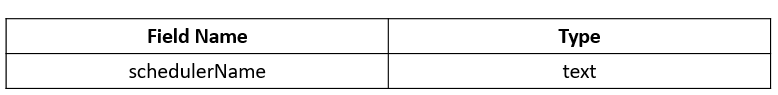
**JSX of ScheduleMeeting:**

* The **JSX** of the **ScheduleMeeting** component should be coded to render the below view:

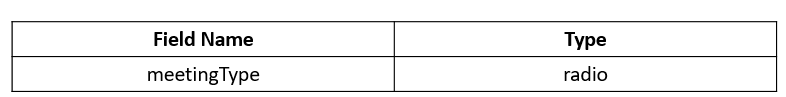
****

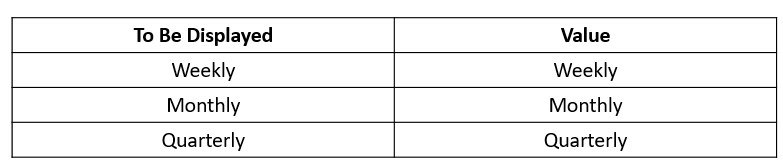
* Form fields and their types:

1. Scheduler Name

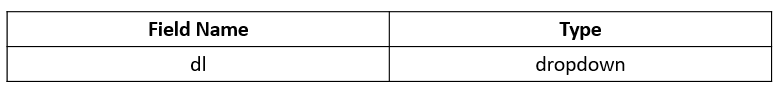


1. Meeting Type



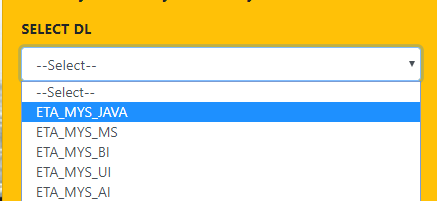


1. SELECT DL

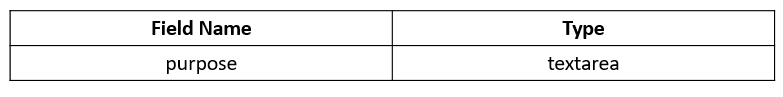


* The **dl** dropdown should be populated with the values of attribute **distributionList** of formValue state objectwhich is already provided.
* First option must be --Select-- with value as empty string.

The dropdown should be populated as shown below.

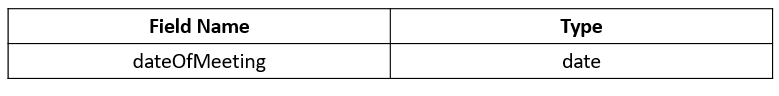


1. Purpose



* Textarea with 2 rows and 30 columns

1. Date of Meeting



**Note:**

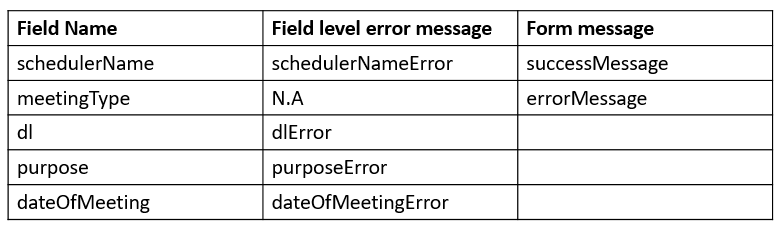
* **Schedule** button should be disabled till valid inputs are given in all the input fields
* The value of name attribute for the **Schedule** button should be **scheduleMeeting**
* Once valid data is entered into the input fields, the error message of the field should disappear
* USE APPROPRIATE BOOTSTRAP CLASSES WHEREVER NECESSARY

**Note:** Use css class **text-danger** provided in **ScheduleMeeting.css** for displaying error message of each input field and form error message.

**Note:** Use css class **text**-**success** provided in **ScheduleMeeting.css** for displaying form success message.

* Useonly **span** tag to display error messages wherever required
* **handleChange ()** method must be invoked every time when the value changes in any of the input fields

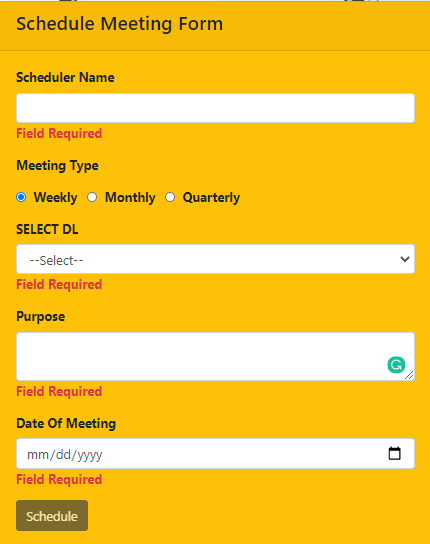
**Do not forget to specify the value for name attributes in respective fields as given below:**

****

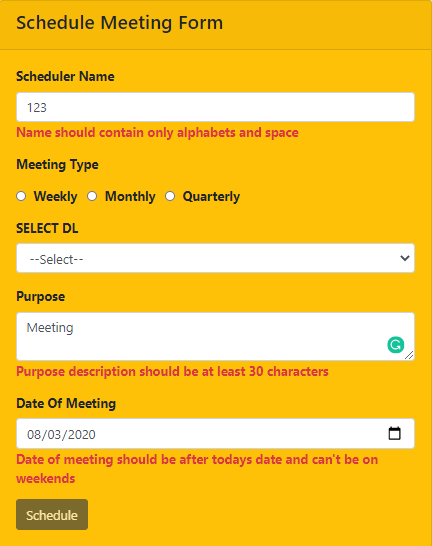
The error messages for the input fields should be displayed as shown below:

Schedule button should be disabled as shown below until all the fields are valid.

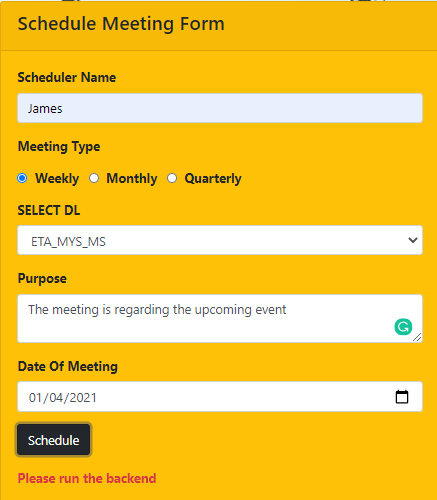
**Required validations:**



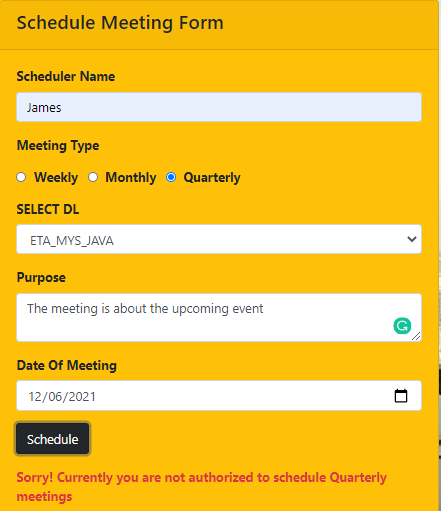
**Other Validations:**



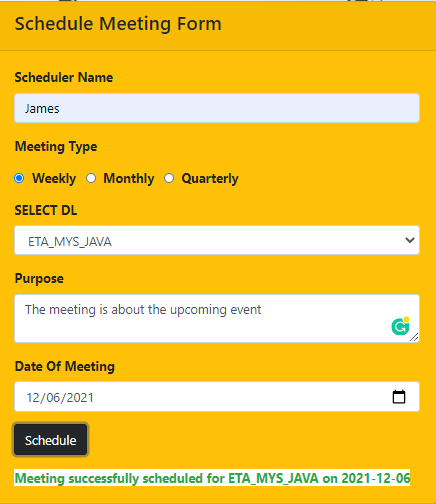
If the **Schedule** button is clicked while the server (backend Project) is not running, it should display error message as shown below:



If the server is running and Meeting Type is selected as Quarterly and valid inputs are provided in all the fields, it should display error message as shown below:



If the server is running and valid inputs are provided in all the fields, it should display success message as shown below:



**~~~~~~~~~~~~~~~~~~~All The Best~~~~~~~~~~~~~~~~~~**