**WeCare**is an online Life Coaching application that helps its users to sign up and log in to seek the guidance of famous Life Coaches. Users can search for a Life Coach based on specialty and can book an appointment within 7 days. They can also see upcoming appointments and can reschedule or cancel the appointments. Similarly, Life Coaches can also sign up and login and can see their upcoming schedule.

**List of Softwares Required:**

* **Node JS**
* **Angular**
* **Visual Studio Code IDE**

Note: Follow the steps mentioned at the beginning of the course to install the required softwares.

**Routing:**

|  |  |  |
| --- | --- | --- |
| **Route Path** | **Child Path** | **Component** |
| /home |  | HomeComponent |
| /login/:role |  | LoginComponent |
| /signup/:role |  | SignupComponent |
| /user | /home | UserhomeComponent |
| /profile | UserprofileComponent |
| /appointments | AppointmentsComponent |
| /coach | /home | CoachhomeComponent |
| /profile | CoachprofileComponent |

**UserhomeComponent**, **UserprofileComponent** and **AppointmentsComponent** are children of **UserComponent**

**CoachhomeComponent** and **CoachprofileComponent** are children of **CoachComponent**

NB: Create a constants.ts in assets folder to store all the string messages in the application

**Home Page:**

This should be coded in **HomeComponent**. The below view should be displayed.



**SignUp Page:**

This should be coded in SignupComponent.

**Properties:**

* role: string;
* signUpTitle: string;
* signUpImg: string;
* formDisplay = true;
* errorMessage = '';
* userregisterForm: FormGroup;
* coachregisterForm: FormGroup;
* userId;
* coachId;

The form control names under form group and corresponding validations are as shown below:

**userregisterFom**:

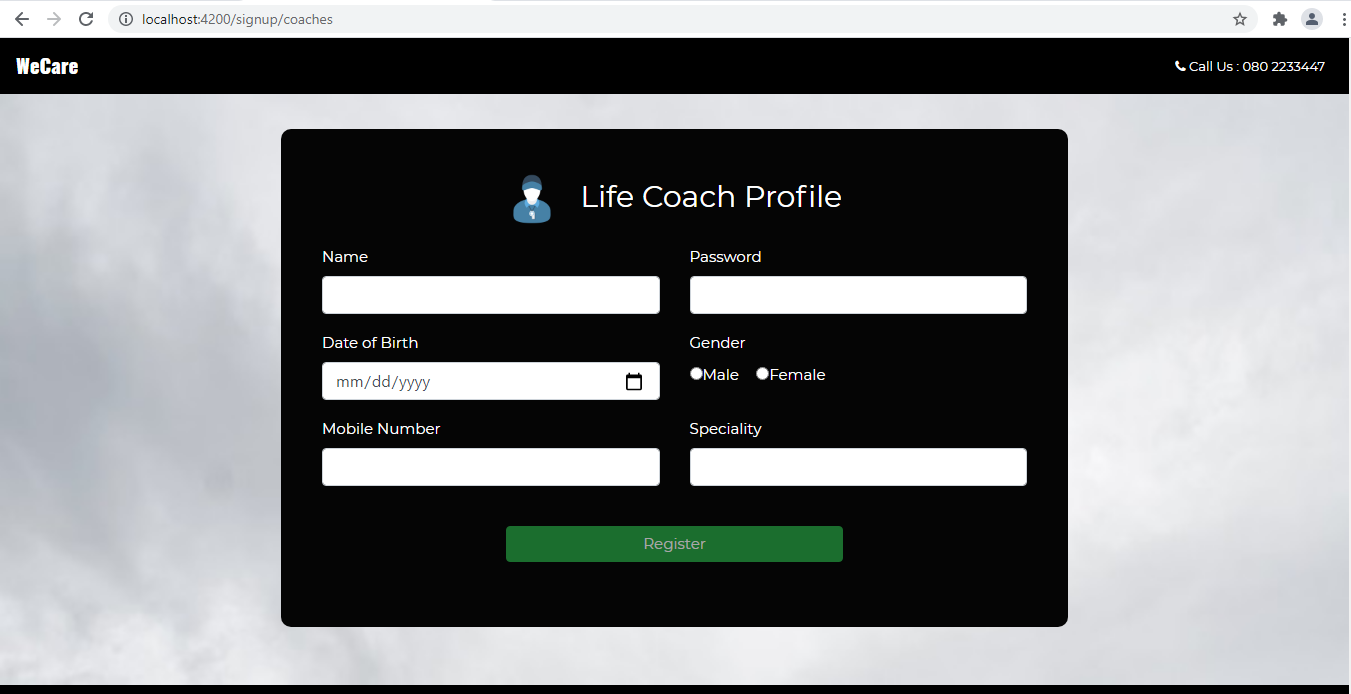
| **FormControlName** | **Validations** |
| --- | --- |
| name | The name should have 3 to 50 characters |
| password | Password should have 5 to 10 characters |
| mobileNumber | Mobile Number should have 10 digits |
| email | Required |
| dateOfBirth | Age should be between 20 and 100 years |
| gender | Required |
| pincode | Pincode should have 6 digits |
| city | The city should have 6 to 20 characters |
| state | The State should have 6 to 20 characters |
| country | The Country should have 6 to 20 characters |

**coachregisterForm**:

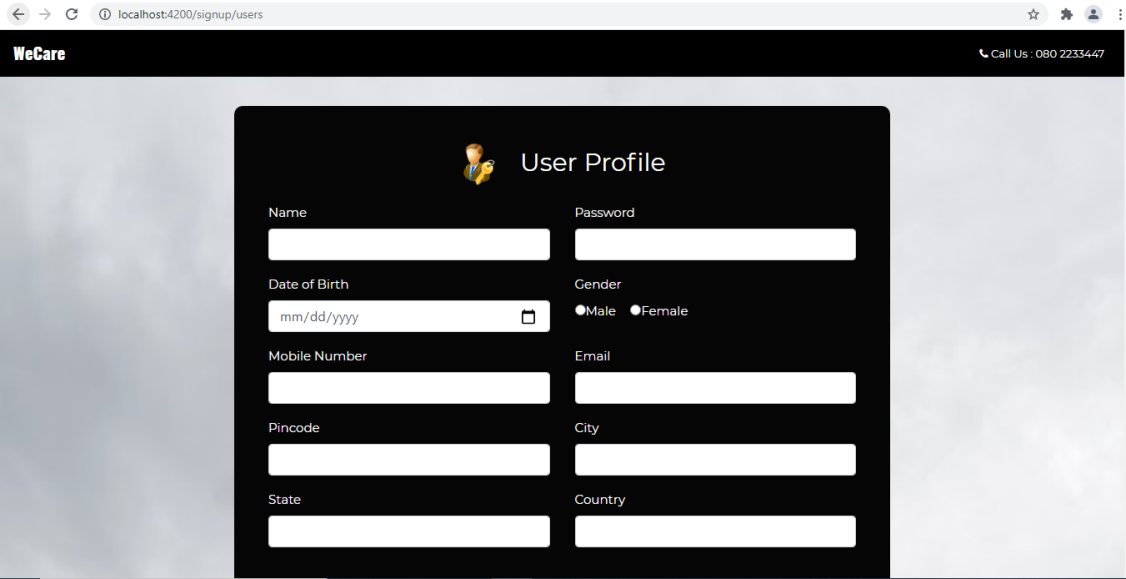
| **FormControlName** | **Validations** |
| --- | --- |
| name | The name should have 3 to 50 characters |
| password | Password should have 5 to 10 characters |
| mobileNumber | Mobile Number should have 10 digits |
| dateOfBirth | Age should be between 20 and 100 years |
| gender | Required |
| speciality | Specialty should have 10 to 50 characters |

Based on the role(users/coaches), registration form should be conditionally rendered as shown below:

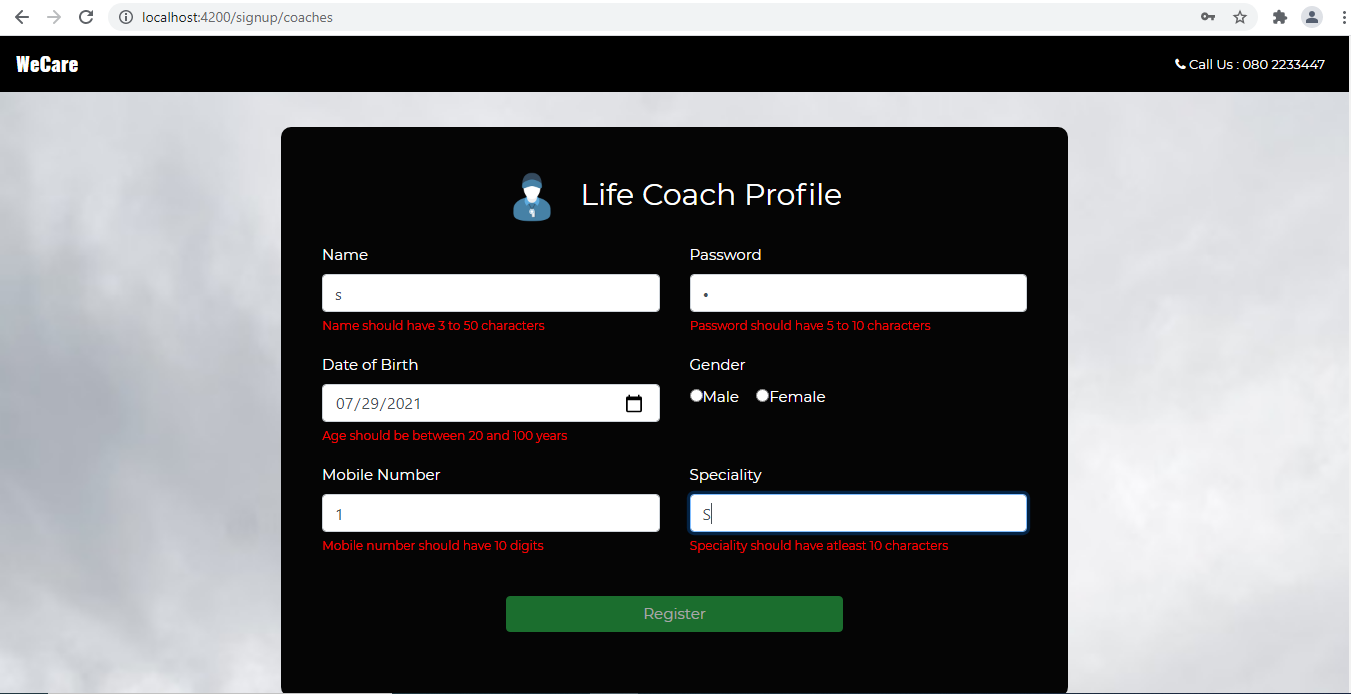
If the role is **coaches (coachregisterForm)**:



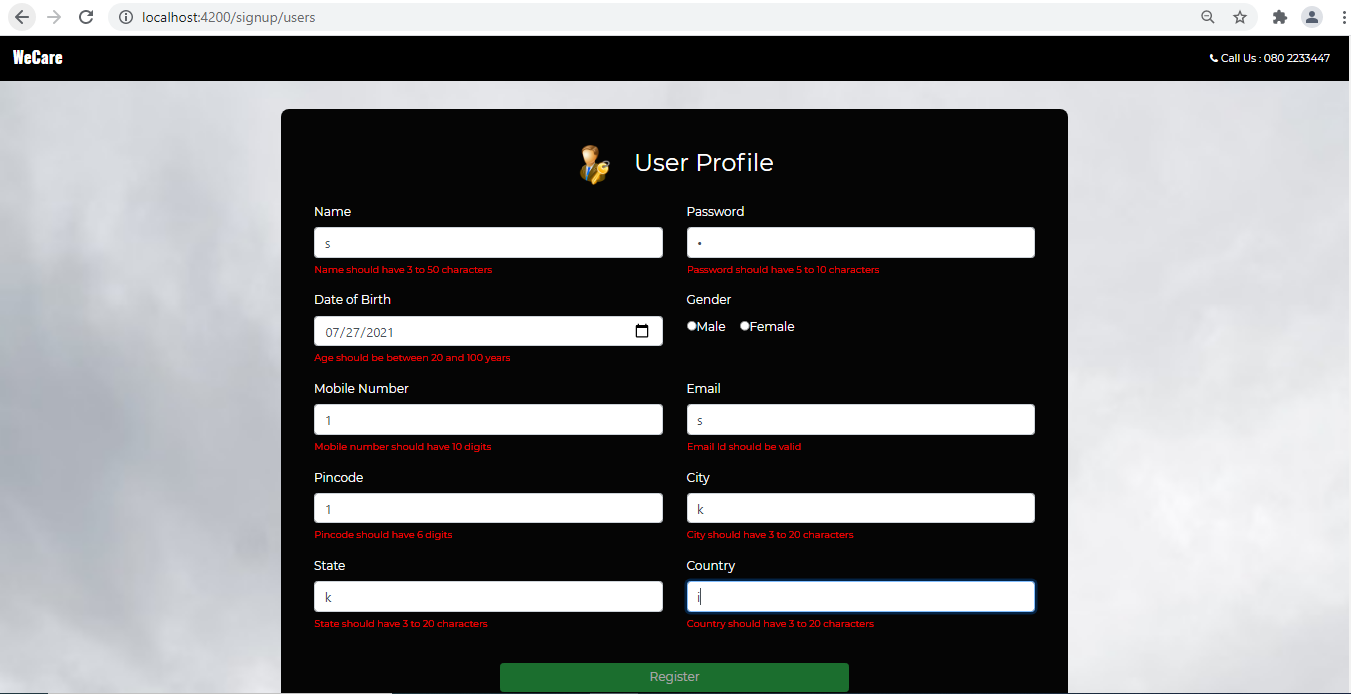
If the role is**users (userregisterForm):**



If **validation fails** show the error messages as shown below for **coaches**:



If **validation fails** show the error messages as shown below for **users**:



Note: All validation should happen whenever there is a modification in the field data

**Methods:**

coachregister(): This method should be invoked on the click of the Register button and perform required operations. Invoke register() method of SignupService.

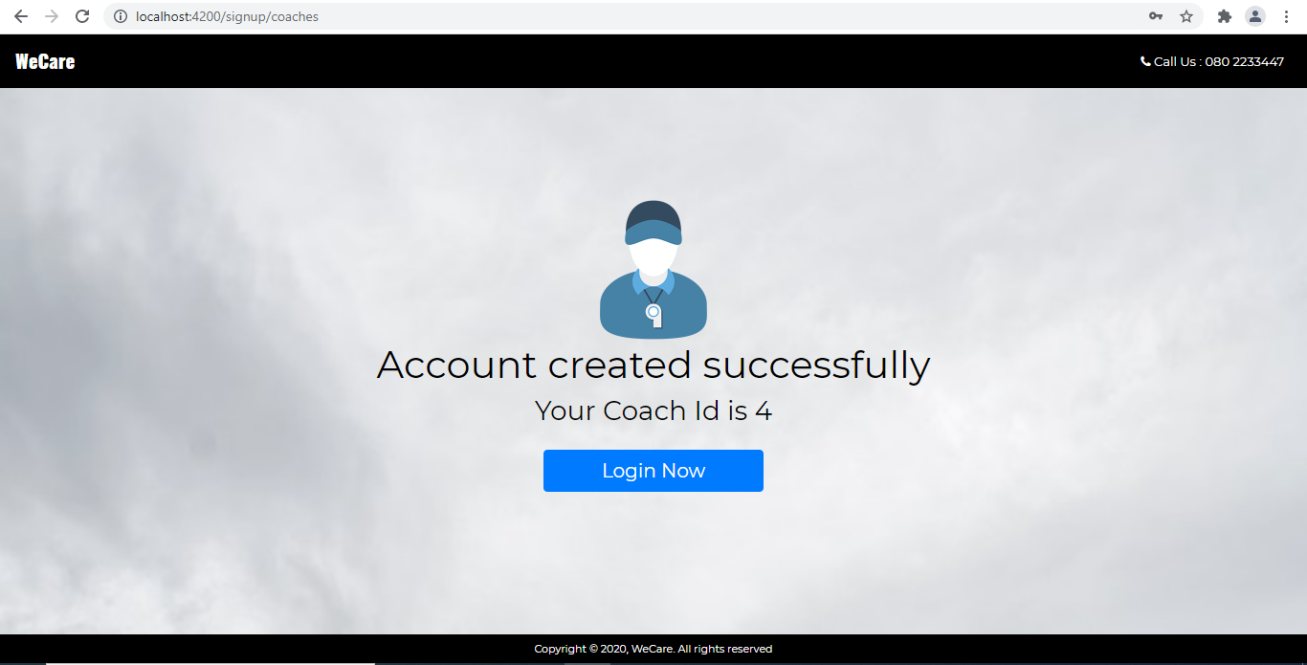
userregister(): This method should be invoked on the click of the Register button and perform required operations. Invoke register() method of SignupService.

**SignupService:**

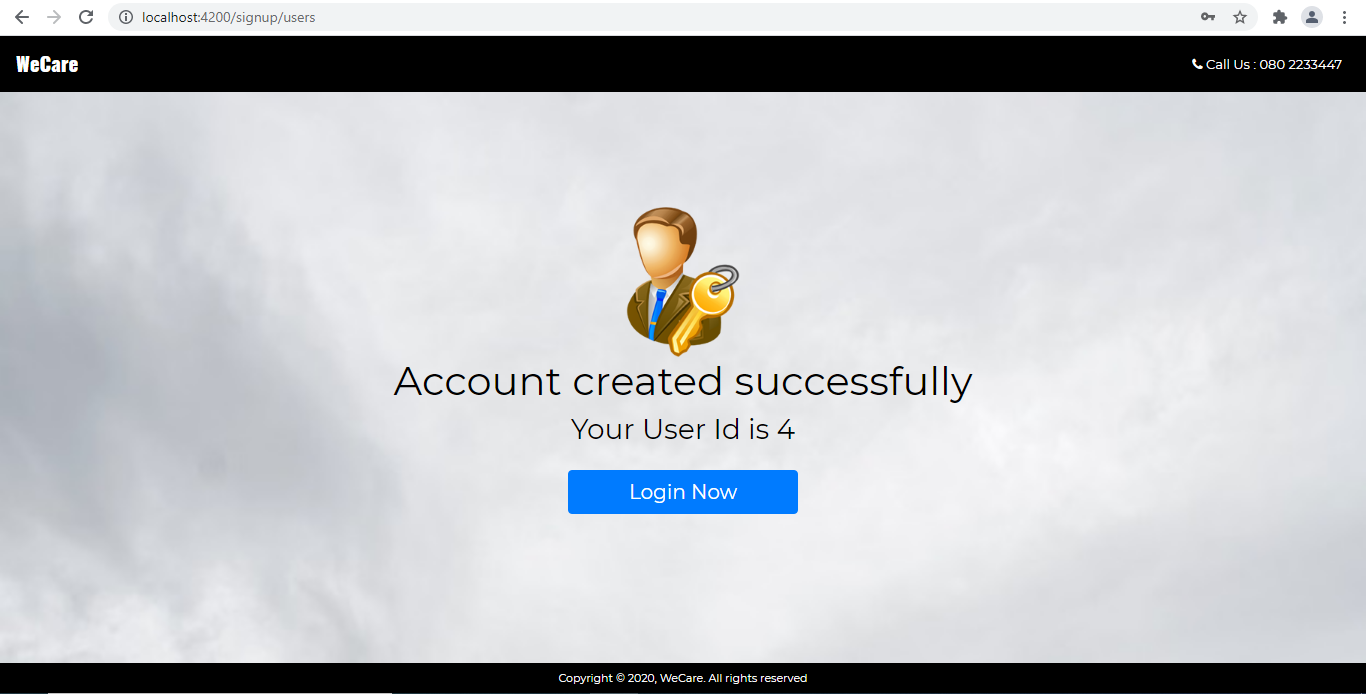
register(): Send a post request to JSON server running at 8080 with role as path and pass the collected data.

If the registration is success make the below page rendered and code should be written in the same component.

Registration Success Page for **coaches:**



Registration Success Page for **users:**

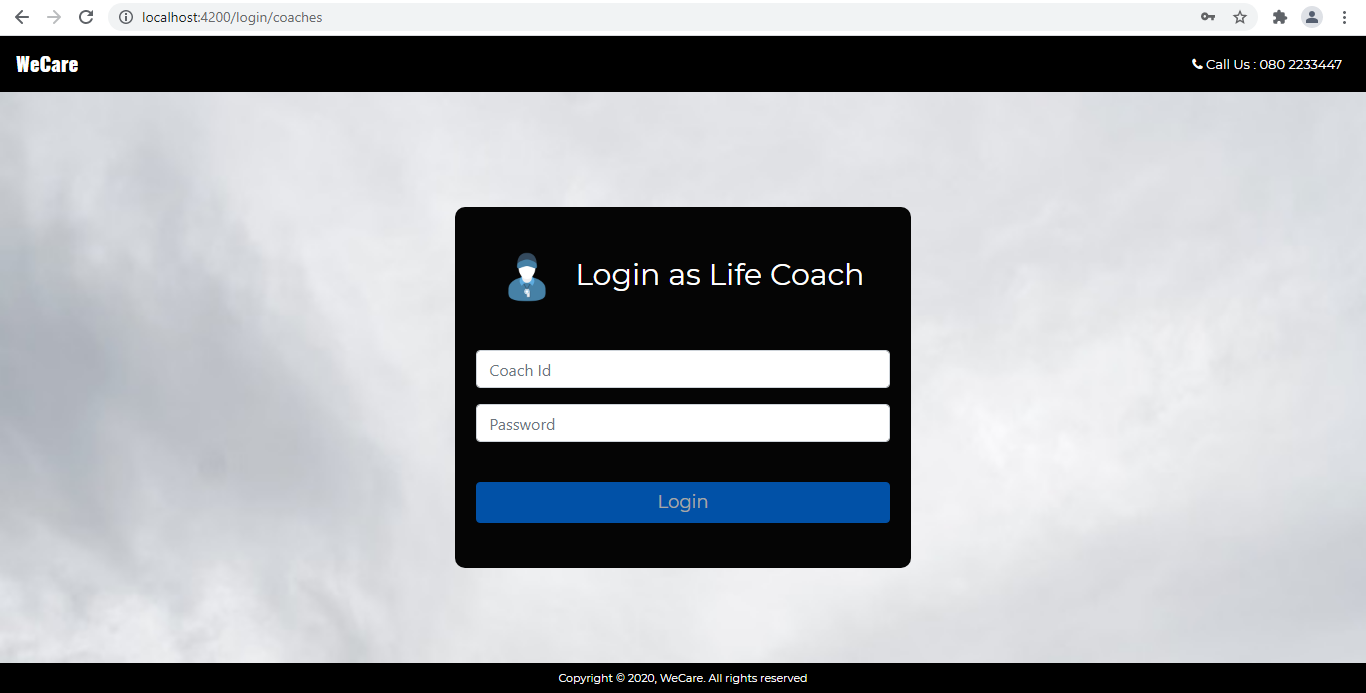


On click of ‘Login Now’ it should load LoginComponent by calling the method **directLogin**(). This method should do the navigation with route path **login/users** or **login/coaches** (based on the role)

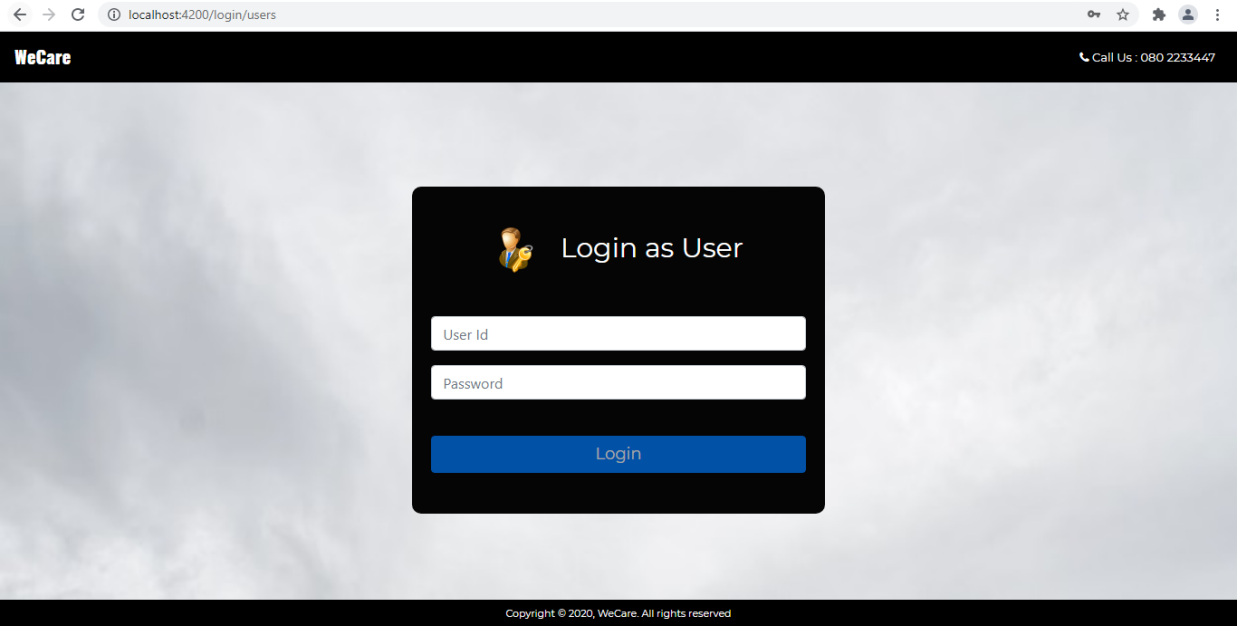
**Login Page:**

This should be coded in LoginComponent and should render a page as shown below:

For **Coaches**:



For**Users:**



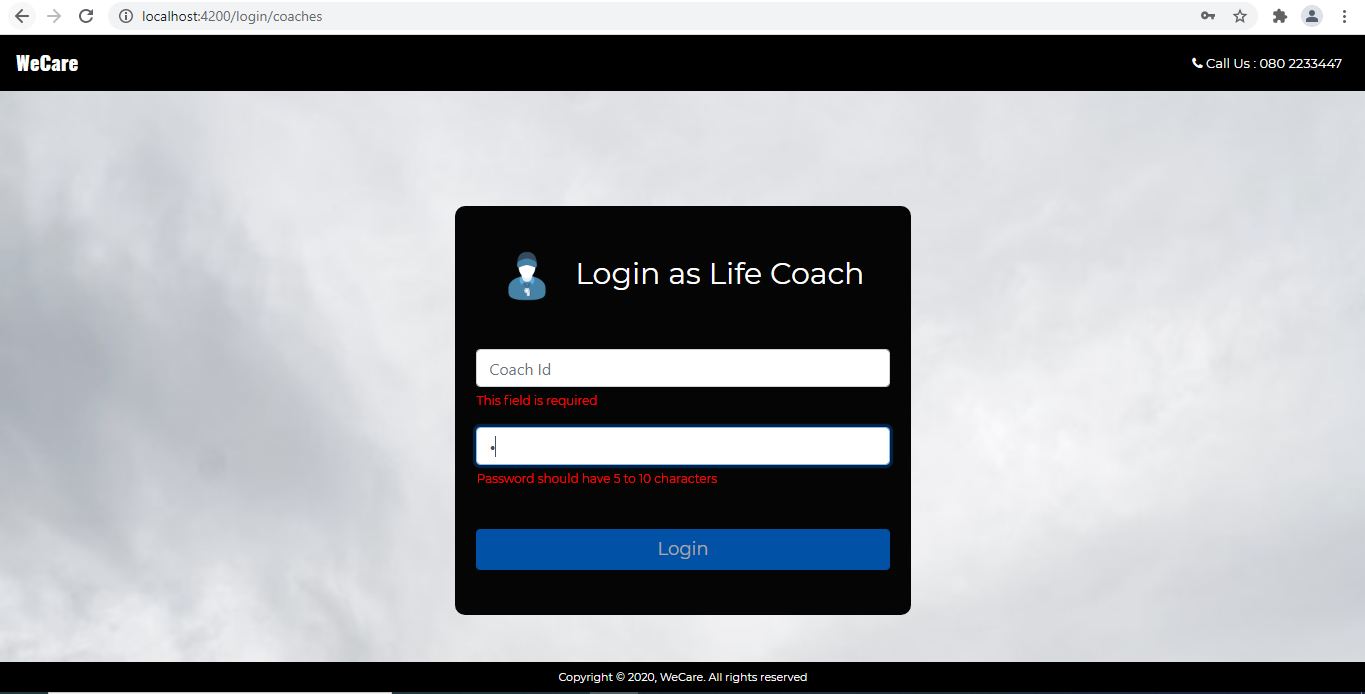
**Properties:**

* role: string;
* loginTitle: string;
* loginImg: string;
* loginForm: FormGroup;
* placeholder: string;
* errorMessage: string;

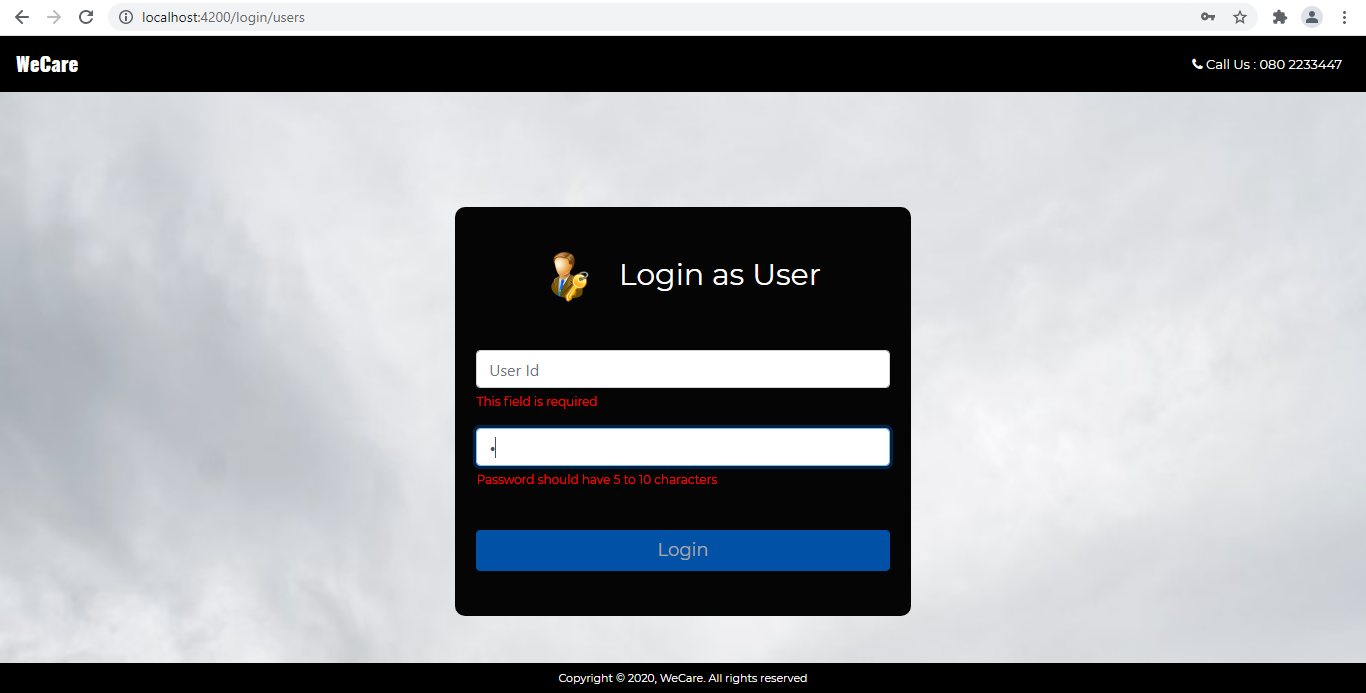
The form control names under form group and corresponding validations are as shown below:

| **FormControlName** | **Validations** |
| --- | --- |
| id | required |
| password | Password should have 5 to 10 characters |

If **validation fails** show the error messages as shown below for **coaches**:



If **validation fails** show the error messages as shown below for **users**:



**Methods:**

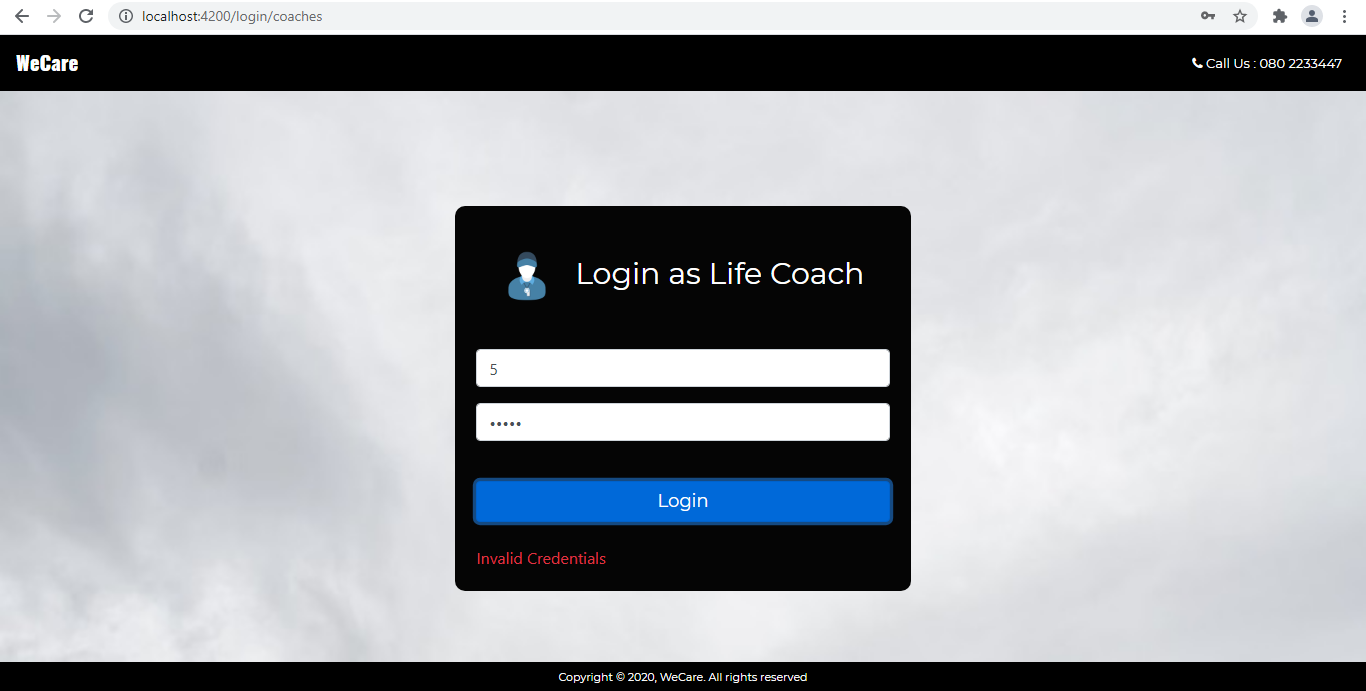
login(): This method should be invoked on the click of the Login button and perform the required operations. Invoke login() method of LoginService.

**LoginService:**

login(): Send a get request to json server running at 8080 with **path /coaches/:id** **or /users/:id** based on the role

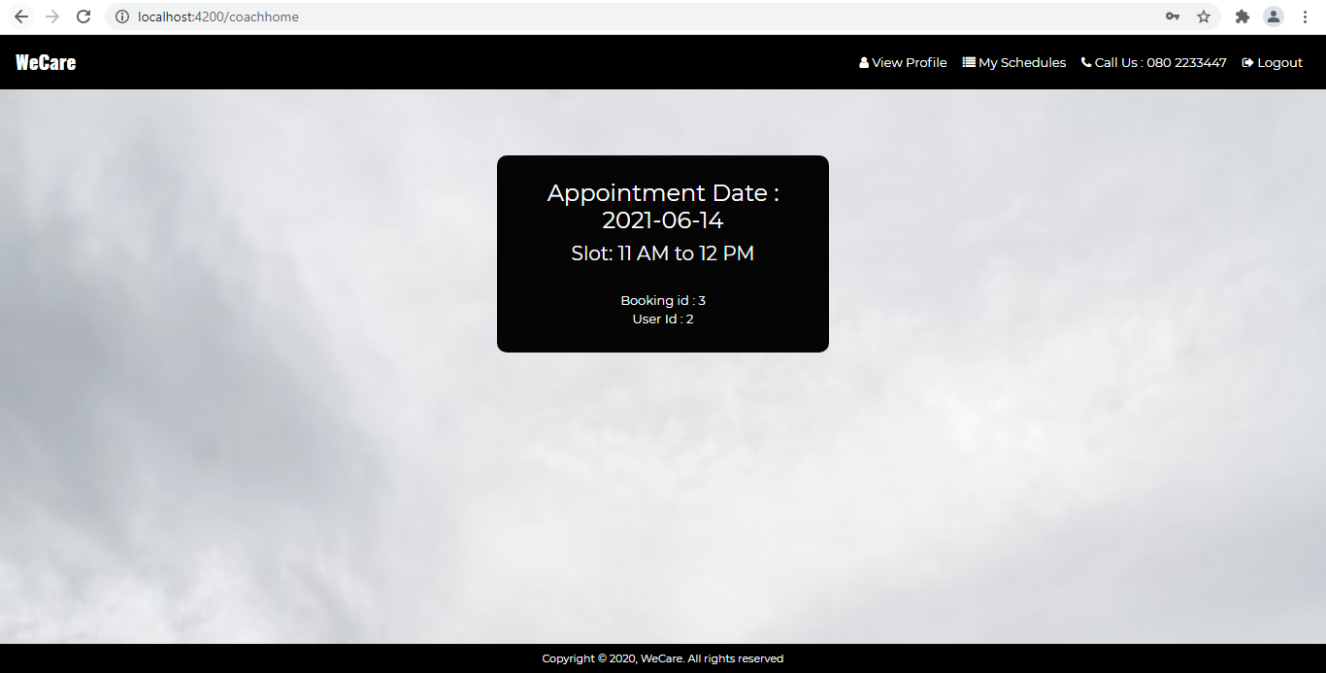
**userId** & **coachId** properties to have data of logged in user & coach

If the Id and password do not exist in coaches/users throw the below error.

On successful login make **CoachComponent**/**UserComponent** loaded based on the role. 

**Coach Home Page:**

On load of this component, it should render the below page. The code must be written in **CoachHomeComponent**. The navbar should be updated as shown below in **AppComponent**.



**Properties:**

* coachId: string
* imgUrl:string;
* scheduleDetails
* msg:string

**Methods:**

* schedules(): This method should be invoked on load of this component and do appropriate operations. Invoke schedules() of CoachhomeService.

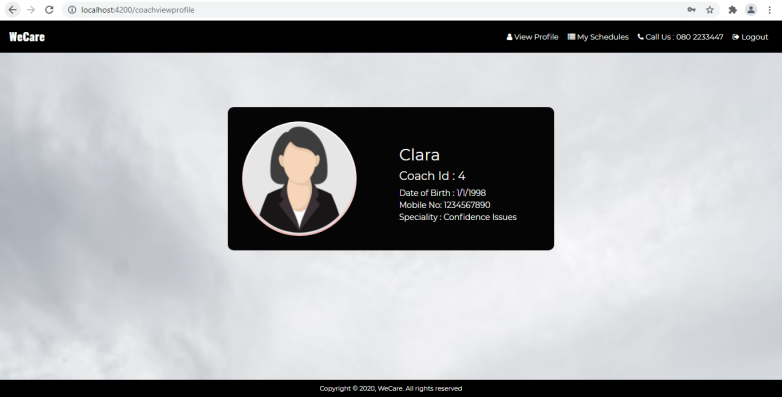
**CoachhomeService:**

* schedules(): Send a get request to JSON server running at 8080 with path ‘/bookings’ and populate scheduleDetails with the success return value.

On click of View Profile link, **Coachprofile** component should be loaded and on click of My Schedules link,  **Coachhome** component should be loaded.

On load of **Coachhome** component, it should show scheduleDetails initially if any. If no schedules fixed yet, the below display should be shown:

**Coachprofile** **Page**:



**Methods:**

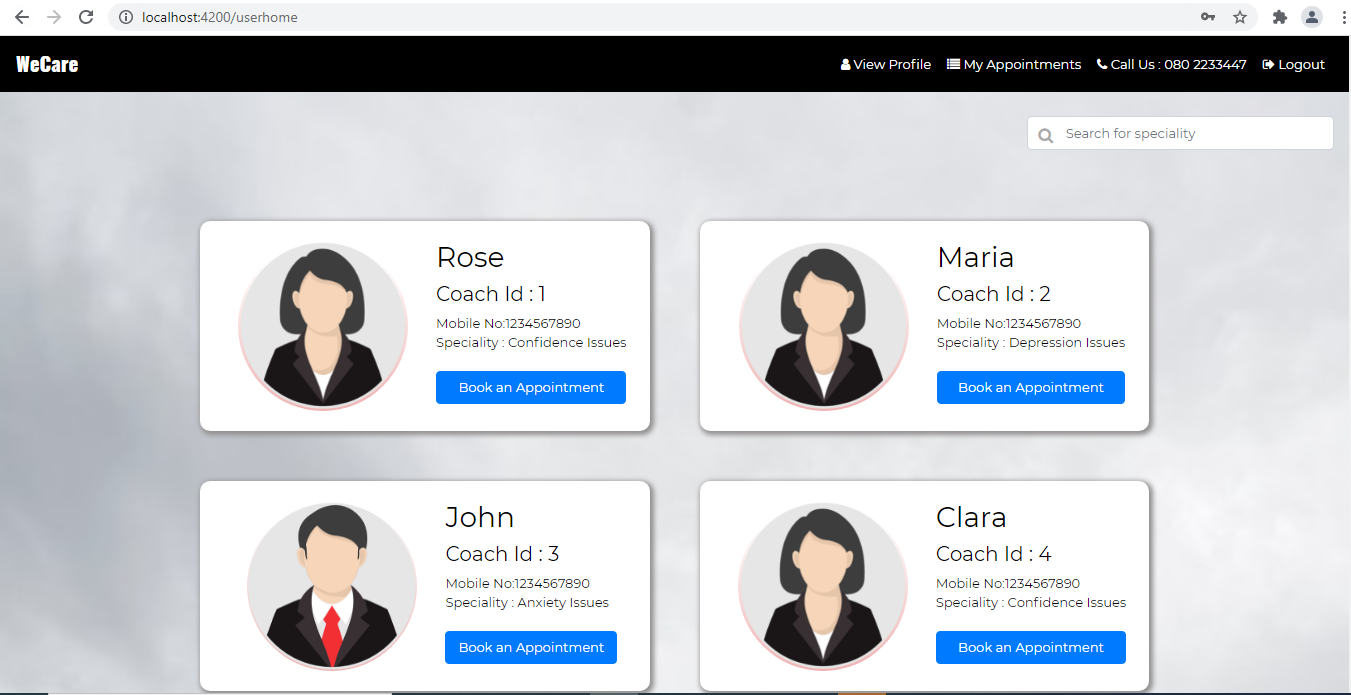
* viewdetails(): This method should be invoked on load of this component and do appropriate operations. Invoke viewDetails () of CoachprofileService.

**CoachprofileService:**

* viewDetails(): Send a get request to JSON server running at 8080 with path ‘/coaches’ and populate details with the success return value.

**User Home Page:**

On load of this component, it should render the below page. The code must be written in UserHomeComponent. The navbar should be updated as shown below in AppComponent.



**Properties:**

* coachArray
* userDetails
* appointmentDetails
* appointmentForm: FormGroup;

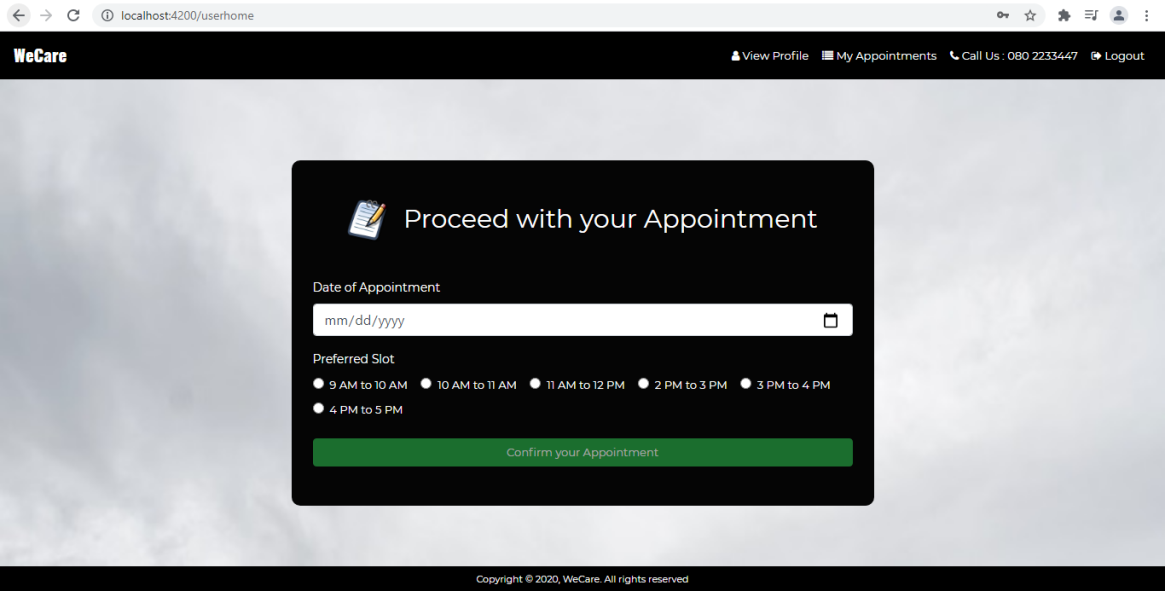
**Methods:**

* allcoaches(): This method should be invoked on the load of this component and do appropriate operations. Invoke allcoaches() of UserhomeService.
* confirmAppointment(): This method should be invoked on booking for a coach and do appropriate operations. Invoke confirmAppointment () of UserhomeService.

**UserhomeService:**

* allcoaches(): Send a get request to JSON server running at 8080 with path ‘/coaches’ and populate coachArray with the success return value.
* confirmAppointment(): Send a post request to JSON server running at 8080 with path ‘/bookings’ with the appointmentDetails.

On click of the Book Appointment button on any coach Card below page should be shown:

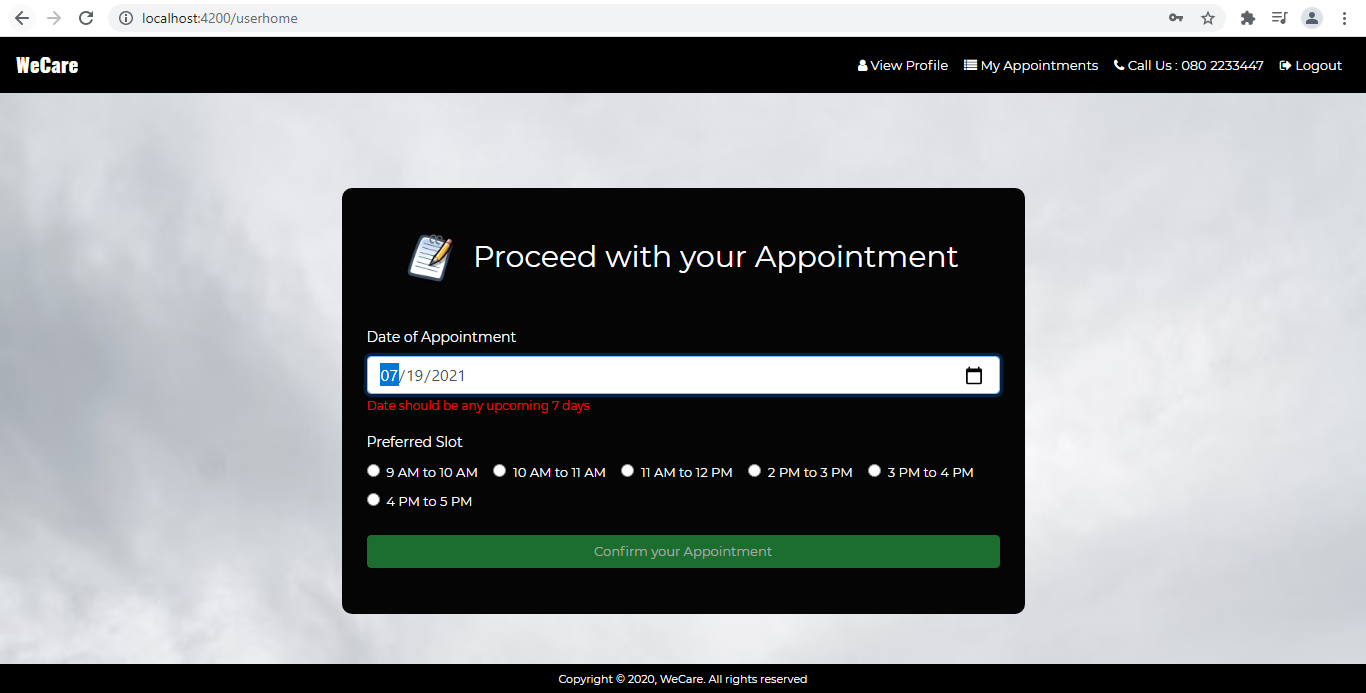


The form control names under form group and corresponding validations are as shown below:

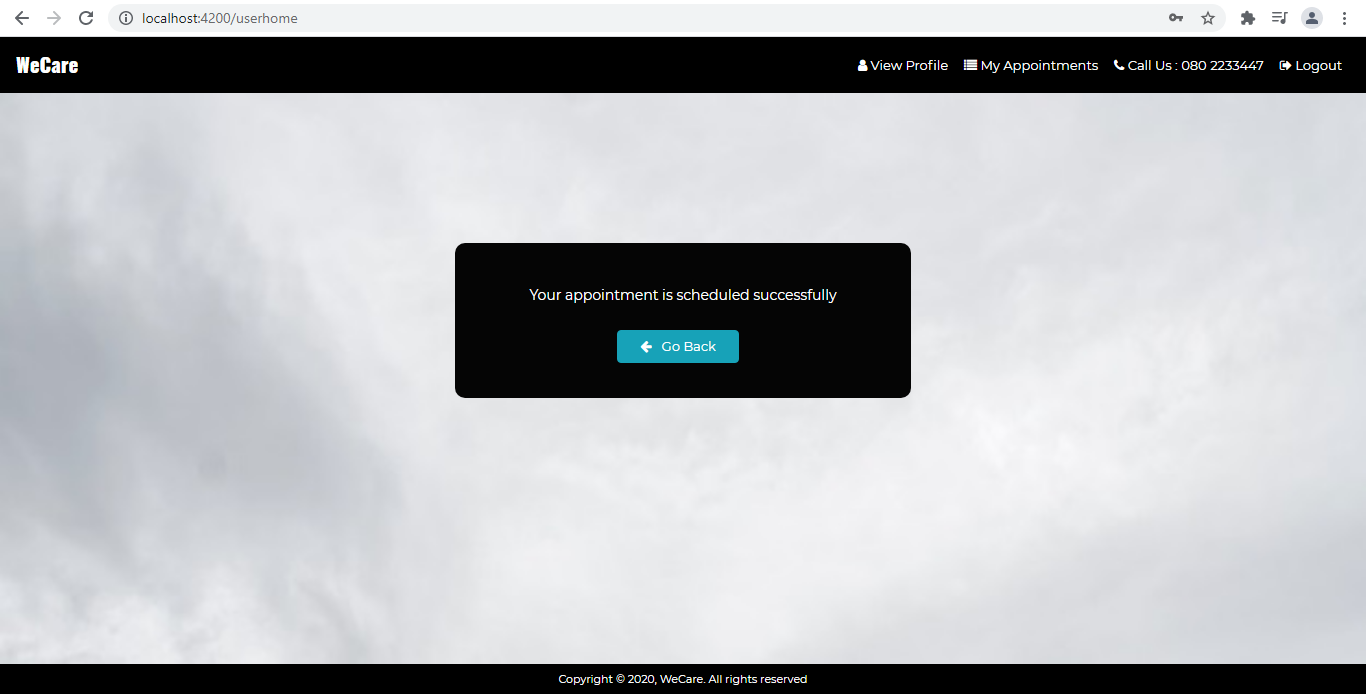
| **FormControlName** | **Validations** |
| --- | --- |
| appointmentDate | Required, the date should be any upcoming 7 days |
| slot | required |

The button will be disabled until all validations are satisfied. On click of Confirm your Appointment it invokes confirmAppointment() which in turn invoke confirmAppointment() of UserhomeService which makes a post request at ‘/bookings’ with data collected

If any validation fails, display the error message as shown below:



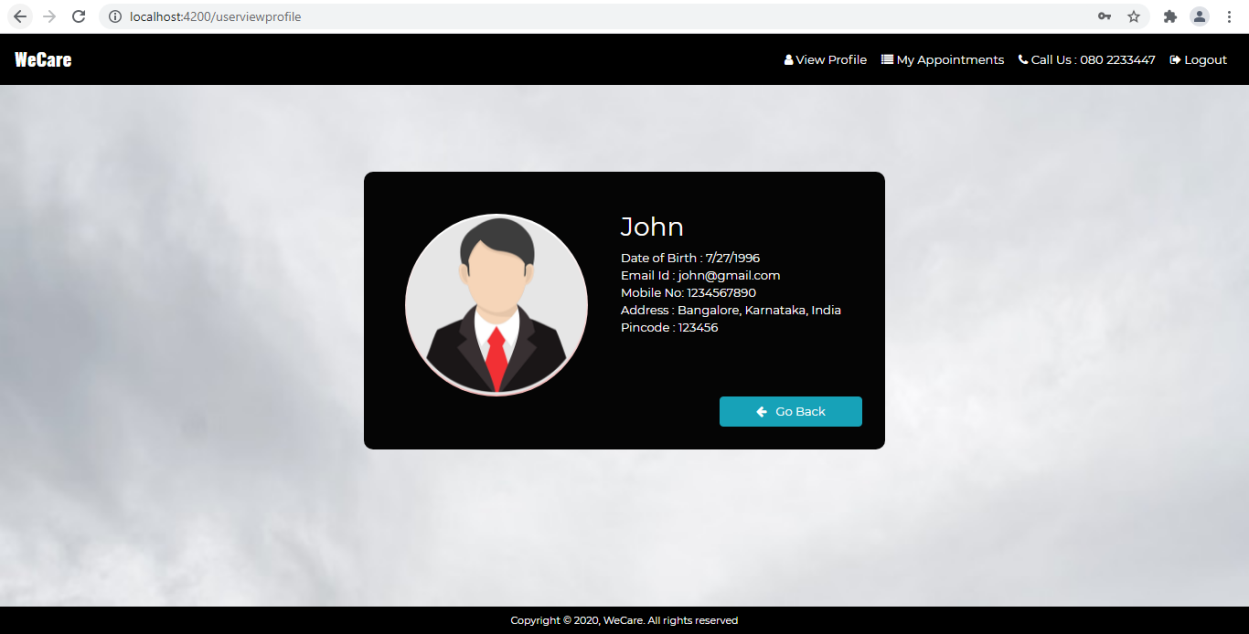
On the successful making of any appointment show the below page:



On click of View Profile link, **Userprofile** Component should be loaded and on click of My Appointments link, **Appointments** Component should be loaded.

**Userprofile Page:**

On clicking the View Profile link it should display the below page:



**Methods:**

viewdetails(): This method should be invoked on the load of this component and do appropriate operations. Invoke viewDetails() of UserprofileService.

**UserprofileService:**

viewDetails (): Send a get request to JSON server running at 8080 with path ‘/users’ and populate details with the success return value

**Appointments Page:**

**Methods:**

appointment(): This method should be invoked on the load of this component and do appropriate operations. Invoke appointment() of AppointmentsService.

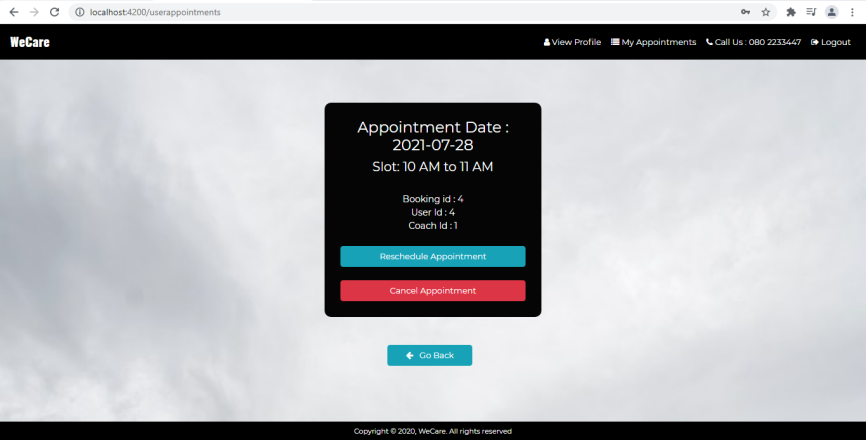
rescheduleAppointment(): This method should be invoked on click of Reschedule your Appointment button and it will invoke rescheduleAppointment() of AppointmentsService

cancel(): This methos should be invoked on Click of cancel button and it will invoke cancel() method of AppointmentsService

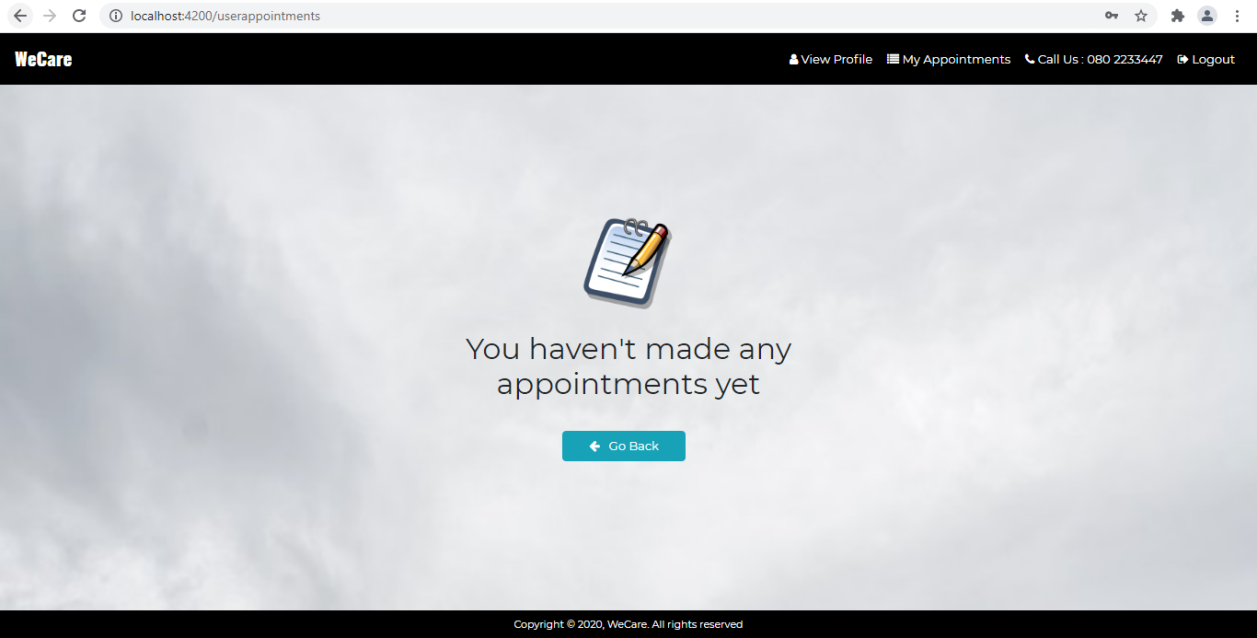
**AppointmentsService:**

* appointment (): Send a get request to JSON server running at 8080 with path ‘/bookings’ and populate details with the success return value.
* rescheduleAppointment(): Send a patch request to JSON server running in 8080 with path ‘/bookings/:bookingId’ with the data to be updated
* cancel(): Send a delete request to JSON server running in 8080 with path ‘bookings/:bookingId’

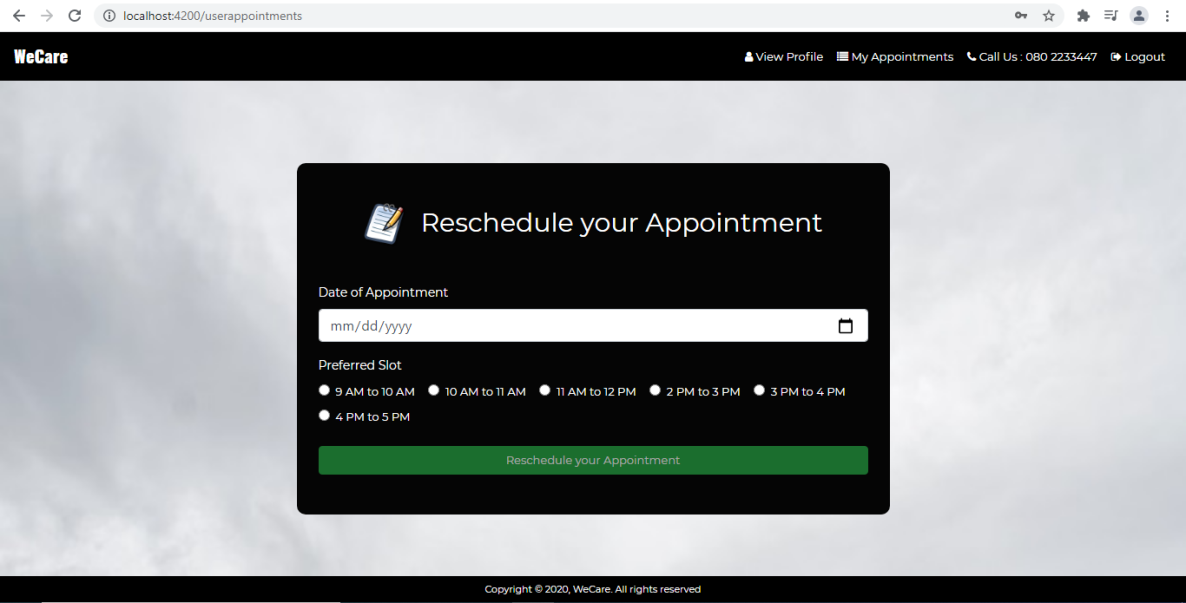
On clicking the My Appointments link it should display all the appointments made by the user as shown below.



If no appointments made yet show the below screen:



On click of the Reschedule Appointment Button display the below page:



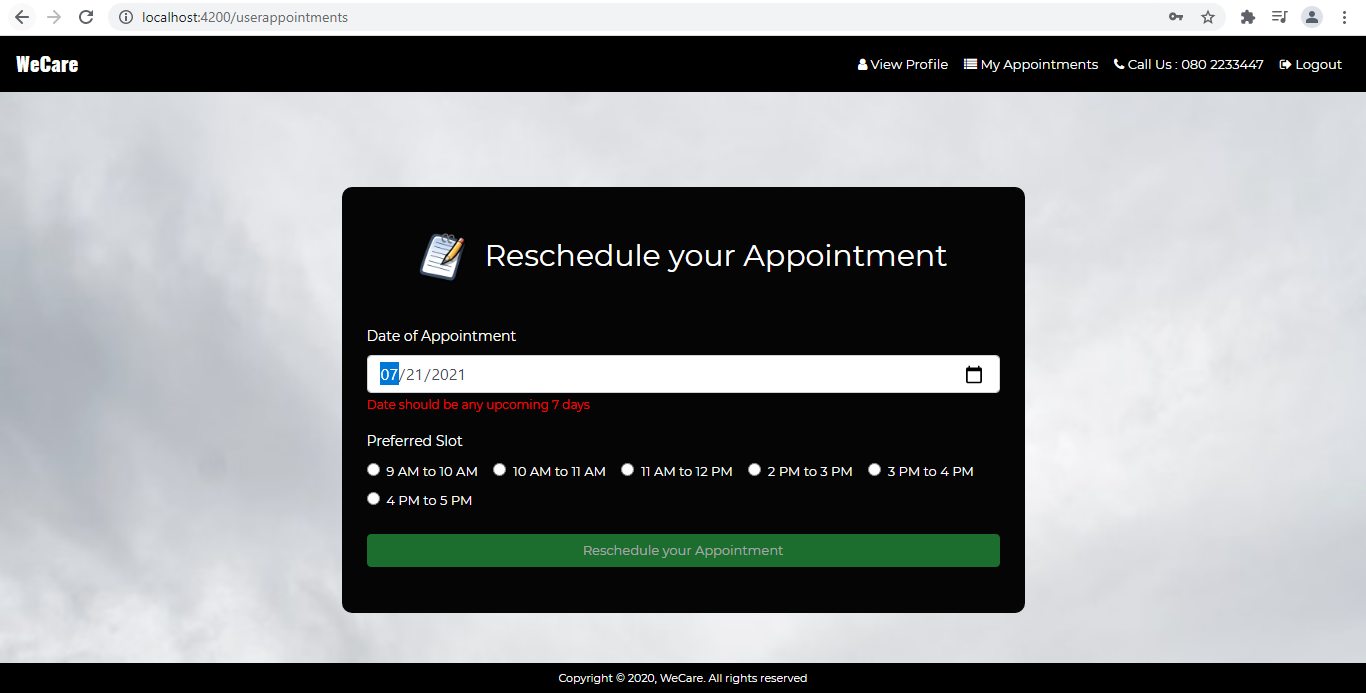
The form control names under form group and corresponding validations are as shown below:

rescheduleForm: FormGroup;

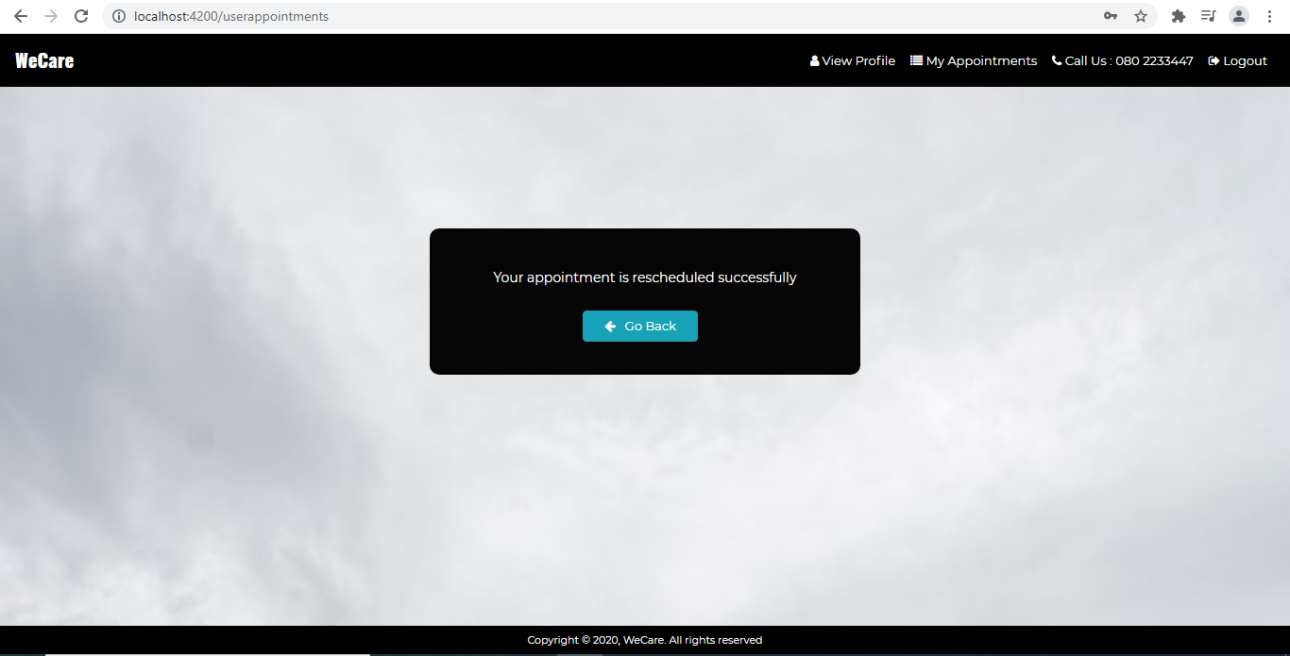
| **FormControlName** | **Validations** |
| --- | --- |
| appointmentDate | Required, the date should be any upcoming 7 days |
| slot | required |

The button will be disabled until all validations are satisfied. On click of Confirm your Appointment, it invokes rescheduleAppointment() which in turn invoke rescheduleAppointment of AppointmentsService which makes a patch request at ‘bookings/:bookingId’ with data collected

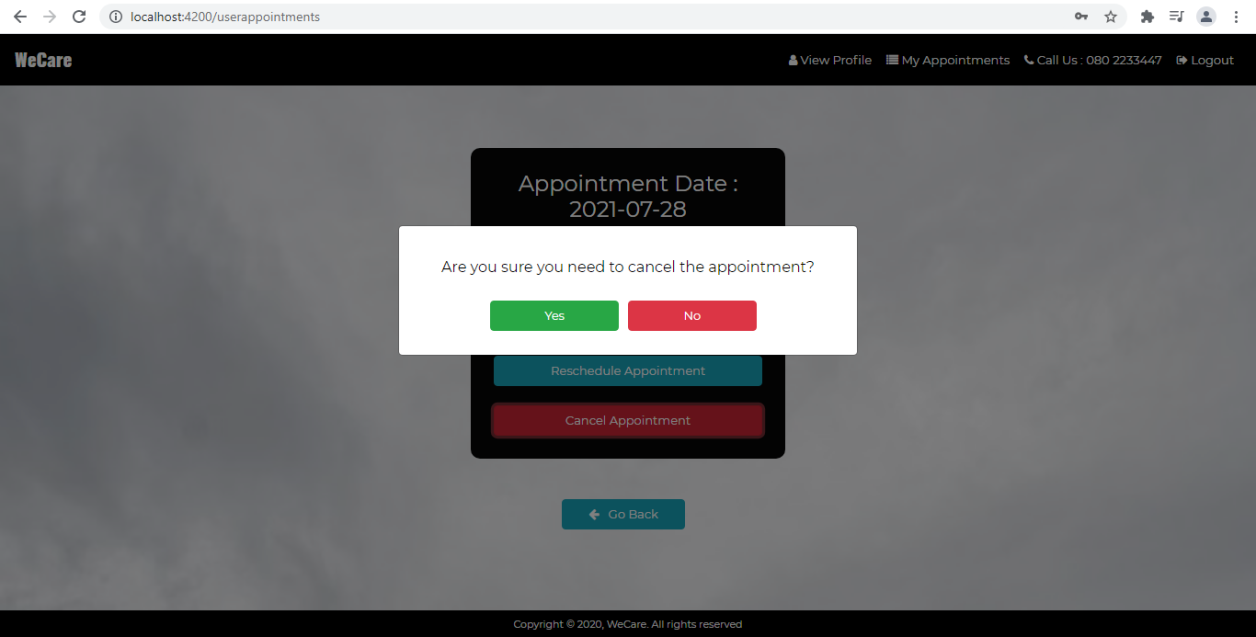
If any validation fails, display the error message as shown below:



On successful rescheduling show the below page:



On click of cancel Appointment Button invoke cancel() which in turn invoke cancel() in AppointmentService where it makes an HTTP delete request at ‘/bookings/:bookingId’



On successful deletion of appointment show the below page:

