CHAT ROOM

By-Rupindar kaur , Priyanka Halijwale , Supriya Jadhav

Batch Id: CHN18ID001

1. **Login Component.**

* **Login.CSS**

.login-container{

margin-top: 5%;

margin-bottom: 5%;

}

.login-form-1{

padding: 5%;

box-shadow: 0 5px 8px 0 rgba(0, 0, 0, 0.2), 0 9px 26px 0 rgba(0, 0, 0, 0.19);

}

.login-form-1 h3{

text-align: center;

color: #333;

}

.login-form-2{

padding: 5%;

background: #0062cc;

box-shadow: 0 5px 8px 0 rgba(0, 0, 0, 0.2), 0 9px 26px 0 rgba(0, 0, 0, 0.19);

}

.login-form-2 h3{

text-align: center;

color: #fff;

}

.login-container form{

padding: 10%;

}

.btnSubmit

{

width: 50%;

border-radius: 1rem;

padding: 1.5%;

border: none;

cursor: pointer;

}

.login-form-1 .btnSubmit{

font-weight: 600;

color: #fff;

background-color: #0062cc;

}

.login-form-2 .btnSubmit{

font-weight: 600;

color: #0062cc;

background-color: #fff;

}

.login-form-2 .ForgetPwd{

color: #fff;

font-weight: 600;

text-decoration: none;

}

.login-form-1 .ForgetPwd{

color: #0062cc;

font-weight: 600;

text-decoration: none;

}

* **Login.Html**

<div class="container login-container">

<div class="row">

<div class="col-md-6 login-form-1">

<h3>Registration</h3>

<form #Userform="ngForm" (submit)="OnSubmit(Userform)" autocomplete="off">

<div class="form-group" [class.has-error]="UserEmail.invalid && UserEmail.touched">

<input type="text" name="UserEmail" placeholder="Username\*" [(ngModel)]="this.service.formData.UserEmail" #UserEmail="ngModel"

class="form-control" required email>

<span class="help-block" \*ngIf="UserEmail.errors?.required && UserEmail.touched" style="color:red">

Username is Required

</span>

<span class="help-block" \*ngIf="UserEmail.errors?.email && UserEmail.touched" style="color:red">

Username is Invalid

</span>

</div>

<div class="form-group" [class.has-error]="UserEmail.invalid && UserEmail.touched">

<input type="password" class="form-control" name="Password" placeholder="Password\*" [(ngModel)]="this.service.formData.Password" #Password="ngModel" required Password>

<span class="help-block" \*ngIf="Password.errors?.required && Password.touched" style="color:red">

Password is Required

</span>

</div>

<div class="form-group" [class.has-error]="DateOfBirth.invalid && DateOfBirth.touched">

<input type="text" placeholder="Date Of Birth\*" [(ngModel)]="this.service.formData.DateOfBirth" id="DateOfBirth" class="form-control" name="DateOfBirth" #DateOfBirth="ngModel" required DateOfBirth>

<span class="help-block" \*ngIf="DateOfBirth.errors?.required && DateOfBirth.touched" style="color:red">

DOB is Required

</span>

</div>

<label class="radio-inline">

<input type="radio" value="male" required #Gender="ngModel" name="Gender" [(ngModel)]="this.service.formData.Gender" >Male

</label>&nbsp;

<label class="radio-inline">

<input type="radio" value="female" required name="Gender" #Gender="ngModel" [(ngModel)]="this.service.formData.Gender" >Female

</label>

<div class="form-group">

<input type="submit" class="btnSubmit" value="Register" [disabled]="Userform.invalid" />

</div>

</form>

</div>

<div class="col-md-6 login-form-2">

<h3>Login</h3>

<form #Loginform="ngForm" (submit)="LoginSubmit(Loginform)" autocomplete="off">

<div class="form-group" [class.has-error]="UserEmail.invalid && UserEmail.touched">

<input type="text" class="form-control" name="UserEmail" placeholder="Your Email \*" [(ngModel)]="this.service.loginData.UserEmail" #UserEmail="ngModel" required email>

<span class="help-block" \*ngIf="UserEmail.errors?.required && UserEmail.touched" style="color:red">

Email is Required

</span>

<span class="help-block" \*ngIf="UserEmail.errors?.UserEmail && UserEmail.touched" style="color:red">

Email is Invalid

</span>

</div>

<div class="form-group" [class.has-error]="Password.invalid && Password.touched">

<input type="password" class="form-control" name="Password" placeholder="Your Password \*" [(ngModel)]="this.service.loginData.Password" #Password="ngModel" required Password>

<span class="help-block" \*ngIf="Password.errors?.required && Password.touched" style="color:red">

Password is Required

</span>

</div>

<div class="form-group">

<input type="submit" class="btnSubmit" value="Login" />

</div>

</form>

</div>

</div>

</div>

* **Login.ts**

import { Component, OnInit } from '@angular/core';

import { Register } from '../shared/Register.Model';

import { Login } from '../shared/Login.Model';

import { RegisterService } from '../shared/register.service';

import { ToastrService } from 'ngx-toastr';

import {HttpClient} from '@angular/common/http';

import { MockNgModuleResolver } from '@angular/compiler/testing';

import { NgForm, NgFormSelectorWarning } from '@angular/forms';

import { Router } from '@angular/router';

import { PersistenceService,StorageType } from 'angular-persistence';

@Component({

selector: 'app-register',

templateUrl: './register.component.html',

styleUrls: ['./register.component.css']

})

export class RegisterComponent implements OnInit {

public empids:number[];

public result:any;

useremail:string;

allUsers:Register[];

userCheck:Login;

reg:Register;

constructor( private router: Router, private service:RegisterService,private toastr:ToastrService,private persistenceService: PersistenceService) {

this.service.formData={

UserId:null,

UserEmail:null,

Password:null,

Gender:null,

DateOfBirth:null

}

this.result=this.service.list;

this.service.loginData={

UserEmail:null,

Password:null,

}

}

ngOnInit() {

this.resetForm();

this.service.refreshList();

}

OnSubmit(form:NgForm)

{

form.value.UserId=1;

console.log(form.value.UserId);

console.log(form.value.Gender);

console.log(form.value.Password);

console.log(form.value.UserEmail);

console.log(form.value.DateOfBirth);

this.useremail=form.value.UserEmail;

this.reg=this.service.list.find(e=>e.UserEmail==this.useremail);

console.log(this.useremail);

if(this.reg==null)

// if(form.value.eid=this.employeeid)

{

this.InsertRecord(form);

{{debugger}}

this.persistenceService.set('LoggedUser', form.value.UserEmail, {type: StorageType.SESSION});

this.router.navigate(['/UserChat']);

}

// else

// {

// this.UpdateRecord(form);

// }

}

getUsers()

{

this.service.refreshList();

this.allUsers=this.service.list;

}

LoginSubmit(form:NgForm)

{

console.log(form.value.UserEmail);

console.log(form.value.Password);

this.userCheck=this.service.list.find(e=>e.UserEmail==form.value.UserEmail && e.Password==form.value.Password)

{{debugger}}

if(this.userCheck!=null)

{

alert("Login Successfully");

this.persistenceService.set('LoggedUser', form.value.UserEmail, {type: StorageType.SESSION});

this.router.navigate(['/UserChat']);

}

else

{

alert("Check Email and Password");

this.router.navigate(['/']);

}

}

InsertRecord(form:NgForm)

{this.persistenceService.set('LoggedUser', form.value.UserEmail, {type: StorageType.SESSION});

this.router.navigate(['/UserChat']);

this.service.postEmployee(form.value).subscribe(res=>{

this.toastr.success("Inserted Successfully","User Register");

this.resetForm(form)

this.service.refreshList();

});

}

// UpdateRecord(form:NgForm)

// {

// this.service.putEmployee(form.value).subscribe(res=>{

// this.toastr.warning("Updated Successfully","Employee Register");

// this.resetForm(form)

// this.service.refreshList();

// });

// }

resetForm(form?:NgForm)

{

if(form!=null)

{

form.resetForm();

this.service.formData={

UserId:null,

UserEmail:null,

Password:null,

Gender:null,

DateOfBirth:null

}

}

}

}

1. **Userchat Component**

* Userchat.css

.container{max-width:1170px;}

img{ max-width:100%;}

.inbox\_people {

background: #f8f8f8 none repeat scroll 0 0;

float: left;

overflow: hidden;

width: 40%; border-right:1px solid #c4c4c4;

}

.inbox\_msg {

border: 1px solid #c4c4c4;

clear: both;

overflow: hidden;

}

.top\_spac{ margin: 20px 0 0;}

.recent\_heading

{

text-align:center;

width:40%;

padding: 10px 10px;

}

/\*.srch\_bar {

display: inline-block;

text-align: right;

width: 60%; padding:

}\*/

.headind\_srch{ padding:10px 29px 10px 20px; overflow:hidden; border-bottom:1px solid #c4c4c4;}

.recent\_heading h4 {

color: #05728f;

font-size: 21px;

margin: auto;

}

.srch\_bar input{ border:1px solid #cdcdcd; border-width:0 0 1px 0; width:80%; padding:2px 0 4px 6px; background:none;}

.srch\_bar .input-group-addon button {

background: rgba(0, 0, 0, 0) none repeat scroll 0 0;

border: medium none;

padding: 0;

color: #707070;

font-size: 18px;

}

.srch\_bar .input-group-addon { margin: 0 0 0 -27px;}

.chat\_ib h5{ font-size:15px; color:#464646; margin:0 0 8px 0;}

.chat\_ib h5 span{ font-size:13px; float:right;}

.chat\_ib p{ font-size:14px; color:#989898; margin:auto}

.chat\_img {

float: left;

width: 11%;

}

.chat\_ib {

float: left;

padding: 0 0 0 15px;

width: 88%;

}

.chat\_people{ overflow:hidden; clear:both;}

.chat\_list {

border-bottom: 1px solid #c4c4c4;

margin: 0;

padding: 18px 16px 10px;

}

.inbox\_chat { height: 550px; overflow-y: scroll;}

.active\_chat{ background:#ebebeb;}

.incoming\_msg\_img {

display: inline-block;

width: 6%;

}

.received\_msg {

display: inline-block;

padding: 0 0 0 10px;

vertical-align: top;

width: 92%;

}

.received\_withd\_msg p {

background: #ebebeb none repeat scroll 0 0;

border-radius: 3px;

color: #646464;

font-size: 14px;

margin: 0;

padding: 5px 10px 5px 12px;

width: 100%;

}

.time\_date {

color: #747474;

display: block;

font-size: 12px;

margin: 8px 0 0;

}

.received\_withd\_msg { width: 57%;}

.mesgs {

float: left;

padding: 30px 15px 0 25px;

width: 60%;

}

.sent\_msg p {

background: #05728f none repeat scroll 0 0;

border-radius: 3px;

font-size: 14px;

margin: 0; color:#fff;

padding: 5px 10px 5px 12px;

width:100%;

}

.outgoing\_msg{ overflow:hidden; margin:26px 0 26px;}

.sent\_msg {

float: right;

width: 46%;

}

.input\_msg\_write input {

background: rgba(0, 0, 0, 0) none repeat scroll 0 0;

border: medium none;

color: #4c4c4c;

font-size: 15px;

min-height: 48px;

width: 100%;

}

.type\_msg {border-top: 1px solid #c4c4c4;position: relative;}

.msg\_send\_btn {

background: #05728f none repeat scroll 0 0;

border: medium none;

border-radius: 50%;

color: #fff;

cursor: pointer;

font-size: 17px;

height: 33px;

position: absolute;

right: 0;

top: 11px;

width: 33px;

}

/\*.messaging { padding: 0 0 50px 0;

}\*/

.msg\_history {

height: 516px;

overflow-y: auto;

}

* Userchat.html

<nav class="navbar navbar-expand-sm bg-primary navbar-dark">

<ul class="nav navbar-nav navbar-right">

<li style="padding:30px" routerLink="/Register">Logout</li>

</ul>

</nav>

<div class="container">

<h3 class=" text-center">Messaging</h3>

<div class="messaging">

<div class="inbox\_msg">

<div class="inbox\_people">

<div class="headind\_srch">

<div class="recent\_heading">

<h4>{{this.currentUser}}</h4>

</div>

<div class="md-form active-pink active-pink-2 mb-3">

<input class="form-control" placeholder="SEARCH" [(ngModel)]="username" id="search1" aria-label="Search" >

</div>

</div>

<div class="inbox\_chat">

<div class="chat\_list active\_chat" \*ngFor="let user of this.filterList|userfilterPipe:username" (click)="selectedUser(user)">

<div class="chat\_people">

<div class="chat\_img"> <img src="https://ptetutorials.com/images/user-profile.png" alt="sunil"> </div>

<div class="chat\_ib">

<h5>{{user.UserEmail}} <span class="chat\_date">{{user.Gender}}</span></h5>

<p>Date of Birth: {{user.DateOfBirth|date:format:longdate}}</p>

</div>

</div>

</div>

</div>

</div>

<div class="mesgs">

<div class="recent\_heading">

<h4>{{this.toUserName}}</h4>

</div>

<div class="msg\_history">

<div \*ngFor="let chat of this.specificChat">

<div class="incoming\_msg" \*ngIf="chat.touser==this.LoggedUserId">

<div class="incoming\_msg\_img"> <img src="https://ptetutorials.com/images/user-profile.png" alt="sunil"> </div>

<div class="received\_msg">

<div class="received\_withd\_msg">

<p>{{chat.msgcontent}}</p>

<span class="time\_date"> {{chat.msgdate|date:'M/d/y'}} | {{chat.msgdate|date:'shortTime'}} </span></div>

</div>

</div>

<div class="outgoing\_msg" \*ngIf="chat.fromUser==this.LoggedUserId">

<div class="sent\_msg">

<p>{{chat.msgcontent}}</p>

<span class="time\_date">{{chat.msgdate|date:'M/d/y'}} | {{chat.msgdate|date:'shortTime'}} </span> </div>

</div>

</div>

</div>

<div class="type\_msg">

<div class="input\_msg\_write">

<form #Userform="ngForm" (submit)="sendMessage(Userform)" autocomplete="off">

<input type="text" name="msgcontent" [(ngModel)]="this.msgservice.msgcontent" #msgcontent="ngModel" class="write\_msg" placeholder="Type a message" />

<button class="msg\_send\_btn" type="submit"><i class="fa fa-paper-plane-o" aria-hidden="true"></i></button>

</form>

</div>

</div>

</div>

</div>

<p class="text-center top\_spac"> Design by <a target="\_blank" href="#">The Three Musketeers</a></p>

</div></div>

* Userchat.ts

import { Component, OnInit } from '@angular/core';

import { Register } from '../shared/Register.Model';

import { MessageM } from '../shared/Message.Model';

import { RegisterService } from '../shared/register.service';

import {UserMessagesService} from '../shared/user-messages.service';

import { ToastrService } from 'ngx-toastr';

import {HttpClient} from '@angular/common/http';

import { MockNgModuleResolver } from '@angular/compiler/testing';

import { NgForm, NgFormSelectorWarning } from '@angular/forms';

import { PersistenceService,StorageType } from 'angular-persistence';

import {formatDate} from '@angular/common';

@Component({

selector: 'app-userchat',

templateUrl: './userchat.component.html',

styleUrls: ['./userchat.component.css']

})

export class UserchatComponent implements OnInit {

currentUser:string;

currentUserId:Register;

ToUser:number;

firstlogin:string;

specificChat:MessageM[];

LoggedUserId:number;

toUserName:string;

filterList:Register[];

constructor(private msgservice:UserMessagesService ,private rsgservice:RegisterService,private toastr:ToastrService,private persistenceService: PersistenceService ) { }

ngOnInit() {

this.rsgservice.refreshList();

{{debugger}}

this.currentUser=this.persistenceService.get('LoggedUser',StorageType.SESSION);

this.filterList=this.rsgservice.list.filter(e=>e.UserEmail!=this.currentUser);

this.currentUserId=this.rsgservice.list.find(e=>e.UserEmail==this.currentUser);

this.LoggedUserId=this.currentUserId.UserId;

}

selectedUser(user:Register)

{

{{debugger}}

this.ToUser=user.UserId;

this.toUserName=user.UserEmail;

this.msgservice.refreshList();

this.specificChat=this.msgservice.list.filter(e=>(e.fromUser==this.currentUserId.UserId && e.touser==user.UserId)||(e.fromUser==user.UserId && e.touser==this.currentUserId.UserId));

}

sendMessage(form:NgForm)

{

alert(form.value.msgcontent);

{{debugger}}

form.value.msgId=1;

form.value.fromUser=this.currentUserId.UserId;

form.value.touser=this.ToUser;

form.value.msgdate=formatDate(new Date(),'dd-MM-yyyy hh:mm:ss a','en-US','+0530');

this.msgservice.postEmployee(form.value).subscribe(res=>{

alert("Inserted Successfully");

this.resetForm(form);

});

this.msgservice.refreshList();

this.specificChat=this.msgservice.list.filter(e=>(e.fromUser==this.currentUserId.UserId && e.touser==this.ToUser)||(e.fromUser==this.ToUser && e.touser==this.currentUserId.UserId));

}

resetForm(form?:NgForm)

{

if(form!=null){

form.resetForm();

}

}

}

1. Usermessage.service.ts

import { Injectable } from '@angular/core';

import { Register } from './../shared/Register.Model';

import { Login } from './../shared/Login.Model';

import { MessageM } from './../shared/Message.Model';

import {HttpClient, HttpHeaders, HttpParams} from '@angular/common/http';

@Injectable({

providedIn: 'root'

})

export class UserMessagesService {

userMessages:MessageM;

list:MessageM[];

readonly rootURL="http://localhost:54319/api"

constructor(private http:HttpClient) { }

postEmployee(userMessages:MessageM)

{

{debugger}

return this.http.post(this.rootURL+'/messages',userMessages);

}

refreshList():void

{

{{debugger}}

this.http.get(this.rootURL+'/messages')

.toPromise().then(res=>this.list=res as MessageM[]);

}

putEmployee(userMessages:MessageM)

{

const headers = new HttpHeaders().set('content-type', 'application/json');

const params = new HttpParams().set('ID', userMessages.msgId.toString());

{

return this.http.put<MessageM>(this.rootURL+'/messages/'+userMessages.msgId,userMessages,{headers,params});

}

}

deleteEmployee(id:number)

{

return this.http.delete(this.rootURL+'/messages/'+id);

}

// validateEmployee(username:string,pswd:string)

// {

// return this.http.get(this.rootURL+'/registers/validateEmployee?UserEmail='+username+);

// }

}

1. Register.service.ts

import { Injectable } from '@angular/core';

import { Register } from './../shared/Register.Model';

import { Login } from './../shared/Login.Model';

import {HttpClient, HttpHeaders, HttpParams} from '@angular/common/http';

@Injectable({

providedIn: 'root'

})

export class RegisterService {

formData:Register;

loginData:Login;

list:Register[];

readonly rootURL="http://localhost:54319/api"

constructor(private http:HttpClient) { }

postEmployee(formData:Register)

{

return this.http.post(this.rootURL+'/registers',formData);

}

refreshList():void

{

this.http.get(this.rootURL+'/registers')

.toPromise().then(res=>this.list=res as Register[]);

}

putEmployee(formData:Register)

{

const headers = new HttpHeaders().set('content-type', 'application/json');

const params = new HttpParams().set('ID', formData.UserId.toString());

{

return this.http.put<Register>(this.rootURL+'/registers/'+formData.UserId,formData,{headers,params});

}

}

deleteEmployee(id:number)

{

return this.http.delete(this.rootURL+'/registers/'+id);

}

// validateEmployee(username:string,pswd:string)

// {

// return this.http.get(this.rootURL+'/registers/validateEmployee?UserEmail='+username+);

// }

}

1. Login.Model.ts

export class Login{

UserEmail:string;

Password:string;

}

1. Message.model.ts

export class MessageM{

msgId:number;

fromUser:number;

touser:number;

msgcontent:string;

msgdate:Date;

}

1. Register.Model.ts

export class Register{

UserId:number;

UserEmail:string;

Password:string;

Gender:string;

DateOfBirth:Date;

}

1. User.pipes.ts

import {Pipe, PipeTransform} from '@angular/core';

import { Register } from './Register.model';

@Pipe ({

name : 'userfilterPipe'

})

export class userpipes implements PipeTransform {

transform(users:Register[],username:string) {

if(!users || !username)

{

return users;

}

return users.filter((x=>x.UserEmail.toLowerCase().indexOf(username.toLowerCase())!==-1));

}

}