Client Management System Source Codes

**To add a new client**

***Register Module Component***

register.component.html

<head>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM" crossorigin="anonymous">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-js.js"></script>

</head>

<div class="gfg">

    <h1 class="first-txt">

       Create a Client

    </h1>

</div>

<h1 name="h1">{{message}}</h1>

<h3> Please fill in this form to create a client</h3>

<!--change made-->

<div>

    <label>ID:</label>

    <input type="number" name="id" [(ngModel)]="id"/>

</div>

<div>

    <label>Name:</label>

    <input type="text" [(ngModel)]="name" name="name" required>

    <span class="error" id="usernameError"></span>

</div>

<div>

    <label>Email:</label>

    <input type="email" name="email" [(ngModel)]="email"/>

</div>

<div>

    <label>Password:</label>

    <input type="password" name="password" [(ngModel)]="password"/>

</div>

<div>

    <label>Adress:</label>

    <input type="text" name="adress" [(ngModel)]="adress"/>

</div>

<button name="register" (click)="registerClient()">Register</button>

***TS file***

register.component.ts

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

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

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({

  selector: 'app-register',

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

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

})

export class RegisterComponent implements OnInit  {

id:number=0;

name:string='';

email:string='';

password:string='';

adress:string='';

message:string='';

  constructor(private http:HttpClient) { }

  ngOnInit(): void {

  }

  registerClient(){

    const client={

      id:this.id,

      name:this.name,

      email:this.email,

      password:this.password,

      adress:this.adress

      }

      if(this.name===null){

        this.message="name can not be empty" ;

        console.error('Error adding the client');

    }

    ;

    this.http.post('http://localhost:3000/registerClient',client)

    .subscribe((response:any)=>

    {this.message=response.message},

    (error)=>{console.error('Error adding the client',error);}

  );

}

}

* To View the List of registered clients there is view Client Component

*view-client.component.html*

{{message}}

<h1>View Client List</h1>

<table border="1">

<tr>

<th>ID</th>

<th>Name</th>

<th>Email</th>

<th>Adress</th>

<th>Password</th>

<th>Edit Action</th>

<th>Delete Action</th>

</tr>

<tr \*ngFor="let client of clients">

<td>{{client.id}}</td>

<td>{{client.name}}</td>

<td>{{client.email}}</td>

<td>{{client.adress}}</td>

<td>{{client.password}}</td>

<td><button routerLink="/edit/{{client.id}}">Edit</button></td>

<td><button (click)="deleteClient(client.id)">Delete</button></td>

</tr>

</table>

***TS File***

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

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

@Component({

  selector: 'app-view-client',

  templateUrl: './view-client.component.html',

  styleUrls: ['./view-client.component.css']

})

export class ViewClientComponent  implements OnInit {

  clients:any[]=[];

message:string='';

  constructor(private http:HttpClient) { }

  ngOnInit(): void {

    this.fetchClients();

      }

      fetchClients(){

        this.http.get('http://localhost:3000/getClients')

          .subscribe((response:any)=>

        {this.clients=response},

        (error)=>{console.error('Error fetching the clients',error);}

      );

      }

      deleteClient(id:number){

        if(confirm('Are you sure you want to delete this Client?')){

          this.http.delete('http://localhost:3000/deleteClient/'+id)

          .subscribe((response:any)=>

        {this.message=response.message;

          this.fetchClients();},

        (error)=>{console.error('Error deleting the client',error);}

      );

    }

    }

}

**To Edit any Client information any point in time is Edit Client Component**

***edit-client.component.html***

<h1>Edit Client</h1>

<div>

    <label>ID:</label>

    <input type="number" name="id" [(ngModel)]="id" [disabled]="true"/>

</div>

<div>

    <label>Name:</label>

    <input type="text" name="name" [(ngModel)]="name"/>

</div>

<div>

    <label>Email:</label>

    <input type="email" name="email" [(ngModel)]="email"/>

</div>

<div>

    <label>Password:</label>

    <input type="password" name="password" [(ngModel)]="password"/>

</div>

<div>

    <label>Adress:</label>

    <input type="text" name="adress" [(ngModel)]="adress"/>

</div>

<button (click)="updateClient()">Update Client</button>

**Ts file**

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

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

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

@Component({

  selector: 'app-edit-client',

  templateUrl: './edit-client.component.html',

  styleUrls: ['./edit-client.component.css']

})

export class EditClientComponent implements OnInit{

  id:number=0;

  name:string='';

  email:string='';

  password:string='';

  adress:string='';

  message:string='';

  constructor(private http:HttpClient,private route:ActivatedRoute,private router:Router) { }

  ngOnInit(): void {

    this.route.paramMap.subscribe(params=>{

      const idParam=params.get('id');

      if(idParam!==null){

        this.id=+idParam;

        this.fetchClient();

      }

      else{

        console.error("id is missing  or null");

      }

    })

  }

  fetchClient(){

    this.http.get('http://localhost:3000/getClient/'+this.id)

        .subscribe((response:any)=>

      {

        const client=response[0];

        this.name=client.name;

        this.email=client.email;

        this.password=client.password;

        this.adress=client.adress;

      },

      (error)=>{console.error('Error fetching the client',error);}

    );

  }

  updateClient(){

    const client={

      id:this.id,

      name:this.name,

      email:this.email,

      password:this.password,

      adress:this.adress

    };

    this.http.put('http://localhost:3000/updateclient',client)

    .subscribe((response:any)=>

    {this.message=response.message;this.router.navigate(['/view'])},

    (error)=>{console.error('Error updating the client',error);}

  );

  }

}

**To Schedule a Meeting with client**

***Create Meeting component***

###HTML

<head>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeRhO02iuK6FUUVM" crossorigin="anonymous">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bkz" crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-js.js"></script>

</head>

<div class="gfg">

    <h1 class="first-txt">

       Create a Client Meeting Schedule

    </h1>

</div>

{{message}}

<h3> Please fill in this form to create a Meeting with client</h3>

<div>

    <label>Meeting ID:</label>

    <input type="number" name="id" [(ngModel)]="meetingid"/>

</div>

<div>

    <label>Meeting Topic:</label>

    <input type="text" name="meeting\_topic" [(ngModel)]="meeting\_topic"/>

</div>

<div>

    <label>Number of Attendees</label>

    <input type="number" name="meeting\_member" [(ngModel)]="meeting\_member"/>

</div>

<div>

    <label>Meeting Date:</label>

    <input type="date" name="meetingdate" [(ngModel)]="meetingdate"/>

</div>

<div>

    <label>Start Time:</label>

    <input type="time" name="meetingtime" [(ngModel)]="meetingtime"/>

</div>

<div>

    <label>Client ID:</label>

    <input type="number" name="client\_id" [(ngModel)]="client\_id"/>

</div>

<button name="addMeeting" (click)="ScheduleMeeting()">Schedule Meeting</button>

###TS

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

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

@Component({

  selector: 'app-create-meeting',

  templateUrl: './create-meeting.component.html',

  styleUrls: ['./create-meeting.component.css']

})

export class CreateMeetingComponent implements OnInit{

  meetingid:number=0;

  meeting\_topic:string='';

  meeting\_member:number=0;

  meetingdate:string='';

  meetingtime:string='';

  client\_id:number=0;

  message:string='';

    constructor(private http:HttpClient) { }

    ngOnInit(): void {

    }

    ScheduleMeeting(){

      const meeting={

        meetingid:this.meetingid,

        meeting\_topic:this.meeting\_topic,

        meeting\_member:this.meeting\_member,

        meetingdate:this.meetingdate,

        meetingtime:this.meetingtime,

        client\_id:this.client\_id

      };

      this.http.post('http://localhost:3000/addMeeting',meeting)

      .subscribe((response:any)=>

      {this.message=response.message},

      (error)=>{console.error('Error adding the Meeting schedule',error);}

    );

}

}

**To View All the meetings Scheduled with many different Clients**

**View-Meeting Component**

**####HTML**

{{message}}

<h1>View Meeting Schedules</h1>

<table border="1">

<tr>

<th>Meeting ID</th>

<th>Meeting Topic</th>

<th>Number of Attendees</th>

<th>Meeting Date</th>

<th>Start Time</th>

<th>Client ID</th>

<th>Edit </th>

<th>Cancel meeting</th>

</tr>

<tr \*ngFor="let meeting of meetings">

<td>{{meeting.meetingid}}</td>

<td>{{meeting.meeting\_topic}}</td>

<td>{{meeting.meeting\_member}}</td>

<td>{{meeting.meetingdate}}</td>

<td>{{meeting.meetingtime}}</td>

<td>{{meeting.client\_id}}</td>

<td><button routerLink="/editMeeting/{{meeting.meetingid}}">Edit</button></td>

<td><button (click)="deleteMeeting(meeting.meetingid)">Cancel</button></td>

</tr>

</table>

###TS

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

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

@Component({

  selector: 'app-view-meeting',

  templateUrl: './view-meeting.component.html',

  styleUrls: ['./view-meeting.component.css']

})

export class ViewMeetingComponent implements OnInit{

  meetings:any[]=[];

message:string='';

  constructor(private http:HttpClient) { }

  ngOnInit(): void {

    this.fetchMeetings();

      }

      fetchMeetings(){

        this.http.get('http://localhost:3000/getMeetings')

          .subscribe((response:any)=>

        {this.meetings=response},

        (error)=>{console.error('Error fetching the Meetings',error);}

      );

      }

      deleteMeeting(meetingid:number){

        if(confirm('Are you sure you want to delete this meeting?')){

          this.http.delete('http://localhost:3000/deleteMeeting/'+meetingid)

          .subscribe((response:any)=>

        {this.message=response.message;

          this.fetchMeetings();},

        (error)=>{console.error('Error deleting the meeting',error);}

      );

    }

    }

}

**To Edit a prescheduled meeting at any point in time**

***Edit-Meeting component***

***###html***

<h1>Edit Meeting Information</h1>

<div>

    <label>Meeting ID:</label>

    <input type="number" name="meetingid" [(ngModel)]="meetingid" [disabled]="true"/>

</div>

<div>

    <label>Meeting Topic:</label>

    <input type="text" name="meeting\_topic" [(ngModel)]="meeting\_topic"/>

</div>

<div>

    <label>Number of Attendees</label>

    <input type="number" name="meeting\_member" [(ngModel)]="meeting\_member"/>

</div>

<div>

    <label>Meeting Date:</label>

    <input type="date" name="meetingdate" [(ngModel)]="meetingdate"/>

</div>

<div>

    <label>Start Time:</label>

    <input type="time" name="meetingtime" [(ngModel)]="meetingtime"/>

</div>

<div>

    <label>Client ID:</label>

    <input type="number" name="client\_id" [(ngModel)]="client\_id"/>

</div>

<button (click)="updateMeeting()">Update Meeting Schedule</button>

***###TS***

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

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

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

@Component({

  selector: 'app-edit-meeting',

  templateUrl: './edit-meeting.component.html',

  styleUrls: ['./edit-meeting.component.css']

})

export class EditMeetingComponent implements OnInit {

  meetingid:number=0;

  meeting\_topic:string='';

  meeting\_member:number=0;

  meetingdate:string='';

  meetingtime:string='';

  client\_id:number=0;

  message:string='';

  constructor(private http:HttpClient,private route:ActivatedRoute,private router:Router) { }

  ngOnInit(): void {

    this.route.paramMap.subscribe(params=>{

      const idParam=params.get('meetingid');

      if(idParam!==null){

        this.meetingid=+idParam;

        this.fetchMeeting();

      }

      else{

        console.error("meetingid is missing  or null");

      }

    })

  }

  fetchMeeting(){

    this.http.get('http://localhost:3000/getMeeting/'+this.meetingid)

        .subscribe((response:any)=>

      {

        const meeting=response[0];

        this.meeting\_topic=meeting.meeting\_topic;

        this.meeting\_member=meeting.meeting\_member;

        this.meetingdate=meeting.meetingdate;

        this.meetingtime=meeting.meetingtime;

        this.client\_id=meeting.client\_id;

      },

      (error)=>{console.error('Error fetching the meeting',error);}

    );

  }

  updateMeeting(){

    const meeting={

      meetingid:this.meetingid,

      meeting\_topic:this.meeting\_topic,

      meeting\_member:this.meeting\_member,

      meetingdate:this.meetingdate,

      meetingtime:this.meetingtime,

      client\_id:this.client\_id

    };

    this.http.put('http://localhost:3000/updateMeeting',meeting)

    .subscribe((response:any)=>

    {this.message=response.message;this.router.navigate(['/viewMeeting'])},

    (error)=>{console.error('Error updating the meeting',error);}

  );

  }

}

***App.component.html***

<ul><li><a routerLink="/register">Register Client</a></li>

  <li><a routerLink="/view">View Client</a></li>

  <li><a routerLink="/clientMeeting">Client Meeting Schedule</a>

    <li><a routerLink="/viewMeeting">View Client Meeting Schedule</a>

  <router-outlet></router-outlet>

***App.component.ts***

import { Component, ElementRef, ViewChild, AfterViewInit } from '@angular/core';

import \* as bootstrap from 'bootstrap';

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'course1Project';

}

***App.modules.ts***

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

import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { RegisterComponent } from './register/register.component';

import { ViewClientComponent } from './view-client/view-client.component';

import { EditClientComponent } from './edit-client/edit-client.component';

import { RouterModule, Routes } from '@angular/router';

import { FormsModule } from '@angular/forms';

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

import { CreateMeetingComponent } from './create-meeting/create-meeting.component';

import { ViewMeetingComponent } from './view-meeting/view-meeting.component';

import { EditMeetingComponent } from './edit-meeting/edit-meeting.component';

const routes:Routes=[

  {path:'',redirectTo:'/view',pathMatch:'full'},

  {path:'view',component:ViewClientComponent},

  {path:'',redirectTo:'/viewMeeting',pathMatch:'full'},

  {path:'viewMeeting',component:ViewMeetingComponent},

  {path:'clientMeeting',component:CreateMeetingComponent},

  {path:'register',component:RegisterComponent},

  {path:'',redirectTo:'/view',pathMatch:'full'},

  {path:'edit/:id',component:EditClientComponent},

  {path:'editMeeting/:meetingid',component:EditMeetingComponent}

  ]

@NgModule({

  declarations: [

    AppComponent,

    RegisterComponent,

    ViewClientComponent,

    EditClientComponent,

    CreateMeetingComponent,

    ViewMeetingComponent,

    EditMeetingComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    HttpClientModule,

    FormsModule,

    RouterModule.forRoot(routes)

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

**server.js**

//dependencies

const express=require('express')

const bodyParser=require('body-parser')

const cors=require('cors')

const mysql=require('mysql')

//define the express operation

const app=express();

const port=3000;

const bcrypt = require('bcrypt');

//defining the cors -cross origin by reciving the data in json format

app.use(cors());

app.use(bodyParser.json())

//establish the connection with the dB

const db=mysql.createConnection({

host:'localhost',

user:'root',

password:'root',

database:'db2'

});

//verifying whether db is connected or not

db.connect(err=>{

if(err){

    console.error('connection is not established with the dB',err);

}

else{

    console.log('Connected to db')

}

});

app.listen(port,()=>{console.log('server port estbalished on 3000')})

//to get all the clients

app.get('/getClients',(req,res)=>{

const sql='select \* from Client';

db.query(sql,(err,result)=>{

    if(err){

    console.error('Error in fetching the client',err);

    res.status(500).json({error:'An error occured'});

}else{

    res.status(200).json(result);

}

});

});

//to get a Client on id

app.get('/getClient/:id',(req,res)=>{

    const id=req.params.id;

    const sql='select \* from client where id=?';

    db.query(sql,[id],(err,result)=>{

        if(err){

        console.error('Error in fetching the client',err);

        res.status(500).json({error:'An error occured'});

    }else{

        res.status(200).json(result);

    }

    });

    });

//to insert client into db

app.post('/registerClient',async(req,res)=>{

    const {id,name,email,adress,password}=req.body;

    const sql='insert into client values(?,?,?,?,?)';

    // Generate a salt and hash the password

    const saltRounds = 10; // Number of salt rounds

    const hashedPassword = await bcrypt.hash(password, saltRounds);

    db.query(sql,[id,name,email,adress,hashedPassword],(err,result)=>{

        if(err){

            console.error('Error in adding the client',err);

            res.status(500).json({error:'An error occured'});

        }else{

            res.status(200).json({message:'client added successfully'});

        }

    });

    });

//updation of client

app.put('/updateClient',(req,res)=>{

    const {id,name,email,adress,password}=req.body;

    const sql='update client set name=?,email=?,adress=?,password=? where id=?';

    db.query(sql,[name,email,adress,password,id],(err,result)=>{

        if(err){

            console.error('Error in updating the client',err);

            res.status(500).json({error:'An error occured'});

        }else{

            res.status(200).json({message:'client updated successfully'});

        }

    });

    });

//deletion of client /deleteClient

app.delete('/deleteClient/:id',(req,res)=>{

    const id=req.params.id;

    const sql='delete from client where id=?';

    db.query(sql,[id],(err,result)=>{

        if(err){

        console.error('Error in deleting the client',err);

        res.status(500).json({error:'An error occured'});

    }else{

        res.status(200).json({message:'client deleted successfully'});

    }

    });

    });

    ///////////////////////meeting functions///////////////

    /////////to insert meeting details  into db

app.post('/addMeeting',(req,res)=>{

    const {meetingid,meeting\_topic,meeting\_member,meetingdate,meetingtime,client\_id}=req.body;

    const sql='insert into meeting values(?,?,?,?,?,?)';

    db.query(sql,[meetingid,meeting\_topic,meeting\_member,meetingdate,meetingtime,client\_id],(err,result)=>{

        if(err){

            console.error('Error in adding the meeting',err);

            res.status(500).json({error:'An error occured in adding meeting'});

        }else{

            res.status(200).json({message:'meeting added successfully'});

        }

    });

    });

//to get all the Meetings

app.get('/getMeetings',(req,res)=>{

    const sql='select \* from Meeting';

    db.query(sql,(err,result)=>{

        if(err){

        console.error('Error in fetching the client',err);

        res.status(500).json({error:'An error occured'});

    }else{

        res.status(200).json(result);

    }

    });

    });

//to get a meeting on meetingid

app.get('/getMeeting/:meetingid',(req,res)=>{

    const meetingid=req.params.meetingid;

    const sql='select \* from Meeting where meetingid=?';

    db.query(sql,[meetingid],(err,result)=>{

        if(err){

        console.error('Error in fetching the meeting',err);

        res.status(500).json({error:'An error in fetching meeting by id occured'});

    }else{

        res.status(200).json(result);

    }

    });

    });

    //updation of meeting

    app.put('/updateMeeting',(req,res)=>{

    const {meetingid,meeting\_topic,meeting\_member,meetingdate,meetingtime,client\_id}=req.body;

    const sql='update meeting set meeting\_topic=?,meeting\_member=?,meetingdate=?,meetingtime=?,client\_id=? where meetingid=?';

    db.query(sql,[meeting\_topic,meeting\_member,meetingdate,meetingtime,client\_id,meetingid],(err,result)=>{

        if(err){

            console.error('Error in updating the meeting',err);

            res.status(500).json({error:'An error occured'});

        }else{

            res.status(200).json({message:'meeting updated successfully'});

        }

    });

    });

    //deletion of meeting /deleteMeeting

app.delete('/deleteMeeting/:meetingid',(req,res)=>{

    const meetingid=req.params.meetingid;

    const sql='delete from meeting where meetingid=?';

    db.query(sql,[meetingid],(err,result)=>{

        if(err){

        console.error('Error in deleting the Meeting',err);

        res.status(500).json({error:'An error occured'});

    }else{

        res.status(200).json({message:'Meeting deleted successfully'});

    }

    });

    });