**FrontEnd(Angular)**

1. **Comp**
2. **Home**

**Home.component.html**

<div class="container" style="margin: 5px 5px;">

    <h3 style="margin:auto;width: 300px;">Create New User</h3>

    <div class="row">

        <div class="col md-3">

            <button class="btn btn-success">New Customer</button>

        </div>

        <div class="col md-3">

            <button class="btn btn-success">New Employee</button>

        </div>

        <div class="col md-3">

            <button class="btn btn-success">New Manager</button>

        </div>

    </div>

</div>

[**Home.component.ts**](http://Home.component.ts)

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

@Component({

  selector: 'app-home',

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

  styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

1. **Login**

**Login.component.html**

<section class="vh-50 gradient-custom">

    <div class="container  py-5 h-100 ">

        <div class="row d-flex justify-content-center align-items-center h-100 ">

            <div class="col-12 col-md-8 col-lg-6 col-xl-5">

                <div class="card bg-light " style="border-radius: 1rem ; width: 28rem; height: 40rem;  ">

                    <div class="card-body p-5 text-center"> <img src="https://www.pointfranchise.co.uk/images/zoom/articles/telecom-company-franchises.jpeg" class="card-img-top" alt="Fissure in Sandstone" />

                        <div class="mb-md-5 mt-md-4 pb-5">

                            <h3 class="fw-bold mb-2 text-uppercase">ABC Telecom Ltd.</h3>

                            <!-- <h2 class="fw-bold mb-2 text-uppercase">Login</h2> -->

                            <div></div>

                            <p class="text-dark-50 mb-5 ">Please enter your email and password! </p>

                            <form [formGroup]="loginForm" (ngSubmit)="loginhandler()">

                                <div class="form-outline form-white mb-3">

                                    <input type="email" formControlName='email' id="typeEmailX" placeholder='Enter Email' class="form-control form-control-lg" />

                                    <!--<label class="form-label"  for="typeEmailX">Email </label> -->

                                </div>

                                <div class="form-outline form-white mb-4">

                                    <input formControlName='password' type="password" id="typePasswordX" placeholder='Enter Password' class="form-control form-control-lg" />

                                    <!--<label class="form-label" for="typePasswordX">Password</label>-->

                                </div>

                                <button class="btn btn-outline-dark  btn-lg px-5" type="submit">Login</button>

                            </form>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </div>

</section>

**Login.component.css**

.container

{

    background-image: url('/assets/images/back.jpg');

    background-position: center;

    background-repeat: no-repeat;

    background-size: cover;

}

**Login.component.ts**

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

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

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

import { UserDto } from 'src/app/common/user-dto';

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

  loginForm:FormGroup;

  constructor(

    private formBulder:FormBuilder,

    private loginService:LoginService,

     private userService:UserService,

     private router:Router

  ) { }

  ngOnInit(): void {

    this.loginForm=this.formBulder.group({

      email:new FormControl('',[Validators.required,Validators.email]),

        password:new FormControl('',Validators.required)

    });

  }

  loginhandler(){

    let userDto: UserDto=this.loginForm.value;

    this.loginService.checkLogin(userDto).subscribe({

      next:response=>{

        if(response!=null){

          alert( `Logged in successfully! User Name: ${response.fullName}`);

          if(response.role=='customer'){

            this.userService.getCustomerByUserId(response.id).subscribe(data=>{

              this.userService.loggedInCustomer=data;

              this.router.navigateByUrl("ticket-list");

            })

          }

         else if(response.role=='engineer'){

            this.userService.getEngineerByUserId(response.id).subscribe(data=>{

              this.userService.loggedInEngineer=data;

              this.router.navigateByUrl("ticket-list");

            })

          }

          else if(response.role=='manager'){

            this.userService.getManagerByUserId(response.id).subscribe(data=>{

              this.userService.loggedInManager=data;

              this.router.navigateByUrl("ticket-list");

            })

          }

          else if(response.role=='admin'){

            this.userService.loggedInCustomer;

            this.loginService.loggedInUser.subscribe(data=>{

              console.log(data);

              this.router.navigateByUrl("home");

            })

          }

          else{

              alert("something went wrong");

          }

        }

      },

      error:err=>{

        // alert(`There was an error: ${err.message}`);

        alert("incorrect credentials")

      }

    })

  }

}

1. **Core**

**Core.component.html**

    <nav \*ngIf="loggedInUser!=null" class="navbar navbar-expand-lg navbar-dark bg-dark">

        <div class="container-fluid">

            <a href="#" class="navbar-brand">ABC Telecom Ltd.</a>

            <button type="button" class="navbar-toggler" data-bs-toggle="collapse" data-bs-target="#navbarCollapse1">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarCollapse1">

                <div class="navbar-nav">

                    <a routerLink="/home" class="nav-item nav-link" routerLinkActive="active-link">Home</a>

                    <li class="nav-item dropdown">

                      <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown" href="#" role="button" aria-expanded="false">Tickets</a>

                      <ul class="dropdown-menu">

                        <li><a class="dropdown-item" [routerLink]="['/ticket-list']" >All Tickets</a></li>

                        <div \*ngIf="loggedInUser.role=='customer'">

                          <li><hr class="dropdown-divider"></li>

                           <li >

                          <a class="dropdown-item" routerLink="/ticket">Create New</a>

                          </li>

                        </div>

                      </ul>

                    </li>

                    <div \*ngIf="loggedInUser.role!='customer'">

                      <li  class="nav-item dropdown">

                        <a class="nav-link dropdown-toggle" data-bs-toggle="dropdown" href="#" role="button" aria-expanded="false">Users</a>

                        <ul class="dropdown-menu">

                          <li><a class="dropdown-item" [routerLink]="['/users','customer']" >Customers</a></li>

                          <li><a class="dropdown-item" [routerLink]="['/users','engineer']">Engineers</a></li>

                          <li><a class="dropdown-item" [routerLink]="['/users','manager']">Managers</a></li>

                          <div \*ngIf="loggedInUser.role=='admin'">

                          <li><hr class="dropdown-divider"></li>

                          <li ><a class="dropdown-item" routerLink="/create">New User</a></li>

                        </div>

                        </ul>

                      </li>

                    </div>

                    <!-- <a routerLink="/create" class="nav-item nav-link">Users</a> -->

                    <a routerLink="/logout" class="nav-item nav-link">Logout</a>

                     <div class="profile">

                </div>

                </div>

                <!-- <form class="d-flex ms-auto">

                    <input type="text" class="form-control me-sm-2" placeholder="Search">

                    <button type="submit" class="btn btn-outline-light">Search</button>

                </form> -->

            </div>

            <div class="profile-icon">

              <h4 class="profile-name">{{loggedInUser.fullName}}</h4>

              <svg width="32px" height="32px" viewBox="0 0 32 32" enable-background="new 0 0 32 32" id="Stock\_cut" version="1.1" xml:space="preserve" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"><desc/><g><circle cx="16" cy="16" fill="none"  r="15" stroke="#c9c8c7" stroke-linejoin="round" stroke-miterlimit="10" stroke-width="2"/><path d="M26,27L26,27   c0-5.523-4.477-10-10-10h0c-5.523,0-10,4.477-10,10v0" fill="none" stroke="#c9c8c7" stroke-linejoin="round" stroke-miterlimit="10" stroke-width="2"/><circle cx="16" cy="11" fill="none" r="6" stroke="#c9c8c7" stroke-linejoin="round" stroke-miterlimit="10" stroke-width="2"/></g></svg>

            </div>

        </div>

    </nav>

**Core.component.ts**

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

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

import { User } from 'src/app/common/user';

import { LoginService } from 'src/app/services/login.service';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-core',

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

  styleUrls: ['./core.component.css']

})

export class CoreComponent implements OnInit {

  loggedInUser:User=null;

  constructor(private userService:UserService, private loginService:LoginService,private router:Router) { }

  ngOnInit(): void {

    if(this.loginService.loggedInUser!=null){

      this.loginService.loggedInUser.subscribe(data=>{

        this.loggedInUser=data;

      })

    }

    else{

      this.router.navigateByUrl('login');

    }

  }

}

1. **Assign-engineer**

**ae.component.html**

<div class="conatiner">

    <h1>Assign Engineer</h1>

    <label for="">

        <!-- {{ticket.id}} -->

    </label>

    <select id=""></select>

</div>

**ae.component.ts**

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

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

import { Ticket } from 'src/app/common/ticket';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-assign-engineer',

  templateUrl: './assign-engineer.component.html',

  styleUrls: ['./assign-engineer.component.css']

})

export class AssignEngineerComponent implements OnInit {

  ticket:Ticket;

  constructor(private userService:UserService,private router:Router) { }

  ngOnInit(): void {

    this.router

  }

}

1. **Create-user**

**Create-user.component.html**

<div class="container">

    <div class="row">

        <div class="col-md-8 offset-md-2">

            <h1>{{formHeading}}</h1>

            <form [formGroup]="userForm" (ngSubmit)="submitHandle()" class="row g-3">

                <div formGroupName="user">

                    <input type="hidden" formControlName='id'>

                    <div class="form-group">

                        <label class="control-lable" for="fullName"> Full Name\* </label>

                        <input id="fullName" formControlName="fullName" class="form-control"> </div>

                    <div class="form-group">

                        <label class="control-lable" for="phone"> phone\* </label>

                        <input id="phone" formControlName="phone" class="form-control"> </div>

                    <div class="form-group">

                        <label class="control-lable" for="email"> email\* </label>

                        <input id="email" formControlName="email" class="form-control"> </div>

                    <div class="form-group">

                        <label class="control-lable" for="password"> password\* </label>

                        <input type="password" id="password" formControlName="password" class="form-control"> </div>

                    <div class="form-group">

                        <label for="role" class="control-label"> Role\* </label>

                        <select id="role" formControlName="role" (change)="roleHandler()" class="form-select">

                            <option value="customer">Customer</option>

                            <option value="engineer">Engineer</option>

                            <option value="manager">Manager</option>

                        </select>

                    </div>

                </div>

                <!-- Form for customer -->

                <!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

                <div \*ngIf="role=='customer'">

                    <div formGroupName="customer">

                        <input type="hidden" formControlName='id'>

                        <div class="form-group">

                            <label class="control-lable" for="address"> Address\* </label>

                            <input id="address" formControlName="address" class="form-control"> </div>

                        <div class="form-group">

                            <label class="control-lable" for="zipcode"> Zipcode\* </label>

                            <select formControlName='zipcode' id="zipcode" class="form-select">

                                <option \*ngFor="let zipcode of zipcodes" [ngValue]="zipcode">{{zipcode}}</option>

                            </select>

                        </div>

                        <div class="form-group">

                            <label class="control-lable" for="address"> Service Type\* </label>

                            <ng-multiselect-dropdown formControlName="serviceType" [placeholder]="'service type'" [settings]="dropdownSettings" [data]="serviceList" (onSelect)="onItemSelect($event)" (onSelectAll)="onSelectAll($event)"> </ng-multiselect-dropdown>

                        </div>

                    </div>

                </div>

                <!-- Form for engineer -->

                <!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

                <div \*ngIf="role=='engineer'">

                    <div formGroupName="engineer">

                        <input type="hidden" formControlName='id'>

                        <div class="form-group">

                            <label class="control-lable" for="zipcode"> Zipcode\* </label>

                            <select formControlName='zipcode' id="zipcode" class="form-select">

                                <option \*ngFor="let zipcode of zipcodes" [ngValue]="zipcode">{{zipcode}}</option>

                            </select>

                        </div>

                    </div>

                </div>

                <!-- Form for Manager -->

                <!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

                <div \*ngIf="role=='manager'">

                    <div formGroupName="manager">

                        <input type="hidden" formControlName='id'>

                        <div class="form-group">

                            <label class="control-lable"> Zipcodes Assigned\* </label>

                            <ng-multiselect-dropdown formControlName="zipcode" [placeholder]="'zipcode'" [settings]="dropdownSettings" [data]="zipcodeList" (onSelect)="onItemSelect($event)" (onSelectAll)="onSelectAll($event)"> </ng-multiselect-dropdown>

                        </div>

                    </div>

                </div>

                <div class="form-group">

                    <div class="col-md-4 offset-md-2">

                        <div class="row">

                            <div class="col">

                                <button class="btn btn-dark">Submit</button>

                            </div>

                            <div class="col">

                                <button type="submit" class="btn btn-dark">Cancel</button>

                            </div>

                        </div>

                    </div>

                </div>

            </form>

        </div>

    </div>

</div>

**Create-user.component.ts**

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

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

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

import { IDropdownSettings } from 'ng-multiselect-dropdown';

import { map } from 'rxjs/operators';

import { Customer } from 'src/app/common/customer';

import { Engineer } from 'src/app/common/engineer';

import { Manager } from 'src/app/common/manager';

import { MultiSelect } from 'src/app/common/multi-select';

import { User } from 'src/app/common/user';

import { Zipcode } from 'src/app/common/zipcode';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-create-user',

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

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

})

export class CreateUserComponent implements OnInit {

  userForm:FormGroup;

  formHeading:string='New user';

  role:string;

  zipcodes:string[]=[];

  constructor(private formBuilder:FormBuilder, private userService:UserService,private route:ActivatedRoute) { }

  serviceList;

  zipcodeList;

  dropdownSettings;

  userId:number=0;

  userRole:string='';

  ngOnInit():void {

    this.serviceList = this.getData();

    this.zipcodeList=this.getZipcodes();

    this.dropdownSettings = {

      singleSelection: false,

      idField: 'item\_id',

      textField: 'item\_text',

      selectAllText: 'Select All',

      unSelectAllText: 'UnSelect All'

    };

    this.userForm=this.formBuilder.group({

      user:this.formBuilder.group({

        id:[''],

        fullName: new FormControl("",[Validators.required]),

        role:new FormControl("",[Validators.required]),

        email:new FormControl("",[Validators.required,Validators.email]),

        phone:new FormControl("",[Validators.required]),

        password:new FormControl("",[Validators.required])

      }),

      customer:this.formBuilder.group({

        id:[''],

        serviceType:new FormControl("",[Validators.required]),

        address:new FormControl("",[Validators.required]),

        zipcode:new FormControl("",[Validators.required]),

      }),

      engineer:this.formBuilder.group({

        id:[''],

        zipcode:new FormControl("",[Validators.required])

      }),

      manager:this.formBuilder.group({

        id:[''],

        zipcode:new FormControl("",[Validators.required])

      })

    });

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

      this.editUser();

    })

  }

  editUser(){

    if(this.route.snapshot.paramMap.has('id')){

      this.formHeading='Edit User';

      this.userId=+this.route.snapshot.paramMap.get('id');

      this.userRole=this.route.snapshot.paramMap.get('role');

      let editUserForm = this.userForm.get('user');

      // If role is Customer then set Customer form

      if(this.userRole=='customer'){

       this.editCustomer(editUserForm);

      }

      if(this.userRole=='engineer'){

        this.editEngineer(editUserForm);

      }

      if(this.userRole=='manager'){

        this.editManager(editUserForm);

      }

    }

  }

  editManager(editUserForm: AbstractControl) {

    let editManagerForm=this.userForm.get('manager');

    this.userService.getManagerById(this.userId).subscribe(data=>{

      editUserForm.setValue(data.user);

      this.roleHandler();

      editManagerForm.get('zipcode').setValue(data.zipcode);

      editManagerForm.get('id').setValue(data.id);

    })

  }

  editEngineer(editUserForm: AbstractControl) {

    let editEngineerForm=this.userForm.get('engineer');

    this.userService.getEngineerById(this.userId).subscribe(data=>{

      editUserForm.setValue(data.user);

      this.roleHandler();

      editEngineerForm.get('zipcode').setValue(data.zipcode);

      editEngineerForm.get('id').setValue(data.id);

    })

  }

  editCustomer(editUserForm:AbstractControl){

    let editCustomerForm = this.userForm.get('customer')

    this.userService.getCustomerById(this.userId).subscribe(data=>{

      console.log(data);

      editUserForm.setValue(data.user);

      this.roleHandler();

      editCustomerForm.get('address').setValue(data.address);

      editCustomerForm.get('serviceType').setValue(data.serviceType);

      editCustomerForm.get('zipcode').setValue(data.zipcode);

      editCustomerForm.get('id').setValue(data.id);

     /\*\*Instead of the below data assigned userform as data.user

      \*  editUserForm.get('fullName').setValue(data.user.fullName);

      editUserForm.get('role').setValue(data.user.role);

      editUserForm.get('phone').setValue(data.user.phone);

      editUserForm.get('email').setValue(data.user.email);

      editUserForm.get('password').setValue(data.user.password);

      \*/

      this.formHeading='Edit User'

    })

  }

  getZipcodes() : Array<MultiSelect>{

    let multiSelect:MultiSelect[]=[];

    this.userService.getZipcodes().subscribe((data: string[])=>{

      this.zipcodes=data;

      console.log(this.zipcodes);

      for (let i = 0; i < data.length; i++) {

        let theMultiSelect = new MultiSelect();

        theMultiSelect.item\_id=i;

        theMultiSelect.item\_text=data[i];

        theMultiSelect.group='z';

        multiSelect.push(theMultiSelect);

      }

    })

    return multiSelect;

  }

  getData() : Array<MultiSelect>{

    return [

      { item\_id: 1, item\_text: 'landline', group : 'S' },

      { item\_id: 2, item\_text: 'phone', group : 'S' },

      { item\_id: 3, item\_text: 'fiber optic', group : 'S' },

    ];

  }

  onItemSelect(item: any) {

    // console.log(item);

  }

  onSelectAll(items: any) {

    // console.log(items);

  }

  roleHandler(){

    this.role= this.userForm.get('user').value.role;

    // console.log("==========>Role Selected: "+this.role);

  }

  submitHandle(){

    // console.log(this.userForm.get('user').value);

    // console.log(this.userForm.get('customer').value);

    let user:User=this.userForm.get('user').value;

  //  For Customer Role

  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

   if(this.role=='customer'){

    const services:MultiSelect[]=this.userForm.get('customer').value.serviceType;

    let serviceToString:string[]=[];

    services.map(data=>serviceToString.push(data.item\_text));

    // serviceToString.map(data=>console.log(data));

    let customer:Customer=new Customer();

    customer =this.userForm.get('customer').value;

    customer.serviceType=serviceToString;

    customer.user=user;

    this.userService.createCustomer(customer).subscribe({

      next:response=>{

        alert( `Customer created successfully! Customer ID: ${response}`);

      },

      error:err=>{

        alert(`There was an error: ${err.message}`);

      }

    })

   }

   //  For Engineer Role

  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

    if(this.role=='engineer'){

      let engineer: Engineer = new Engineer();

      engineer=this.userForm.get('engineer').value;

      engineer.user=user;

      this.userService.createEngineer(engineer).subscribe({

        next:response=>{

          alert( `Engineer created successfully! Engineer ID: ${response}`);

        },

        error:err=>{

          alert(`There was an error: ${err.message}`);

        }

      })

    }

  //  For Manager Role

  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  if(this.role=="manager"){

    let manager:Manager=new Manager();

    const pincodes:MultiSelect[]=this.userForm.get('manager').value.zipcode;

    let pincodesToString:string[]=[];

    pincodes.map(data=>pincodesToString.push(data.item\_text));

    manager=this.userForm.get('manager').value;

    manager.user=user;

    manager.zipcode=pincodesToString;

    this.userService.createManager(manager).subscribe({

      next:response=>{

        alert( `Manager created successfully! Manager ID: ${response}`);

      },

      error:err=>{

        alert(`There was an error: ${err.message}`);

      }

    })

  }

  }

}

**Feedback.component.html**

<div class="container">

    <form [formGroup]="feedbackForm" (ngSubmit)="submitHandler()">

        <div class="row justify-content-around">

            <div class="col-sm-3">

                <h4>Feedback - Ticket# {{ticket.id}} </h4>

                <div class="form-group">

                    <label for="rating" class="control-label">

                      Rating

                    </label>

                    <select id="rating" class="form-select" formControlName="rating">

                      <option value="1">1</option>

                      <option value="2">2</option>

                      <option value="3">3</option>

                      <option value="4">4</option>

                      <option value="5">5</option>

                    </select>

                  </div>

            </div>

        </div>

        <div class="row justify-content-around">

            <div class="col-sm-3">

                <div class="form-group">

                    <label class="control-lable" for="comments">

                      Comments

                    </label>

                    <textarea id="comments" formControlName="comments" class="form-control" cols="30" rows="5"></textarea>

                  </div>

            </div>

        </div>

        <br>

        <div class="row justify-content-around">

            <div class="col-sm-3">

                <button class="btn btn-dark">Submit</button>

            </div>

        </div>

        <br>

        <div class="row justify-content-around">

            <div class="col-sm-3">

                <!-- <button class="btn btn-dark">Cancel</button> -->

            </div>

        </div>

    </form>

</div>

**Feedback.component.ts**

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

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

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

import { Feedback } from 'src/app/common/feedback';

import { Ticket } from 'src/app/common/ticket';

import { TicketService } from 'src/app/services/ticket.service';

@Component({

  selector: 'app-feedback',

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

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

})

export class FeedbackComponent implements OnInit {

  ticket:Ticket;

  feedbackForm: FormGroup;

  constructor(private ticketService:TicketService,private router:ActivatedRoute, private formBuilder:FormBuilder) { }

  ngOnInit(): void {

    if(this.router.snapshot.paramMap.has('ticketId'))

    this.ticketService.getTicketById(+this.router.snapshot.paramMap.get('ticketId')).subscribe(data=>{

      this.ticket=data;

    });

    this.feedbackForm=this.formBuilder.group({

      id:[''],

      rating:[''],

      comments:['']

    });

  }

  submitHandler(){

    let feedback:Feedback=new Feedback();

    feedback = this.feedbackForm.value;

    feedback.ticket = this.ticket;

    console.log(feedback);

    this.ticketService.saveFeedback(feedback).subscribe({

      next:res=>{

        alert("Feedback submitted successfully"+res);

      },

      error:err=>{

        alert("There was an error"+err.message);

      }

    })

  }

}

1. **Ticket**

**Ticket.component.html**

<div class="container">

    <div class="row">

      <div class="col-md-8 offset-md-2">

        <h1 \*ngIf="ticket==null">New Complaint</h1>

        <h1 \*ngIf="ticket!=null">Ticket# {{ticket.id}}</h1>

        <div \*ngIf="feedback!=null">

          <h3>Feedback</h3>

          <div class="feedback"  style="border: 2 solid;">

            <p><b>Submitted On:</b> {{feedback.createdOn}}</p>

            <p><b>Rating:</b> {{feedback.rating}}</p>

            <p><b>Comments:</b> {{feedback.comments}}</p>

          </div>

        </div>

        <form [formGroup]="complaintForm" (ngSubmit)="submitHandle()" class="row g-3">

          <div formGroupName="ticket">

            <input type="hidden" formControlName='id'>

          <div formGroupName="customer">

            <input type="hidden" formControlName='id'>

            <div class="form-group" formGroupName="user">

            <label class="control-lable" for="fullName">

                Full Name\*

            </label>

            <input id="fullName" formControlName="fullName" class="form-control" required="required" autofocus="autofocus">

          </div>

        </div>

        <div class="form-group">

          <label class="control-lable" for="mobile">

              Mobile\*

          </label>

          <input id="mobile" formControlName="mobile" class="form-control" required="required" autofocus="autofocus">

        </div>

        <div class="form-group">

          <label for="serviceType" class="control-label">

            Service Type\*

          </label>

          <select id="serviceType" formControlName="serviceType" class="form-select">

            <option \*ngFor="let service of loggedInCustomer.serviceType" [ngValue]='service'>

                {{service}}

            </option>

      <!-- In edit mode service type will not be auto populated so manually doing it -->

             <option \*ngIf="serviceType.length>1" value="{{serviceType}}">{{serviceType}}</option>

          </select>

        </div>

        <div class="form-group">

          <label for="issueType" class="control-label">

            Issue Type\*

          </label>

          <select id="issueType" class="form-select" formControlName="issueType">

            <option value="Cannot make calls">Cannot make calls</option>

            <option value="Cannot receive calls">Cannot receive calls</option>

            <option value="Cannot make/receive calls">Cannot make/receive calls</option>

            <option value="other">Other</option>

          </select>

        </div>

        <div class="form-group">

          <label class="control-lable" for="description">

             Detailed Description\*

          </label>

          <textarea id="description" formControlName="description" class="form-control" cols="30" rows="5"></textarea>

        </div>

        <div class="form-group">

          <label class="control-lable" for="address">

              Address\*

          </label>

          <input id="address" formControlName="address" class="form-control" required="required" autofocus="autofocus">

        </div>

        <div class="form-group">

          <label class="control-lable" for="zipcode">

            Zipcode\*

          </label>

          <input id="zipcode" formControlName="zipcode" class="form-control" required="required" autofocus="autofocus">

        </div>

        <div \*ngIf="loggedInUser.role=='manager'" formGroupName='engineer'>

          <div class="form-group">

            <label for="">Assign To</label>

            <select formControlName="id" class="form-select">

              <option \*ngFor="let engineer of engineers" [ngValue]="engineer.id">

                {{engineer.user.email}}

              </option>

            </select>

            <!-- <input type="text" (keyup)="getAutoComplList($event)" class="form-control" id='dynamicEngineerIds'  list="dynamicEngineerIds" />

            <datalist id="dynamicEngineerIds">

              <option \*ngFor="let engineer of engineers" [value]="engineer.user.email">{{engineer.user.fullName}}</option>

            </datalist> -->

          </div>

        </div>

        <div \*ngIf="ticket!=null" class="form-grop">

          <label for="status">Status</label>

          <select id="status" class="form-select" formControlName="status">

            <option value="RAISED">RAISED</option>

            <option value="WIP">WIP</option>

            <option value="RESOLVED">RESOLVED</option>

            <option value="ESCALATED">ESCALATED</option>

          </select>

        </div>

  </div>

 <!-- Displaying notes -->

  <div class="work-notes" \*ngFor="let note of notes" style="border: 2 solid;">

    <p><b>Comments</b></p>

    <p><b>Updated On:</b> {{note.createdOn}}</p>

    <p><b>Updated By:</b> {{note.updatedBy}}</p>

    <p><b>Comments:</b> {{note.comments}}</p>

  </div>

   <!-- Form Group Notes -->

  <div \*ngIf="feedback==null" formGroupName='notes'>

    <div \*ngIf="ticket!=null" class="form-group">

      <label class="control-lable" for="notes">

         Add Comments

      </label>

      <textarea id="notes" formControlName="comments" class="form-control" cols="30" rows="5"></textarea>

    </div>

   </div>

   <div \*ngIf="feedback==null" class="form-group">

    <div class="col-md-4 offset-md-2">

    <div class="row">

      <div class="col">

        <button class="btn btn-dark">Submit</button>

      </div>

      <div class="col">

        <button type="submit" class="btn btn-dark">Cancel</button>

      </div>

    </div>

</div>

</div>

        </form>

      </div>

    </div>

  </div>

**Ticket.component.ts**

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

import { tick } from '@angular/core/testing';

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

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

import { timestamp } from 'rxjs/operators';

import { Customer } from 'src/app/common/customer';

import { Engineer } from 'src/app/common/engineer';

import { Feedback } from 'src/app/common/feedback';

import { Manager } from 'src/app/common/manager';

import { Notes } from 'src/app/common/notes';

import { Ticket } from 'src/app/common/ticket';

import { User } from 'src/app/common/user';

import { LoginService } from 'src/app/services/login.service';

import { TicketService } from 'src/app/services/ticket.service';

@Component({

  selector: 'app-ticket',

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

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

})

export class TicketComponent implements OnInit {

  complaintForm: FormGroup;

  loggedInUser:User=null;

  loggedInCustomer:Customer=new Customer();

  loggedInEngineer:Engineer;

  loggedInmanagre:Manager;

  serviceType:string='';

  assignedTo:string='';

  ticket:Ticket=null;

  allEngineers:Engineer[];

  engineers:Engineer[];

  notes:Notes[]=null;

  lastkeydown1: number = 0;

  subscription: any;

  feedback:Feedback=null;

  constructor(

    private formBuilder: FormBuilder,

    private ticketService:TicketService,

    private loginService:LoginService,

    private route:ActivatedRoute

  ) { }

// End of constructor

  ngOnInit(): void {

    this.complaintForm = this.formBuilder.group({

      ticket:this.formBuilder.group({

        id:[''],

        mobile:[''],

        serviceType:[''],

        issueType:[''],

        description:[''],

        zipcode:[''],

        address:[''],

        createdOn:[''],

        updatedOn:[''],

        status:[''],

        // notes:[''],

        customer:this.formBuilder.group({

          id:[''],

          serviceType:new FormControl("",[Validators.required]),

          address:new FormControl("",[Validators.required]),

          zipcode:new FormControl("",[Validators.required]),

          user:this.formBuilder.group({

              id:[''],

              fullName: new FormControl("",[Validators.required]),

              role:new FormControl("",[Validators.required]),

              email:new FormControl("",[Validators.required,Validators.email]),

              phone:new FormControl("",[Validators.required]),

              password:new FormControl("",[Validators.required])

          })

        }),

        // engineer:['']

        engineer:this.formBuilder.group({

            id:[''],

            zipcode:new FormControl("",[Validators.required]),

            user:this.formBuilder.group({

                id:[''],

                fullName: new FormControl("",[Validators.required]),

                role:new FormControl("",[Validators.required]),

                email:new FormControl("",[Validators.required,Validators.email]),

                phone:new FormControl("",[Validators.required]),

                password:new FormControl("",[Validators.required])

            })

        }),

      }),

      notes:this.formBuilder.group({

        id:[''],

        ticket:[''],

        comments:[''],

        createdOn:[''],

        updatedBy:[''],

        isCustomer:['']

      })

    });

    this.loginService.loggedInUser.subscribe(data=>{

      this.loggedInUser=data;

      if(this.route.snapshot.paramMap.has('id')){

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

          this.getTicketDetails();

        })

      }

      else if(this.loggedInUser.role="customer"){

        console.log("line 103");

        this.getTicketDetailsByCustomer();

      }

    });

  }

  // End ngOnInit

  getTicketDetails() {

    //  "+" prefixed here will cast string to a number

    this.ticketService.getTicketById(+this.route.snapshot.paramMap.get('id')).subscribe(data=>{

      this.ticket=data;

      this.ticketService.getFeedback(data.id).subscribe(data=>{

        this.feedback=data;

      })

     this.ticketService.getNotesByTicketId(data.id).subscribe(data=>{

      this.notes=data;

     });

      console.log(this.ticket.serviceType+" at line 121");

      this.serviceType=this.ticket.serviceType;

      if(this.ticket.engineer!=null){

        this.assignedTo=this.ticket.engineer.user.email;

        this.complaintForm.get('ticket').get('engineer').patchValue(this.ticket.engineer);

        console.log("At line 136===>"+this.assignedTo);

      }

      this.complaintForm.get('ticket').patchValue({

        id:this.ticket.id,

        mobile:this.ticket.mobile,

        serviceType:this.ticket.serviceType,

        issueType:this.ticket.issueType,

        description:this.ticket.description,

        zipcode:this.ticket.zipcode,

        address:this.ticket.address,

        createdOn:this.ticket.createdOn,

        updatedOn:this.ticket.updatedOn,

        status:this.ticket.status,

        customer:this.ticket.customer

      });

      this.complaintForm.get('ticket').disable();

      if(this.loggedInUser.role=='engineer'){

        this.loginService.loggedInEngineer.subscribe(data=>{

          this.loggedInEngineer=data;

          if(this.ticket.engineer.id==data.id){

            this.complaintForm.get('ticket').get('status').enable({onlySelf:true});

          }

        });

      }

      else if(this.loggedInUser.role=='manager'){

        console.log("line 153");

        this.loginService.loggedInManager.subscribe(data=>{

          this.loggedInmanagre=data;

          console.log("line 157");

          this.complaintForm.get('ticket').get('engineer').enable({onlySelf:true});

          console.log("Enabled Engineer");

        });

        // populate Engineers for auto complete textbox

        this.ticketService.getEngineersByZipcode(this.ticket.zipcode).subscribe(data=>{

          this.allEngineers=data;

          this.engineers=data;

        })

      }

      else if(this.loggedInUser.role=='customer'){

        this.loginService.loggedInCustomer.subscribe(data=>this.loggedInCustomer=data);

      }

      console.log(this.ticket.serviceType+" at line 125");

      // this.complaintForm.get('ticket').get('serviceType').setValue(ticket.serviceType);

    })

  }

  getTicketDetailsByCustomer() {

    console.log("line 110");

    this.loginService.loggedInUser.subscribe(data=>{

      console.log("line 113");

      this.loggedInUser=data;

        let ticketFrom=this.complaintForm.get('ticket');

        this.loginService.loggedInCustomer.subscribe(data=>{

          this.loggedInCustomer=data;

          // assign all the customer details to ticket in order to auto fill the form

          console.log("line 119");

          ticketFrom.get('customer').get('user').get("fullName").setValue(data.user.fullName);

          ticketFrom.get('customer').get('user').get("fullName").disable();

          ticketFrom.get('mobile').setValue(data.user.phone);

          ticketFrom.get('address').setValue(data.address);

          ticketFrom.get('zipcode').setValue(data.zipcode);

        })

    })

  }

  // Auto complete list function

  getAutoComplList($event) {

    let engineerEmail = this.complaintForm.get('ticket').get('engineer').get('user').get('email').value;

    // this.engineers = [];

    // if (engineerEmail.length > 2) {

    //   if ($event.timeStamp+200> 200) {

    //     this.engineers = this.searchFromArray(this.allEngineers, engineerEmail);

    //   }

    // }

    console.log(engineerEmail+" Selected Engineers");

  }

  searchFromArray(arr:Engineer[], input:string) {

    let matches:Engineer[] = [], i:number;

    for (i = 0; i < arr.length; i++) {

      if (arr[i].user.email.match(input)) {

        matches.push(arr[i]);

      }

    }

    return matches;

  };

  submitHandle(){

    if(this.ticket!=null){

      console.log(this.complaintForm.get('ticket').get('engineer').get('id').value);

      if(this.loggedInUser.role=='manager'){

        let engineer:Engineer=new Engineer();

        engineer.id=this.complaintForm.get('ticket').get('engineer').get('id').value;

        this.ticket.engineer=engineer;

        this.updateTicket(this.ticket);

      }

      else if(this.loggedInUser.role=='engineer'){

        this.ticket.status=this.complaintForm.get('ticket').get('status').value;

        this.updateTicket(this.ticket);

      }

      let comments:string='';

      comments =this.complaintForm.get('notes').get('comments').value;

      console.log(comments+"====>Comments");

      if((comments.trim().length>0)){

        let notes:Notes = new Notes();

        notes.comments=comments;

        notes.isCustomer=(this.loggedInUser.role=='customer');

        notes.updatedBy=this.loggedInUser.email;

        notes.ticket=this.ticket;

        this.ticketService.saveNotes(notes).subscribe({

          next:response=>{

            alert(`Ticket# ${JSON.stringify(response)} updated successfully`)

          },

          error:err=>{

            alert(`There was an error: ${err.message}`);

            console.log(err.message);

          }

        })

      }

    }

    else{

      let ticket:Ticket = this.complaintForm.controls['ticket'].value;

      ticket.customer=this.loggedInCustomer;

      ticket.engineer=null;

      console.log(ticket);

      this.ticketService.raiseNewTicket(ticket).subscribe({

        next:response=>{

          alert( `Your ticket has been submitted! Ticket number: ${JSON.stringify(response)}`);

        },

        error:err=>{

          alert(`There was an error: ${err.message}`);

          console.log(err.message);

        }

      });

    }

  }

  updateTicket(ticket: Ticket) {

    this.ticketService.raiseNewTicket(ticket).subscribe({

      next:response=>{

        alert(`Ticket# ${JSON.stringify(response)} updated successfully`)

      },

      error:err=>{

        alert(`There was an error: ${err.message}`);

        console.log(err.message);

      }

    })

  }

}

1. **Ticket-customer**

* **Ticket-list**

**Ticket-list.component.html**

<div class="container">

    <h2 class="text-center">List of Tickets</h2>

    <table class="table table-danger table-striped">

        <thead>

            <tr>

             <th>Customer Id</th>

             <th>ticket Id</th>

             <th>Customer Name</th>

             <th>Service Type</th>

             <th>Issue type</th>

             <th>Address</th>

             <th>Pincode</th>

             <th>Status</th>

             <th>Assigned to</th>

             <th>Actions</th>

            </tr>

        </thead>

         <tbody>

             <tr \*ngFor="let ticket of tickets">

                 <td>

                     {{ticket.customer.id}}

                 </td>

                 <td>

                     {{ticket.id}}

                 </td>

                 <td>

                     {{ticket.customer.user.fullName}}

                 </td>

                 <td>

                    {{ticket.serviceType}}

                </td>

                 <td>

                     {{ticket.issueType}}

                 </td>

                 <td>

                     {{ticket.address}}

                 </td>

                 <td>

                     {{ticket.zipcode}}

                 </td>

                 <td>

                     {{ticket.status}}

                 </td>

                 <td \*ngIf="ticket.engineer!=null">

                     {{ticket.engineer.user.fullName}}

                 </td>

                 <td class="text-center" \*ngIf="ticket.engineer==null">

                    -

                </td>

                 <td>

                     <a class="btn btn-sm btn-info" [routerLink]="['/viewTicket',ticket.id]">View</a>&nbsp;

                     <a \*ngIf="loggedInUser.role=='manager'" class="btn btn-sm btn-info" [routerLink]="[ '/assign-engineer', routeParam ]">Assign</a>

                     <a \*ngIf="loggedInUser.role=='customer'&&(ticket.status=='RESOLVED')||(ticket.status=='ESCALATED')" class="btn btn-sm btn-info" [routerLink]="[ '/feedback', ticket.id ]">Give Feedback</a>

                 </td>

             </tr>

         </tbody>

     </table>

</div>

**Ticket-list.component.ts**

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

import { Ticket } from 'src/app/common/ticket';

import { User } from 'src/app/common/user';

import { LoginService } from 'src/app/services/login.service';

import { TicketService } from 'src/app/services/ticket.service';

@Component({

  selector: 'app-ticket-list',

  templateUrl: './ticket-list.component.html',

  styleUrls: ['./ticket-list.component.css']

})

export class TicketListComponent implements OnInit {

  tickets:Ticket[]=[];

  selectedTicket:Ticket;

  loggedInUser:User=null;

  constructor(private ticketService:TicketService,private loginService:LoginService) { }

  ngOnInit(): void {

   this.loginService.loggedInUser.subscribe(data=>{

     console.log("line 22");

     this.loggedInUser=data;

     if(data.role=='customer'){

      console.log("line 25");

       this.loginService.loggedInCustomer.subscribe(data2=>this.getTicketsByCustomerId(data2.id));

       console.log("line 27");

     }

     else{

      this.getAllTickets();

      console.log("line 31");

     }

   })

  }

  // End Of onInit

  getTicketsByCustomerId(id:number) {

    this.ticketService.getCustomerTickets(id).subscribe(data=>{

      this.tickets=data;

    })

  }

  getAllTickets() {

    this.ticketService.getAllTickets().subscribe(data=>{

      this.tickets=data;

    });

  }

}

* **User-list**

**User-list.component.html**

<div class="container">

    <!-- List of Customers -->

    <!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

    <div \*ngIf="role=='customer'">

        <h2 class="text-center">List of Customers</h2>

    <table class="table table-success table-striped">

       <thead>

           <tr>

            <th>User Id</th>

            <th>Customer Id</th>

            <th>Full Name</th>

            <th>Email</th>

            <th>Role</th>

            <th>Address</th>

            <th>Services</th>

            <th>Pincode</th>

            <th>Actions</th>

           </tr>

       </thead>

        <tbody>

            <tr \*ngFor="let customer of customers">

                <td>

                    {{customer.user.id}}

                </td>

                <td>

                    {{customer.id}}

                </td>

                <td>

                    {{customer.user.fullName}}

                </td>

                <td>

                    {{customer.user.email}}

                </td>

                <td>

                    {{customer.user.role}}

                </td>

                <td>

                    {{customer.address}}

                </td>

                <td>

                    {{customer.serviceType}}

                </td>

                <td>

                    {{customer.zipcode}}

                </td>

                <td>

                    <a class="btn btn-sm btn-info" [routerLink]="['/editUser',customer.id,customer.user.role]">Edit</a>&nbsp;

                    <button class="btn btn-sm btn-danger" (click)="deleteUser(customer.id,customer.user.role)">Delete</button>

                </td>

            </tr>

        </tbody>

    </table>

    </div>

    <!-- List for Engineers -->

     <!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

     <div \*ngIf="role=='engineer'">

        <h2 class="text-center">List of Engineers</h2>

    <table class="table table-success table-striped">

       <thead>

           <tr>

            <th>User Id</th>

            <th>Engineer Id</th>

            <th>Full Name</th>

            <th>Email</th>

            <th>Role</th>

            <th>Pincode</th>

            <th>Actions</th>

           </tr>

       </thead>

        <tbody>

            <tr \*ngFor="let engineer of engineers">

                <td>

                    {{engineer.user.id}}

                </td>

                <td>

                    {{engineer.id}}

                </td>

                <td>

                    {{engineer.user.fullName}}

                </td>

                <td>

                    {{engineer.user.email}}

                </td>

                <td>

                    {{engineer.user.role}}

                </td>

                <td>

                    {{engineer.zipcode}}

                </td>

                <td>

                    <a class="btn btn-sm btn-info" [routerLink]="['/editUser',engineer.id,engineer.user.role]">Edit</a>&nbsp;

                    <button class="btn btn-sm btn-danger" (click)="deleteUser(engineer.id,engineer.user.role)">Delete</button>

                </td>

            </tr>

        </tbody>

    </table>

    </div>

    <!-- List for Managers -->

     <!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

     <div \*ngIf="role=='manager'">

        <h2 class="text-center">List of Managers</h2>

    <table class="table table-success table-striped">

       <thead>

           <tr>

            <th>User Id</th>

            <th>Manager Id</th>

            <th>Full Name</th>

            <th>Email</th>

            <th>Role</th>

            <th>Pincodes</th>

            <th>Actions</th>

           </tr>

       </thead>

        <tbody>

            <tr \*ngFor="let manager of managers">

                <td>

                    {{manager.user.id}}

                </td>

                <td>

                    {{manager.id}}

                </td>

                <td>

                    {{manager.user.fullName}}

                </td>

                <td>

                    {{manager.user.email}}

                </td>

                <td>

                    {{manager.user.role}}

                </td>

                <td>

                    {{manager.zipcode}}

                </td>

                <td>

                    <a class="btn btn-sm btn-info" [routerLink]="['/editUser',manager.id,manager.user.role]">Edit</a> &nbsp;

                    <button class="btn btn-sm btn-danger" (click)="deleteUser(engineer.id,engineer.user.role)">Delete</button>

                </td>

            </tr>

        </tbody>

    </table>

    </div>

</div>

**User-list.component.ts**

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

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

import { Customer } from 'src/app/common/customer';

import { Engineer } from 'src/app/common/engineer';

import { Manager } from 'src/app/common/manager';

import { UserService } from 'src/app/services/user.service';

@Component({

  selector: 'app-user-list',

  templateUrl: './user-list.component.html',

  styleUrls: ['./user-list.component.css']

})

export class UserListComponent implements OnInit {

  customers:Customer[]=[];

  engineers:Engineer[]=[];

  managers:Manager[]=[];

  role:string='';

  constructor(private userService:UserService, private route:ActivatedRoute) { }

  ngOnInit(): void {

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

      this.loadusers();

    })

  }

  loadusers() {

    if(this.route.snapshot.paramMap.has('role')){

      this.role = this.route.snapshot.paramMap.get('role');

      if(this.role=='customer'){

        this.userService.getCustomers().subscribe(data=>[

          this.customers=data

        ]);

      }

      if(this.role=='engineer'){

        this.userService.getEngineers().subscribe(data=>{

          this.engineers=data;

        });

      }

      if(this.role=='manager'){

        this.userService.getManagers().subscribe(data=>{

          this.managers=data;

        });

      }

    }

  }

  deleteUser(id:number,role:string){

    this.userService.deleteCustomer(id).subscribe(data=>{

      alert(`User ${data} has been deleted successfully`)

      this.loadusers();

    });

  }

}

1. **Services**

**Login.service.ts**

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

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

import { BehaviorSubject, Observable, Subject } from 'rxjs';

import { Customer } from '../common/customer';

import { Engineer } from '../common/engineer';

import { Manager } from '../common/manager';

import { User } from '../common/user';

import { UserDto } from '../common/user-dto';

import { UserService } from './user.service';

@Injectable({

  providedIn: 'root'

})

export class LoginService {

  loggedInUser:Subject<User>=new BehaviorSubject<User>(null);

  loggedInCustomer:Subject<Customer> = new BehaviorSubject<Customer>(null);

  loggedInEngineer:Subject<Engineer> = new BehaviorSubject<Engineer>(null);

  loggedInManager:Subject<Manager> = new BehaviorSubject<Manager>(null);

  constructor(private http:HttpClient,private userService:UserService) { }

  checkLogin(userDto:UserDto):Observable<User>{

    const loginUrl="http://localhost:8081/login/check";

     this.http.post<User>(loginUrl,userDto).subscribe(data=>{

      this.loggedInUser.next(data);

      // If user is a customer then get their details

      if(data.role=='customer'){

        this.userService.getCustomerByUserId(data.id).subscribe(data=>{

          this.loggedInCustomer.next(data);

        })

      }

       // If user is an Engineer then get their details

        if(data.role=='engineer'){

         this.userService.getEngineerByUserId(data.id).subscribe(data=>{

           this.loggedInEngineer.next(data);

         })

       }

        // If user is an Manager then get their details

        if(data.role=='manager'){

          this.userService.getManagerByUserId(data.id).subscribe(data=>{

            this.loggedInManager.next(data);

          })

        }

     })

     return  this.http.post<User>(loginUrl,userDto);

  }

}

**Ticket.service.ts**

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

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

import { Observable } from 'rxjs';

import { Engineer } from '../common/engineer';

import { Feedback } from '../common/feedback';

import { Notes } from '../common/notes';

import { Ticket } from '../common/ticket';

@Injectable({

  providedIn: 'root'

})

export class TicketService {

  private postUrl:string='http://localhost:8081/tickets/new';

  constructor(private httpClient:HttpClient) { }

  raiseNewTicket(ticket:Ticket):Observable<any>{

    return this.httpClient.post<Ticket>(this.postUrl,ticket);

  }

  getCustomerTickets(customerId:number):Observable<Ticket[]>{

    const getUrl="http://localhost:8081/tickets/customer/"+customerId;

    return this.httpClient.get<Ticket[]>(getUrl);

  }

  getAllTickets():Observable<Ticket[]>{

    const getUrl="http://localhost:8081/tickets/";

    return this.httpClient.get<Ticket[]>(getUrl);

  }

  getTicketById(id:number):Observable<Ticket>{

    const getUrl="http://localhost:8081/tickets/"+id;

    return this.httpClient.get<Ticket>(getUrl);

  }

  getEngineersByZipcode(zipcode:string):Observable<Engineer[]>{

    const getUrl='http://localhost:8081/users/engineers/byZipcode/'+zipcode;

    return this.httpClient.get<Engineer[]>(getUrl);

  }

  getNotesByTicketId(id:number):Observable<Notes[]>{

    const getUrl='http://localhost:8081/tickets/notes/'+id;

    return this.httpClient.get<Notes[]>(getUrl);

  }

  saveNotes(notes:Notes):Observable<any>{

    const postUrl='http://localhost:8081/tickets/notes/save';

    return this.httpClient.post<Notes>(postUrl,notes);

  }

  saveFeedback(feedBack:Feedback):Observable<any>{

    const postUrl='http://localhost:8081/tickets/feedback/submit';

    return this.httpClient.post<Feedback>(postUrl,feedBack);

  }

  getFeedback(ticketId:number):Observable<Feedback>{

    const getUrl='http://localhost:8081/tickets/feedback/'+ticketId;

    return this.httpClient.get<Feedback>(getUrl);

  }

}

**User.service.ts**

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

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

import { Observable } from 'rxjs';

import { Customer } from '../common/customer';

import { Engineer } from '../common/engineer';

import { Manager } from '../common/manager';

import { Zipcode } from '../common/zipcode';

import { map } from 'rxjs/operators';

@Injectable({

  providedIn: 'root'

})

export class UserService {

  loggedInCustomer:Customer=null;

  loggedInEngineer:Engineer=null;

  loggedInManager:Manager=null;

  constructor(private http:HttpClient) { }

  createManager(manager: Manager):Observable<any> {

    const postUrl='http://localhost:8081/users/managers/new';

    return this.http.post<Manager>(postUrl,manager);

  }

  createEngineer(engineer: Engineer):Observable<any> {

    const postUrl='http://localhost:8081/users/engineers/new';

    return this.http.post<Engineer>(postUrl,engineer);

  }

  createCustomer(customer:Customer):Observable<any>{

    const postUrl='http://localhost:8081/users/customers/new';

    // customer.ticket=null;

    return this.http.post<Customer>(postUrl,customer);

  }

  getZipcodes():Observable<string[]>{

    const getUrl = "http://localhost:8081/users/zipcodes/";

   console.log( this.http.get(getUrl));

    return this.http.get<string[]>(getUrl).pipe(

      map(response=>response)

      )

  }

  getCustomers():Observable<Customer[]>{

    const getUrl='http://localhost:8081/users/customers';

    return this.http.get<Customer[]>(getUrl);

  }

  getCustomerById(id:number):Observable<Customer>{

    const getUrl='http://localhost:8081/users/customers/'+id;

    return this.http.get<Customer>(getUrl);

  }

  getCustomerByUserId(id:number):Observable<Customer>{

    const getUrl="http://localhost:8081/users/customers/findByUserId/"+id;

    return this.http.get<Customer>(getUrl);

  }

  getEngineerByUserId(id:number):Observable<Engineer>{

    const getUrl="http://localhost:8081/users/engineers/findByUserId/"+id;

    return this.http.get<Engineer>(getUrl);

  }

  getManagerByUserId(id:number):Observable<Manager>{

    const getUrl="http://localhost:8081/users/managers/findByUserId/"+id;

    return this.http.get<Manager>(getUrl);

  }

  getEngineers():Observable<Engineer[]>{

    const getUrl='http://localhost:8081/users/engineers';

    return this.http.get<Engineer[]>(getUrl);

  }

  getEngineerById(id:number):Observable<Engineer>{

    const getUrl='http://localhost:8081/users/engineers/'+id;

    return this.http.get<Engineer>(getUrl);

  }

  getManagers():Observable<Manager[]>{

    const getUrl='http://localhost:8081/users/managers';

    return this.http.get<Manager[]>(getUrl);

  }

  getManagerById(id:number):Observable<Manager>{

    const getUrl='http://localhost:8081/users/managers/'+id;

    return this.http.get<Manager>(getUrl);

  }

  deleteCustomer(id:number):Observable<any>{

    const deleteUrl='http://localhost:8081/users/customers/'+id;

    return this.http.delete(deleteUrl);

  }

}

1. **Common**

**User.ts**

export class User {

    id:number;

    fullName:string;

    role:string;

    email:string;

    phone:string;

    password:string;

}

**Ticket.ts**

import { Customer } from "./customer";

import { Engineer } from "./engineer";

export class Ticket {

    id:number;

    address: string

    customer:Customer;

    engineer:Engineer;

    description: string;

    issueType: string;

    mobile: string;

    serviceType: string;

    zipcode: string;

    createdOn:Date;

    updatedOn:Date;

    status:string;

    // notes:string[];

}

**Manager.ts**

import { User } from "./user";

export class Manager {

    id:number;

    user:User;

    zipcode:string[];

}

**Multi-select.ts**

export class MultiSelect {

    item\_id:number;

    item\_text:string;

    group:string;

}

**Notes.ts**

import { Ticket } from "./ticket";

export class Notes {

    id:number;

    ticket:Ticket;

    comments:string;

    updatedBy:string;

    createdOn:Date;

    isCustomer:boolean;

}

**Engineer.ts**

import { User } from "./user";

export class Engineer {

    id:number;

    user:User;

    zipcode:string;

}

**Customer.ts**

import { User } from "./user";

export class Customer {

    id:number;

    user:User;

    serviceType:string[];

    address:string;

    zipcode:string;

}

**Feedback.ts**

import { Ticket } from "./ticket";

export class Feedback {

    id:number;

    rating:number;

    comments:string;

    ticket:Ticket;

}

**Zipcode.ts**

export class Zipcode {

    zipcode:string;

}

**BackEnd(SpringBoot)**

1. **Entity**

**Customer.java**

package com.abctelecom.entity;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.ElementCollection;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import com.abctelecom.dto.CustomerDto;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

@Entity

@Getter

@Setter

@NoArgsConstructor

@Table

public class Customer {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)

    private User user;

    @ElementCollection(fetch = FetchType.EAGER)

    @JoinTable(name = "customer\_service",joinColumns = @JoinColumn(name="customer\_id"))

//  @Enumerated(EnumType.STRING)

    private Set<String> serviceType;

//  @OneToMany(mappedBy = "customer")

//  private Set<Ticket> tickets;

    @Column

    private String address;

    @Column

    private String zipcode;

    public Customer(CustomerDto customerDto) {

        this.user = customerDto.getUser();

        this.serviceType = customerDto.getServiceType();

        this.address = customerDto.getAddress();

        this.zipcode = customerDto.getZipcode();

    }

}

**Engineer.java**

package com.abctelecom.entity;

import java.time.LocalDate;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import org.hibernate.annotations.CreationTimestamp;

import lombok.Getter;

import lombok.Setter;

@Entity

@Table

@Getter

@Setter

public class Engineer {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)

    private User user;

    @CreationTimestamp

    private LocalDate createdOn;

    @Column

    private String zipcode;

}

**Feedback.java**

package com.abctelecom.entity;

import java.time.LocalDate;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import org.hibernate.annotations.CreationTimestamp;

import lombok.Getter;

import lombok.Setter;

@Entity

@Table

@Getter

@Setter

public class Feedback {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @Column

    private Integer rating;

    @Column

    private String comments;

    @OneToOne

    private Ticket ticket;

    @CreationTimestamp

    private LocalDate createdOn;

}

**Manager.java**

package com.abctelecom.entity;

import java.time.LocalDate;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.ElementCollection;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import org.hibernate.annotations.CreationTimestamp;

import lombok.Getter;

import lombok.Setter;

@Entity

@Table

@Getter

@Setter

public class Manager {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)

    private User user;

    @CreationTimestamp

    private LocalDate createdOn;

    @ElementCollection

    @JoinTable(name = "manager\_pincode",joinColumns = @JoinColumn(name = "manager\_id"))

    private Set<String> zipcode;

}

**Notes.java**

package com.abctelecom.entity;

import java.time.LocalDate;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import org.hibernate.annotations.CreationTimestamp;

import lombok.Getter;

import lombok.Setter;

@Entity

@Table

@Getter

@Setter

public class Notes {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @Column

    private String comments;

    @ManyToOne

    private Ticket ticket;

    @CreationTimestamp

    private LocalDate createdOn;

    @Column

    private boolean isCustomer;

//  updated by USER/ENGINEER's Name

    @Column

    private String updatedBy;

}

**Servicetype.java**

package com.abctelecom.entity;

public enum ServiceType {

    LANDLINE, MOBILE, FIBER\_OPTIC;

}

**Status.java**

package com.abctelecom.entity;

public enum Status {

    RAISED, ASSIGNED, WIP, RESOLVED, ESCALATED;

}

**Ticket.java**

package com.abctelecom.entity;

import java.time.LocalDate;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import javax.persistence.TableGenerator;

import org.hibernate.annotations.CreationTimestamp;

import org.hibernate.annotations.UpdateTimestamp;

import com.fasterxml.jackson.annotation.JsonManagedReference;

import lombok.Getter;

import lombok.Setter;

@Entity

@Getter

@Setter

@Table

public class Ticket {

//   @TableGenerator(name = "Address\_Gen", table = "ID\_GEN", pkColumnName = "GEN\_NAME", valueColumnName = "GEN\_VAL", pkColumnValue = "Addr\_Gen", initialValue = 10000, allocationSize = 100)

    @TableGenerator(name = "Ticket\_Gen" ,initialValue = 100000, allocationSize = 1 )

    @Id

    @GeneratedValue(strategy = GenerationType.TABLE, generator = "Ticket\_Gen")

    private Long id;

//  @Enumerated(EnumType.STRING)

    @Column

    private String serviceType;

    @Column

    private String issueType;

    @CreationTimestamp

    private LocalDate createdOn;

    @UpdateTimestamp

    private LocalDate updatedOn;

    @Column

    private String description;

//  @OneToMany(mappedBy = "ticket")

//  private List<Notes> notes;

    @ManyToOne

    private Customer customer;

    @Column

    private String address;

    @Column

    private String zipcode;

    @Column

    private String mobile;

//  @Enumerated(EnumType.STRING)

    @Column

    private String status;

    @ManyToOne

    private Engineer engineer;

}

**User.java**

package com.abctelecom.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.Getter;

import lombok.Setter;

@Entity

@Table

@Getter

@Setter

public class User {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @Column

    private String fullName;

    @Column

    private String role;

    @Column

    private String email;

    @Column

    private String phone;

    @Column

    private String password;

}

**Zipcode.java**

package com.abctelecom.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.Getter;

import lombok.Setter;

@Entity

@Table(name = "zipcodes")

@Getter

@Setter

public class Zipcode {

    @Id

    private Long id;

    @Column

    private String zipcode;

}

1. **Repository**

**Customerrepository.java**

package com.abctelecom.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import com.abctelecom.entity.Customer;

import com.abctelecom.entity.User;

public interface CustomerRepository extends JpaRepository<Customer, Long> {

    List<Customer> findByUser(User user);

}

**Engeneerrepository.java**

package com.abctelecom.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import com.abctelecom.entity.Engineer;

import com.abctelecom.entity.User;

public interface EngineerRepository extends JpaRepository<Engineer, Long>{

    List<Engineer> findByUser(User user);

    List<Engineer> findByZipcode(String zipcode);

}

**Feedbackrepository.java**

package com.abctelecom.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import com.abctelecom.entity.Feedback;

import com.abctelecom.entity.Ticket;

public interface FeedbackRepository extends JpaRepository<Feedback, Long> {

    List<Feedback> findByTicket(Ticket ticket);

}

**Managerrepository.java**

package com.abctelecom.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import com.abctelecom.entity.Manager;

import com.abctelecom.entity.User;

public interface ManagerRepository extends JpaRepository<Manager, Long>{

    List<Manager> findByUser(User user);

}

**Notesrepository.java**

package com.abctelecom.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.web.bind.annotation.CrossOrigin;

import com.abctelecom.entity.Notes;

import com.abctelecom.entity.Ticket;

@CrossOrigin

public interface NotesRepository extends JpaRepository<Notes, Long> {

    List<Notes> findByTicket(Ticket ticket);

}

**Ticketrepository.java**

package com.abctelecom.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.web.bind.annotation.CrossOrigin;

import com.abctelecom.entity.Customer;

import com.abctelecom.entity.Engineer;

import com.abctelecom.entity.Ticket;

@CrossOrigin

public interface TicketRepository extends JpaRepository<Ticket, Long>{

    List<Ticket> findByCustomer(Customer customer);

    List<Ticket> findByZipcode(String zipcode);

    List<Ticket> findByEngineer(Engineer engineer);

}

**Userrepository.java**

package com.abctelecom.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import com.abctelecom.entity.User;

public interface UserRepository extends JpaRepository<User, Long>{

    List<User> findByEmail(String email);

}

**Zipcoderepository.java**

package com.abctelecom.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.web.bind.annotation.CrossOrigin;

import com.abctelecom.entity.Zipcode;

@CrossOrigin

public interface ZipcodeRepository extends JpaRepository<Zipcode, Long> {

}

1. **Controller**

**Logincontroller.java**

package com.abctelecom.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.abctelecom.dto.UserDto;

import com.abctelecom.entity.User;

import com.abctelecom.service.UserService;

@RestController

@CrossOrigin

@RequestMapping("/login")

public class LoginController {

    @Autowired

    private UserService userService;

    @PostMapping("/check")

    public User loginHandler(@RequestBody UserDto userDto) {

        System.out.println(userDto.getEmail());

        System.out.println(userService.loginCheck(userDto).getEmail());

        return userService.loginCheck(userDto);

    }

}

**Ticketcontroller.java**

package com.abctelecom.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.abctelecom.entity.Feedback;

import com.abctelecom.entity.Notes;

import com.abctelecom.entity.Ticket;

import com.abctelecom.service.TicketService;

@RestController

@CrossOrigin

@RequestMapping("/tickets")

public class TicketController {

    private TicketService ticketService;

    @Autowired

    public TicketController(TicketService ticketService) {

        this.ticketService = ticketService;

    }

    @PostMapping("/new")

    public String raiseNew(@RequestBody Ticket ticket) {

        return ticketService.save(ticket);

    }

    @GetMapping("/")

    public List<Ticket> getAllTickets(){

        return ticketService.getAllTickets();

    }

    @GetMapping("/customer/{id}")

    public List<Ticket> getTicketsByCustomerId(@PathVariable String id){

        return ticketService.getTicketsByCustomerId(Long.parseLong(id));

    }

    @GetMapping("/{id}")

    public Ticket getTicketById(@PathVariable String id) {

        return ticketService.getTicketById(Long.parseLong(id));

    }

    @GetMapping("/notes/{ticketId}")

    public List<Notes> getNotesByTicketId(@PathVariable String ticketId) {

        return ticketService.getNotesByTicketId(Long.parseLong(ticketId));

    }

    @PostMapping("/notes/save")

    public String saveNotes(@RequestBody Notes notes) {

        return ticketService.saveNotes(notes);

    }

    @GetMapping("/feedback/{ticketId}")

    public Feedback getFeedbackByTicket(@PathVariable String ticketId ) {

        return ticketService.findFeedbackByTicket(Long.parseLong(ticketId));

    }

    @PostMapping("/feedback/submit")

    public String submitFeedback(@RequestBody Feedback feedback) {

        return ticketService.saveFeedback(feedback);

    }

}

**Usercontroller.java**

package com.abctelecom.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.abctelecom.dto.CustomerDto;

import com.abctelecom.entity.Customer;

import com.abctelecom.entity.Engineer;

import com.abctelecom.entity.Manager;

import com.abctelecom.service.UserService;

@RestController

@CrossOrigin

@RequestMapping("/users")

public class UserController {

    @Autowired

    private UserService userService;

    @GetMapping("/")

    public ResponseEntity<String> apiTest(){

        return new ResponseEntity<String>("Hi user", HttpStatus.OK);

    }

    @GetMapping("/zipcodes")

    public List<String> getZipcodes(){

        System.out.println(userService.getZipcodes());

        return userService.getZipcodes();

    }

    @PostMapping("/customers/new")

    public String createCustomer(@RequestBody CustomerDto customerDto) {

        Customer customer = new Customer(customerDto);

        return userService.addCustomer(customer);

    }

    @PostMapping("/engineers/new")

    public String createEngineer(@RequestBody Engineer engineer) {

        return userService.addEngineer(engineer);

    }

    @PostMapping("/managers/new")

    public String createManager(@RequestBody Manager manager) {

        return userService.addManager(manager);

    }

    @GetMapping("/customers")

    public List<Customer> getAllCustomers(){

        return userService.getAllCustomers();

    }

    @GetMapping("customers/{id}")

    public Customer getCustomerById(@PathVariable String id) {

        return userService.getCustomerById(Long.parseLong(id));

    }

    @GetMapping("customers/findByUserId/{id}")

    public Customer findCustomerByUserId(@PathVariable String id) {

        return userService.findCustomerByUserId(Long.parseLong(id));

    }

    @GetMapping("engineers/findByUserId/{id}")

    public Engineer findEngineerByUserId(@PathVariable String id) {

        return userService.findEngineerByUserId(Long.parseLong(id));

    }

    @GetMapping("managers/findByUserId/{id}")

    public Manager findBYUserId(@PathVariable String id) {

        return userService.findManagerByUserId(Long.parseLong(id));

    }

    @GetMapping("engineers/{id}")

    public Engineer getEngineerById(@PathVariable String id) {

        return userService.getEngineerById(Long.parseLong(id));

    }

    @GetMapping("managers/{id}")

    public Manager getManagerById(@PathVariable String id) {

        return userService.getManagerById(Long.parseLong(id));

    }

    @GetMapping("/engineers")

    public List<Engineer> getAllEngineers(){

        return userService.getAllEngineers();

    }

    @GetMapping("engineers/byZipcode/{zipcode}")

    public List<Engineer> getEngineersByZipcode(@PathVariable String zipcode){

        return userService.getEngineersByZipcode(zipcode);

    }

    @GetMapping("/managers")

    public List<Manager> getAllManagers(){

        return userService.getAllManagers();

    }

    @DeleteMapping("/customers/{id}")

    public String deleteCustomer(@PathVariable String id) {

        return userService.deleteCustomer(Long.parseLong(id));

    }

}

1. **Dto**

**Customerdto.java**

package com.abctelecom.dto;

import java.util.Set;

import com.abctelecom.entity.User;

import lombok.Getter;

import lombok.Setter;

@Getter

@Setter

public class CustomerDto {

    private Long id;

    private User user;

    private Set<String> serviceType;

    private String address;

    private String zipcode;

}

**Userdto.java**

package com.abctelecom.dto;

import lombok.Getter;

import lombok.Setter;

@Getter

@Setter

public class UserDto {

    private String email;

    private String password;

}

**AbcTelecomeApplication.java**

package com.abctelecom;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AbcTelecomApplication {

    public static void main(String[] args) {

        SpringApplication.run(AbcTelecomApplication.class, args);

    }

}

1. **Resources**

**Application.properties**

server.port=8081

spring.datasource.url=jdbc:mysql://localhost:3306/ABCTelecom

spring.datasource.username=root

spring.datasource.password=Nilesh@1991

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.generate-ddl=true

spring.jpa.properties.hibernate.format\_sql=true

logging.level.org.hibernate.type=trace

spring.data.rest.base-path=/api

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>2.6.2</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>com.abctelecom</groupId>

    <artifactId>abcTelecom</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>abcTelecom</name>

    <description>Project for complaint redressal system</description>

    <properties>

        <java.version>1.8</java.version>

    </properties>

    <dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-jpa</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-rest</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

            <groupId>mysql</groupId>

            <artifactId>mysql-connector-java</artifactId>

            <scope>runtime</scope>

        </dependency>

        <dependency>

            <groupId>org.projectlombok</groupId>

            <artifactId>lombok</artifactId>

            <optional>true</optional>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-devtools</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>