**FrontEnd(Angular)**

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

**Home.component.html**

<divclass="container"style="margin: 5px 5px;">

    <h3style="margin:auto;width: 300px;">Create New User</h3>

    <divclass="row">

        <divclass="col md-3">

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

        </div>

        <divclass="col md-3">

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

        </div>

        <divclass="col md-3">

            <buttonclass="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']

})

exportclassHomeComponentimplementsOnInit {

  constructor() { }

  ngOnInit():void {

  }

}

1. **Login**

**Login.component.html**

<sectionclass="vh-50 gradient-custom">

    <divclass="container  py-5 h-100 ">

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

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

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

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

                        <divclass="mb-md-5 mt-md-4 pb-5">

                            <h3class="fw-bold mb-2 text-uppercase">ABC Telecom Ltd.</h3>

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

                            <div></div>

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

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

                                <divclass="form-outline form-white mb-3">

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

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

                                </div>

                                <divclass="form-outline form-white mb-4">

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

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

                                </div>

                                <buttonclass="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']

})

exportclassLoginComponentimplementsOnInit {

  loginForm:FormGroup;

  constructor(

    privateformBulder:FormBuilder,

    privateloginService:LoginService,

     privateuserService:UserService,

     privaterouter:Router

  ) { }

  ngOnInit():void {

    this.loginForm=this.formBulder.group({

      email:newFormControl('',[Validators.required,Validators.email]),

        password:newFormControl('',Validators.required)

    });

  }

  loginhandler(){

    letuserDto: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");

            })

          }

         elseif(response.role=='engineer'){

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

              this.userService.loggedInEngineer=data;

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

            })

          }

          elseif(response.role=='manager'){

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

              this.userService.loggedInManager=data;

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

            })

          }

          elseif(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">

        <divclass="container-fluid">

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

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

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

            </button>

            <divclass="collapse navbar-collapse"id="navbarCollapse1">

                <divclass="navbar-nav">

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

                    <liclass="nav-item dropdown">

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

                      <ulclass="dropdown-menu">

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

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

                          <li><hrclass="dropdown-divider"></li>

                           <li>

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

                          </li>

                        </div>

                      </ul>

                    </li>

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

                      <li  class="nav-item dropdown">

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

                        <ulclass="dropdown-menu">

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

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

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

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

                          <li><hrclass="dropdown-divider"></li>

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

                        </div>

                        </ul>

                      </li>

                    </div>

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

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

                     <divclass="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>

            <divclass="profile-icon">

              <h4class="profile-name">{{loggedInUser.fullName}}</h4>

              <svgwidth="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><circlecx="16"cy="16"fill="none"  r="15"stroke="#c9c8c7"stroke-linejoin="round"stroke-miterlimit="10"stroke-width="2"/><pathd="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"/><circlecx="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']

})

exportclassCoreComponentimplementsOnInit {

  loggedInUser:User=null;

  constructor(privateuserService:UserService, privateloginService:LoginService,privaterouter: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**

<divclass="conatiner">

    <h1>Assign Engineer</h1>

    <labelfor="">

        <!-- {{ticket.id}} -->

    </label>

    <selectid=""></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']

})

exportclassAssignEngineerComponentimplementsOnInit {

  ticket:Ticket;

  constructor(privateuserService:UserService,privaterouter:Router) { }

  ngOnInit():void {

    this.router

  }

}

1. **Create-user**

**Create-user.component.html**

<divclass="container">

    <divclass="row">

        <divclass="col-md-8 offset-md-2">

            <h1>{{formHeading}}</h1>

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

                <divformGroupName="user">

                    <inputtype="hidden"formControlName='id'>

                    <divclass="form-group">

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

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

                    <divclass="form-group">

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

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

                    <divclass="form-group">

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

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

                    <divclass="form-group">

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

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

                    <divclass="form-group">

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

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

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

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

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

                        </select>

                    </div>

                </div>

                <!-- Form for customer -->

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

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

                    <divformGroupName="customer">

                        <inputtype="hidden"formControlName='id'>

                        <divclass="form-group">

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

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

                        <divclass="form-group">

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

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

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

                            </select>

                        </div>

                        <divclass="form-group">

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

                            <ng-multiselect-dropdownformControlName="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'">

                    <divformGroupName="engineer">

                        <inputtype="hidden"formControlName='id'>

                        <divclass="form-group">

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

                            <selectformControlName='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'">

                    <divformGroupName="manager">

                        <inputtype="hidden"formControlName='id'>

                        <divclass="form-group">

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

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

                        </div>

                    </div>

                </div>

                <divclass="form-group">

                    <divclass="col-md-4 offset-md-2">

                        <divclass="row">

                            <divclass="col">

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

                            </div>

                            <divclass="col">

                                <buttontype="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']

})

exportclassCreateUserComponentimplementsOnInit {

  userForm:FormGroup;

  formHeading:string='New user';

  role:string;

  zipcodes:string[]=[];

  constructor(privateformBuilder:FormBuilder, privateuserService:UserService,privateroute: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:newFormControl("",[Validators.required]),

        role:newFormControl("",[Validators.required]),

        email:newFormControl("",[Validators.required,Validators.email]),

        phone:newFormControl("",[Validators.required]),

        password:newFormControl("",[Validators.required])

      }),

      customer:this.formBuilder.group({

        id:[''],

        serviceType:newFormControl("",[Validators.required]),

        address:newFormControl("",[Validators.required]),

        zipcode:newFormControl("",[Validators.required]),

      }),

      engineer:this.formBuilder.group({

        id:[''],

        zipcode:newFormControl("",[Validators.required])

      }),

      manager:this.formBuilder.group({

        id:[''],

        zipcode:newFormControl("",[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');

      leteditUserForm=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) {

    leteditManagerForm=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) {

    leteditEngineerForm=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){

    leteditCustomerForm=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>{

    letmultiSelect:MultiSelect[]=[];

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

      this.zipcodes=data;

      console.log(this.zipcodes);

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

        lettheMultiSelect=newMultiSelect();

        theMultiSelect.item\_id=i;

        theMultiSelect.item\_text=data[i];

        theMultiSelect.group='z';

        multiSelect.push(theMultiSelect);

      }

    })

    returnmultiSelect;

  }

  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);

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

  //  For Customer Role

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

   if(this.role=='customer'){

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

    letserviceToString:string[]=[];

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

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

    letcustomer:Customer=newCustomer();

    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'){

      letengineer:Engineer=newEngineer();

      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"){

    letmanager:Manager=newManager();

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

    letpincodesToString: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**

<divclass="container">

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

        <divclass="row justify-content-around">

            <divclass="col-sm-3">

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

                <divclass="form-group">

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

                      Rating

                    </label>

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

                      <optionvalue="1">1</option>

                      <optionvalue="2">2</option>

                      <optionvalue="3">3</option>

                      <optionvalue="4">4</option>

                      <optionvalue="5">5</option>

                    </select>

                  </div>

            </div>

        </div>

        <divclass="row justify-content-around">

            <divclass="col-sm-3">

                <divclass="form-group">

                    <labelclass="control-lable"for="comments">

                      Comments

                    </label>

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

                  </div>

            </div>

        </div>

        <br>

        <divclass="row justify-content-around">

            <divclass="col-sm-3">

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

            </div>

        </div>

        <br>

        <divclass="row justify-content-around">

            <divclass="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']

})

exportclassFeedbackComponentimplementsOnInit {

  ticket:Ticket;

  feedbackForm:FormGroup;

  constructor(privateticketService:TicketService,privaterouter:ActivatedRoute, privateformBuilder: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(){

    letfeedback:Feedback=newFeedback();

    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**

<divclass="container">

    <divclass="row">

      <divclass="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>

          <divclass="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">

          <divformGroupName="ticket">

            <inputtype="hidden"formControlName='id'>

          <divformGroupName="customer">

            <inputtype="hidden"formControlName='id'>

            <divclass="form-group"formGroupName="user">

            <labelclass="control-lable"for="fullName">

                Full Name\*

            </label>

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

          </div>

        </div>

        <divclass="form-group">

          <labelclass="control-lable"for="mobile">

              Mobile\*

          </label>

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

        </div>

        <divclass="form-group">

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

            Service Type\*

          </label>

          <selectid="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>

        <divclass="form-group">

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

            Issue Type\*

          </label>

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

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

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

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

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

          </select>

        </div>

        <divclass="form-group">

          <labelclass="control-lable"for="description">

             Detailed Description\*

          </label>

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

        </div>

        <divclass="form-group">

          <labelclass="control-lable"for="address">

              Address\*

          </label>

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

        </div>

        <divclass="form-group">

          <labelclass="control-lable"for="zipcode">

            Zipcode\*

          </label>

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

        </div>

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

          <divclass="form-group">

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

            <selectformControlName="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">

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

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

            <optionvalue="RAISED">RAISED</option>

            <optionvalue="WIP">WIP</option>

            <optionvalue="RESOLVED">RESOLVED</option>

            <optionvalue="ESCALATED">ESCALATED</option>

          </select>

        </div>

  </div>

 <!-- Displaying notes -->

  <divclass="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">

      <labelclass="control-lable"for="notes">

         Add Comments

      </label>

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

    </div>

   </div>

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

    <divclass="col-md-4 offset-md-2">

    <divclass="row">

      <divclass="col">

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

      </div>

      <divclass="col">

        <buttontype="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']

})

exportclassTicketComponentimplementsOnInit {

  complaintForm:FormGroup;

  loggedInUser:User=null;

  loggedInCustomer:Customer=newCustomer();

  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(

    privateformBuilder:FormBuilder,

    privateticketService:TicketService,

    privateloginService:LoginService,

    privateroute: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:newFormControl("",[Validators.required]),

          address:newFormControl("",[Validators.required]),

          zipcode:newFormControl("",[Validators.required]),

          user:this.formBuilder.group({

              id:[''],

              fullName:newFormControl("",[Validators.required]),

              role:newFormControl("",[Validators.required]),

              email:newFormControl("",[Validators.required,Validators.email]),

              phone:newFormControl("",[Validators.required]),

              password:newFormControl("",[Validators.required])

          })

        }),

        // engineer:['']

        engineer:this.formBuilder.group({

            id:[''],

            zipcode:newFormControl("",[Validators.required]),

            user:this.formBuilder.group({

                id:[''],

                fullName:newFormControl("",[Validators.required]),

                role:newFormControl("",[Validators.required]),

                email:newFormControl("",[Validators.required,Validators.email]),

                phone:newFormControl("",[Validators.required]),

                password:newFormControl("",[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();

        })

      }

      elseif(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});

          }

        });

      }

      elseif(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;

        })

      }

      elseif(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;

        letticketFrom=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) {

    letengineerEmail=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) {

    letmatches:Engineer[] = [], i:number;

    for (i=0; i<arr.length; i++) {

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

        matches.push(arr[i]);

      }

    }

    returnmatches;

  };

  submitHandle(){

    if(this.ticket!=null){

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

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

        letengineer:Engineer=newEngineer();

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

        this.ticket.engineer=engineer;

        this.updateTicket(this.ticket);

      }

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

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

        this.updateTicket(this.ticket);

      }

      letcomments:string='';

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

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

      if((comments.trim().length>0)){

        letnotes:Notes=newNotes();

        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{

      letticket: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**

<divclass="container">

    <h2class="text-center">List of Tickets</h2>

    <tableclass="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>

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

                    -

                </td>

                 <td>

                     <aclass="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']

})

exportclassTicketListComponentimplementsOnInit {

  tickets:Ticket[]=[];

  selectedTicket:Ticket;

  loggedInUser:User=null;

  constructor(privateticketService:TicketService,privateloginService: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**

<divclass="container">

    <!-- List of Customers -->

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

    <div\*ngIf="role=='customer'">

        <h2class="text-center">List of Customers</h2>

    <tableclass="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>

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

                    <buttonclass="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'">

        <h2class="text-center">List of Engineers</h2>

    <tableclass="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>

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

                    <buttonclass="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'">

        <h2class="text-center">List of Managers</h2>

    <tableclass="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>

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

                    <buttonclass="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']

})

exportclassUserListComponentimplementsOnInit {

  customers:Customer[]=[];

  engineers:Engineer[]=[];

  managers:Manager[]=[];

  role:string='';

  constructor(privateuserService:UserService, privateroute: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'

})

exportclassLoginService {

  loggedInUser:Subject<User>=newBehaviorSubject<User>(null);

  loggedInCustomer:Subject<Customer>=newBehaviorSubject<Customer>(null);

  loggedInEngineer:Subject<Engineer>=newBehaviorSubject<Engineer>(null);

  loggedInManager:Subject<Manager>=newBehaviorSubject<Manager>(null);

  constructor(privatehttp:HttpClient,privateuserService:UserService) { }

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

    constloginUrl="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'

})

exportclassTicketService {

  privatepostUrl:string='http://localhost:8081/tickets/new';

  constructor(privatehttpClient:HttpClient) { }

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

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

  }

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

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

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

  }

  getAllTickets():Observable<Ticket[]>{

    constgetUrl="http://localhost:8081/tickets/";

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

  }

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

    constgetUrl="http://localhost:8081/tickets/"+id;

    returnthis.httpClient.get<Ticket>(getUrl);

  }

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

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

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

  }

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

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

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

  }

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

    constpostUrl='http://localhost:8081/tickets/notes/save';

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

  }

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

    constpostUrl='http://localhost:8081/tickets/feedback/submit';

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

  }

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

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

    returnthis.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'

})

exportclassUserService {

  loggedInCustomer:Customer=null;

  loggedInEngineer:Engineer=null;

  loggedInManager:Manager=null;

  constructor(privatehttp:HttpClient) { }

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

    constpostUrl='http://localhost:8081/users/managers/new';

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

  }

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

    constpostUrl='http://localhost:8081/users/engineers/new';

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

  }

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

    constpostUrl='http://localhost:8081/users/customers/new';

    // customer.ticket=null;

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

  }

  getZipcodes():Observable<string[]>{

    constgetUrl="http://localhost:8081/users/zipcodes/";

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

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

      map(response=>response)

      )

  }

  getCustomers():Observable<Customer[]>{

    constgetUrl='http://localhost:8081/users/customers';

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

  }

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

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

    returnthis.http.get<Customer>(getUrl);

  }

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

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

    returnthis.http.get<Customer>(getUrl);

  }

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

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

    returnthis.http.get<Engineer>(getUrl);

  }

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

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

    returnthis.http.get<Manager>(getUrl);

  }

  getEngineers():Observable<Engineer[]>{

    constgetUrl='http://localhost:8081/users/engineers';

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

  }

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

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

    returnthis.http.get<Engineer>(getUrl);

  }

  getManagers():Observable<Manager[]>{

    constgetUrl='http://localhost:8081/users/managers';

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

  }

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

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

    returnthis.http.get<Manager>(getUrl);

  }

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

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

    returnthis.http.delete(deleteUrl);

  }

}

1. **Common**

**User.ts**

exportclassUser {

    id:number;

    fullName:string;

    role:string;

    email:string;

    phone:string;

    password:string;

}

**Ticket.ts**

import { Customer } from"./customer";

import { Engineer } from"./engineer";

exportclassTicket {

    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";

exportclassManager {

    id:number;

    user:User;

    zipcode:string[];

}

**Multi-select.ts**

exportclassMultiSelect {

    item\_id:number;

    item\_text:string;

    group:string;

}

**Notes.ts**

import { Ticket } from"./ticket";

exportclassNotes {

    id:number;

    ticket:Ticket;

    comments:string;

    updatedBy:string;

    createdOn:Date;

    isCustomer:boolean;

}

**Engineer.ts**

import { User } from"./user";

exportclassEngineer {

    id:number;

    user:User;

    zipcode:string;

}

**Customer.ts**

import { User } from"./user";

exportclassCustomer {

    id:number;

    user:User;

    serviceType:string[];

    address:string;

    zipcode:string;

}

**Feedback.ts**

import { Ticket } from"./ticket";

exportclassFeedback {

    id:number;

    rating:number;

    comments:string;

    ticket:Ticket;

}

**Zipcode.ts**

exportclassZipcode {

    zipcode:string;

}

**BackEnd(SpringBoot)**

1. **Entity**

**Customer.java**

packagecom.abctelecom.entity;

importjava.util.Set;

importjavax.persistence.CascadeType;

importjavax.persistence.Column;

importjavax.persistence.ElementCollection;

importjavax.persistence.Entity;

importjavax.persistence.FetchType;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.JoinColumn;

importjavax.persistence.JoinTable;

importjavax.persistence.OneToOne;

importjavax.persistence.Table;

importcom.abctelecom.dto.CustomerDto;

importlombok.Getter;

importlombok.NoArgsConstructor;

importlombok.Setter;

@Entity

@Getter

@Setter

@NoArgsConstructor

@Table

publicclassCustomer {

    @Id

    @GeneratedValue(strategy =GenerationType.IDENTITY)

    privateLongid;

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

    privateUseruser;

    @ElementCollection(fetch =FetchType.EAGER)

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

//  @Enumerated(EnumType.STRING)

    privateSet<String>serviceType;

//  @OneToMany(mappedBy = "customer")

//  private Set<Ticket> tickets;

    @Column

    privateStringaddress;

    @Column

    privateStringzipcode;

    publicCustomer(CustomerDtocustomerDto) {

        this.user=customerDto.getUser();

        this.serviceType=customerDto.getServiceType();

        this.address=customerDto.getAddress();

        this.zipcode=customerDto.getZipcode();

    }

}

**Engineer.java**

packagecom.abctelecom.entity;

importjava.time.LocalDate;

importjavax.persistence.CascadeType;

importjavax.persistence.Column;

importjavax.persistence.Entity;

importjavax.persistence.FetchType;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.OneToOne;

importjavax.persistence.Table;

importorg.hibernate.annotations.CreationTimestamp;

importlombok.Getter;

importlombok.Setter;

@Entity

@Table

@Getter

@Setter

publicclassEngineer {

    @Id

    @GeneratedValue(strategy =GenerationType.IDENTITY)

    privateLongid;

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

    privateUseruser;

    @CreationTimestamp

    privateLocalDatecreatedOn;

    @Column

    privateStringzipcode;

}

**Feedback.java**

packagecom.abctelecom.entity;

importjava.time.LocalDate;

importjavax.persistence.Column;

importjavax.persistence.Entity;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.OneToOne;

importjavax.persistence.Table;

importorg.hibernate.annotations.CreationTimestamp;

importlombok.Getter;

importlombok.Setter;

@Entity

@Table

@Getter

@Setter

publicclassFeedback {

    @Id

    @GeneratedValue(strategy =GenerationType.IDENTITY)

    privateLongid;

    @Column

    privateIntegerrating;

    @Column

    privateStringcomments;

    @OneToOne

    privateTicketticket;

    @CreationTimestamp

    privateLocalDatecreatedOn;

}

**Manager.java**

packagecom.abctelecom.entity;

importjava.time.LocalDate;

importjava.util.Set;

importjavax.persistence.CascadeType;

importjavax.persistence.ElementCollection;

importjavax.persistence.Entity;

importjavax.persistence.FetchType;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.JoinColumn;

importjavax.persistence.JoinTable;

importjavax.persistence.OneToOne;

importjavax.persistence.Table;

importorg.hibernate.annotations.CreationTimestamp;

importlombok.Getter;

importlombok.Setter;

@Entity

@Table

@Getter

@Setter

publicclassManager {

    @Id

    @GeneratedValue(strategy =GenerationType.IDENTITY)

    privateLongid;

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

    privateUseruser;

    @CreationTimestamp

    privateLocalDatecreatedOn;

    @ElementCollection

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

    privateSet<String>zipcode;

}

**Notes.java**

packagecom.abctelecom.entity;

importjava.time.LocalDate;

importjavax.persistence.Column;

importjavax.persistence.Entity;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.ManyToOne;

importjavax.persistence.Table;

importorg.hibernate.annotations.CreationTimestamp;

importlombok.Getter;

importlombok.Setter;

@Entity

@Table

@Getter

@Setter

publicclassNotes {

    @Id

    @GeneratedValue(strategy =GenerationType.IDENTITY)

    privateLongid;

    @Column

    privateStringcomments;

    @ManyToOne

    privateTicketticket;

    @CreationTimestamp

    privateLocalDatecreatedOn;

    @Column

    privatebooleanisCustomer;

//  updated by USER/ENGINEER's Name

    @Column

    privateStringupdatedBy;

}

**Servicetype.java**

packagecom.abctelecom.entity;

publicenumServiceType {

    LANDLINE, MOBILE, FIBER\_OPTIC;

}

**Status.java**

packagecom.abctelecom.entity;

publicenumStatus {

    RAISED, ASSIGNED, WIP, RESOLVED, ESCALATED;

}

**Ticket.java**

packagecom.abctelecom.entity;

importjava.time.LocalDate;

importjavax.persistence.Column;

importjavax.persistence.Entity;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.ManyToOne;

importjavax.persistence.Table;

importjavax.persistence.TableGenerator;

importorg.hibernate.annotations.CreationTimestamp;

importorg.hibernate.annotations.UpdateTimestamp;

importcom.fasterxml.jackson.annotation.JsonManagedReference;

importlombok.Getter;

importlombok.Setter;

@Entity

@Getter

@Setter

@Table

publicclassTicket {

//   @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")

    privateLongid;

//  @Enumerated(EnumType.STRING)

    @Column

    privateStringserviceType;

    @Column

    privateStringissueType;

    @CreationTimestamp

    privateLocalDatecreatedOn;

    @UpdateTimestamp

    privateLocalDateupdatedOn;

    @Column

    privateStringdescription;

//  @OneToMany(mappedBy = "ticket")

//  private List<Notes> notes;

    @ManyToOne

    privateCustomercustomer;

    @Column

    privateStringaddress;

    @Column

    privateStringzipcode;

    @Column

    privateStringmobile;

//  @Enumerated(EnumType.STRING)

    @Column

    privateStringstatus;

    @ManyToOne

    privateEngineerengineer;

}

**User.java**

packagecom.abctelecom.entity;

importjavax.persistence.Column;

importjavax.persistence.Entity;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.Table;

importlombok.Getter;

importlombok.Setter;

@Entity

@Table

@Getter

@Setter

publicclassUser {

    @Id

    @GeneratedValue(strategy =GenerationType.IDENTITY)

    privateLongid;

    @Column

    privateStringfullName;

    @Column

    privateStringrole;

    @Column

    privateStringemail;

    @Column

    privateStringphone;

    @Column

    privateStringpassword;

}

**Zipcode.java**

packagecom.abctelecom.entity;

importjavax.persistence.Column;

importjavax.persistence.Entity;

importjavax.persistence.Id;

importjavax.persistence.Table;

importlombok.Getter;

importlombok.Setter;

@Entity

@Table(name ="zipcodes")

@Getter

@Setter

publicclassZipcode {

    @Id

    privateLongid;

    @Column

    privateStringzipcode;

}

1. **Repository**

**Customerrepository.java**

packagecom.abctelecom.repository;

importjava.util.List;

importorg.springframework.data.jpa.repository.JpaRepository;

importcom.abctelecom.entity.Customer;

importcom.abctelecom.entity.User;

publicinterfaceCustomerRepositoryextendsJpaRepository<Customer, Long> {

    List<Customer>findByUser(Useruser);

}

**Engeneerrepository.java**

packagecom.abctelecom.repository;

importjava.util.List;

importorg.springframework.data.jpa.repository.JpaRepository;

importcom.abctelecom.entity.Engineer;

importcom.abctelecom.entity.User;

publicinterfaceEngineerRepositoryextendsJpaRepository<Engineer, Long>{

    List<Engineer>findByUser(Useruser);

    List<Engineer>findByZipcode(Stringzipcode);

}

**Feedbackrepository.java**

packagecom.abctelecom.repository;

importjava.util.List;

importorg.springframework.data.jpa.repository.JpaRepository;

importcom.abctelecom.entity.Feedback;

importcom.abctelecom.entity.Ticket;

publicinterfaceFeedbackRepositoryextendsJpaRepository<Feedback, Long> {

    List<Feedback>findByTicket(Ticketticket);

}

**Managerrepository.java**

packagecom.abctelecom.repository;

importjava.util.List;

importorg.springframework.data.jpa.repository.JpaRepository;

importcom.abctelecom.entity.Manager;

importcom.abctelecom.entity.User;

publicinterfaceManagerRepositoryextendsJpaRepository<Manager, Long>{

    List<Manager>findByUser(Useruser);

}

**Notesrepository.java**

packagecom.abctelecom.repository;

importjava.util.List;

importorg.springframework.data.jpa.repository.JpaRepository;

importorg.springframework.web.bind.annotation.CrossOrigin;

importcom.abctelecom.entity.Notes;

importcom.abctelecom.entity.Ticket;

@CrossOrigin

publicinterfaceNotesRepositoryextendsJpaRepository<Notes, Long> {

    List<Notes>findByTicket(Ticketticket);

}

**Ticketrepository.java**

packagecom.abctelecom.repository;

importjava.util.List;

importorg.springframework.data.jpa.repository.JpaRepository;

importorg.springframework.web.bind.annotation.CrossOrigin;

importcom.abctelecom.entity.Customer;

importcom.abctelecom.entity.Engineer;

importcom.abctelecom.entity.Ticket;

@CrossOrigin

publicinterfaceTicketRepositoryextendsJpaRepository<Ticket, Long>{

    List<Ticket>findByCustomer(Customercustomer);

    List<Ticket>findByZipcode(Stringzipcode);

    List<Ticket>findByEngineer(Engineerengineer);

}

**Userrepository.java**

packagecom.abctelecom.repository;

importjava.util.List;

importorg.springframework.data.jpa.repository.JpaRepository;

importcom.abctelecom.entity.User;

publicinterfaceUserRepositoryextendsJpaRepository<User, Long>{

    List<User>findByEmail(Stringemail);

}

**Zipcoderepository.java**

packagecom.abctelecom.repository;

importorg.springframework.data.jpa.repository.JpaRepository;

importorg.springframework.web.bind.annotation.CrossOrigin;

importcom.abctelecom.entity.Zipcode;

@CrossOrigin

publicinterfaceZipcodeRepositoryextendsJpaRepository<Zipcode, Long> {

}

1. **Controller**

**Logincontroller.java**

packagecom.abctelecom.controller;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.web.bind.annotation.CrossOrigin;

importorg.springframework.web.bind.annotation.PostMapping;

importorg.springframework.web.bind.annotation.RequestBody;

importorg.springframework.web.bind.annotation.RequestMapping;

importorg.springframework.web.bind.annotation.RestController;

importcom.abctelecom.dto.UserDto;

importcom.abctelecom.entity.User;

importcom.abctelecom.service.UserService;

@RestController

@CrossOrigin

@RequestMapping("/login")

publicclassLoginController {

    @Autowired

    privateUserServiceuserService;

    @PostMapping("/check")

    publicUserloginHandler(@RequestBodyUserDtouserDto) {

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

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

        returnuserService.loginCheck(userDto);

    }

}

**Ticketcontroller.java**

packagecom.abctelecom.controller;

importjava.util.List;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.web.bind.annotation.CrossOrigin;

importorg.springframework.web.bind.annotation.GetMapping;

importorg.springframework.web.bind.annotation.PathVariable;

importorg.springframework.web.bind.annotation.PostMapping;

importorg.springframework.web.bind.annotation.RequestBody;

importorg.springframework.web.bind.annotation.RequestMapping;

importorg.springframework.web.bind.annotation.RestController;

importcom.abctelecom.entity.Feedback;

importcom.abctelecom.entity.Notes;

importcom.abctelecom.entity.Ticket;

importcom.abctelecom.service.TicketService;

@RestController

@CrossOrigin

@RequestMapping("/tickets")

publicclassTicketController {

    privateTicketServiceticketService;

    @Autowired

    publicTicketController(TicketServiceticketService) {

        this.ticketService= ticketService;

    }

    @PostMapping("/new")

    publicStringraiseNew(@RequestBodyTicketticket) {

        returnticketService.save(ticket);

    }

    @GetMapping("/")

    publicList<Ticket>getAllTickets(){

        returnticketService.getAllTickets();

    }

    @GetMapping("/customer/{id}")

    publicList<Ticket>getTicketsByCustomerId(@PathVariableStringid){

        returnticketService.getTicketsByCustomerId(Long.parseLong(id));

    }

    @GetMapping("/{id}")

    publicTicketgetTicketById(@PathVariableStringid) {

        returnticketService.getTicketById(Long.parseLong(id));

    }

    @GetMapping("/notes/{ticketId}")

    publicList<Notes>getNotesByTicketId(@PathVariableStringticketId) {

        returnticketService.getNotesByTicketId(Long.parseLong(ticketId));

    }

    @PostMapping("/notes/save")

    publicStringsaveNotes(@RequestBodyNotesnotes) {

        returnticketService.saveNotes(notes);

    }

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

    publicFeedbackgetFeedbackByTicket(@PathVariableStringticketId ) {

        returnticketService.findFeedbackByTicket(Long.parseLong(ticketId));

    }

    @PostMapping("/feedback/submit")

    publicStringsubmitFeedback(@RequestBodyFeedbackfeedback) {

        returnticketService.saveFeedback(feedback);

    }

}

**Usercontroller.java**

packagecom.abctelecom.controller;

importjava.util.List;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.http.HttpStatus;

importorg.springframework.http.ResponseEntity;

importorg.springframework.web.bind.annotation.CrossOrigin;

importorg.springframework.web.bind.annotation.DeleteMapping;

importorg.springframework.web.bind.annotation.GetMapping;

importorg.springframework.web.bind.annotation.PathVariable;

importorg.springframework.web.bind.annotation.PostMapping;

importorg.springframework.web.bind.annotation.RequestBody;

importorg.springframework.web.bind.annotation.RequestMapping;

importorg.springframework.web.bind.annotation.RestController;

importcom.abctelecom.dto.CustomerDto;

importcom.abctelecom.entity.Customer;

importcom.abctelecom.entity.Engineer;

importcom.abctelecom.entity.Manager;

importcom.abctelecom.service.UserService;

@RestController

@CrossOrigin

@RequestMapping("/users")

publicclassUserController {

    @Autowired

    privateUserServiceuserService;

    @GetMapping("/")

    publicResponseEntity<String>apiTest(){

        returnnewResponseEntity<String>("Hi user", HttpStatus.OK);

    }

    @GetMapping("/zipcodes")

    publicList<String>getZipcodes(){

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

        returnuserService.getZipcodes();

    }

    @PostMapping("/customers/new")

    publicStringcreateCustomer(@RequestBodyCustomerDtocustomerDto) {

        Customercustomer=newCustomer(customerDto);

        returnuserService.addCustomer(customer);

    }

    @PostMapping("/engineers/new")

    publicStringcreateEngineer(@RequestBodyEngineerengineer) {

        returnuserService.addEngineer(engineer);

    }

    @PostMapping("/managers/new")

    publicStringcreateManager(@RequestBodyManagermanager) {

        returnuserService.addManager(manager);

    }

    @GetMapping("/customers")

    publicList<Customer>getAllCustomers(){

        returnuserService.getAllCustomers();

    }

    @GetMapping("customers/{id}")

    publicCustomergetCustomerById(@PathVariableStringid) {

        returnuserService.getCustomerById(Long.parseLong(id));

    }

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

    publicCustomerfindCustomerByUserId(@PathVariableStringid) {

        returnuserService.findCustomerByUserId(Long.parseLong(id));

    }

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

    publicEngineerfindEngineerByUserId(@PathVariableStringid) {

        returnuserService.findEngineerByUserId(Long.parseLong(id));

    }

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

    publicManagerfindBYUserId(@PathVariableStringid) {

        returnuserService.findManagerByUserId(Long.parseLong(id));

    }

    @GetMapping("engineers/{id}")

    publicEngineergetEngineerById(@PathVariableStringid) {

        returnuserService.getEngineerById(Long.parseLong(id));

    }

    @GetMapping("managers/{id}")

    publicManagergetManagerById(@PathVariableStringid) {

        returnuserService.getManagerById(Long.parseLong(id));

    }

    @GetMapping("/engineers")

    publicList<Engineer>getAllEngineers(){

        returnuserService.getAllEngineers();

    }

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

    publicList<Engineer>getEngineersByZipcode(@PathVariableStringzipcode){

        returnuserService.getEngineersByZipcode(zipcode);

    }

    @GetMapping("/managers")

    publicList<Manager>getAllManagers(){

        returnuserService.getAllManagers();

    }

    @DeleteMapping("/customers/{id}")

    publicStringdeleteCustomer(@PathVariableStringid) {

        returnuserService.deleteCustomer(Long.parseLong(id));

    }

}

1. **Dto**

**Customerdto.java**

packagecom.abctelecom.dto;

importjava.util.Set;

importcom.abctelecom.entity.User;

importlombok.Getter;

importlombok.Setter;

@Getter

@Setter

publicclassCustomerDto {

    privateLongid;

    privateUseruser;

    privateSet<String>serviceType;

    privateStringaddress;

    privateStringzipcode;

}

**Userdto.java**

packagecom.abctelecom.dto;

importlombok.Getter;

importlombok.Setter;

@Getter

@Setter

publicclassUserDto {

    privateStringemail;

    privateStringpassword;

}

**AbcTelecomeApplication.java**

packagecom.abctelecom;

importorg.springframework.boot.SpringApplication;

importorg.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

publicclassAbcTelecomApplication {

    publicstaticvoidmain(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"?>

<projectxmlns="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>