|  |
| --- |
| **SimpliLearn Capstone Project: ComplaintRedressalSystem Complaint Management System for ABC Telecom Ltd.**  **Submitted by: Seema Bhuyan** |
| GitHub Project Repository URL :  <https://github.com/Seema-beep/SimplilearnComplaintRedressalSystemJobReadyPhase.git> |
| **Backend Rest API:**  **Project structure:** |
| **ComplaintRedressalSystemApiApplication.java** package com.simplilearn.crs;    import java.util.ArrayList;  import java.util.List;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.boot.CommandLineRunner; import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.context.annotation.Bean;  import org.springframework.security.crypto.password.PasswordEncoder;    import com.simplilearn.crs.entities.Engineer; import com.simplilearn.crs.entities.Manager; import com.simplilearn.crs.entities.Users; import com.simplilearn.crs.entities.pin; import com.simplilearn.crs.repository.engineerRepo; import com.simplilearn.crs.repository.managerRepo; |

|  |
| --- |
| import com.simplilearn.crs.repository.pinRepo; import com.simplilearn.crs.repository.usersRepo; import com.simplilearn.crs.services.usersService;      @SpringBootApplication    public class ComplaintRedressalSystemApiApplication implements CommandLineRunner{    @Autowired  engineerRepo repo; @Autowired pinRepo pRepo;  @Autowired  PasswordEncoder encoder;  @Autowired  usersService usrservice;    public static void main(String[] args) {  SpringApplication.run(ComplaintRedressalSystemApiApplication.class, args);  }    @Override  public void run(String... args) throws Exception {    }    } |
| **ComplaintController.java**  package com.simplilearn.crs.controllers;    import java.util.HashMap; import java.util.List;  import java.util.Map;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.security.access.prepost.PreAuthorize; import org.springframework.web.bind.annotation.GetMapping; 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.RequestParam; import org.springframework.web.bind.annotation.RestController;    import com.simplilearn.crs.entities.Complaint; import com.simplilearn.crs.entities.Manager; import com.simplilearn.crs.entities.Ticket; import com.simplilearn.crs.entities.pin; import com.simplilearn.crs.services.complaintService; import com.simplilearn.crs.services.managerService; import com.simplilearn.crs.services.pinService; import com.simplilearn.crs.services.ticketService; |

|  |
| --- |
| @RestController  @RequestMapping("api/complaint")    public class ComplaintController {  @Autowired  complaintService complaintservice;    @Autowired  ticketService ticketservice;    @Autowired  managerService managerservice;    @Autowired  pinService pinservice;    @PreAuthorize("hasAuthority('ADMIN')")  @GetMapping("/hi")  public String Handshake() {  return "HI!!";  }    @PreAuthorize("hasAnyAuthority('ADMIN','CUSTOMER')")  @PostMapping("/raise")  public ResponseEntity<Map<String,Long>> raiseComplaint(@RequestBody Complaint cmp){  Map<String,Long> res = new HashMap<>();    try{  Complaint createdcomplaint = complaintservice.createComplaint(cmp);  Ticket tempticket =new Ticket();  Manager tempmgr = managerservice.getManagerForPin(cmp.getPin()); tempticket.setStatus("RAISED");  tempticket.setManager(tempmgr);  Ticket createdticket = ticketservice.createTicket(tempticket); complaintservice.addTicketToComplaint(createdcomplaint,createdticket); ticketservice.setComplaint(createdticket, createdcomplaint); res.put("status",1L);  res.put("complaintId", createdcomplaint.getComplaintId()); res.put("ticketId",createdticket.getTicketId());  return new ResponseEntity<Map<String,Long>>(res,HttpStatus.OK);  }catch(Exception e) { res.put("status",0L);  return new  ResponseEntity<Map<String,Long>>(res,HttpStatus.EXPECTATION\_FAILED);  }  }  @PreAuthorize("hasAnyAuthority('ADMIN','MANAGER')")  @GetMapping("/all")  public List<Complaint> getAllComplaints(){ return complaintservice.getAllComplaints();  }  @PreAuthorize("hasAnyAuthority('ADMIN','MANAGER','CUSTOMER')") @GetMapping("/cuscomplaints") |

|  |
| --- |
| public List<Complaint> getAllCustomerComplaints(@RequestParam(name = "cid") Long cusId){  return complaintservice.getAllCustomerComplaints(cusId);  }    @PreAuthorize("hasAnyAuthority('ADMIN','CUSTOMER')")  @PostMapping("/ticketreraise")  public ResponseEntity<Map<String,Long>> reRaiseTicket(@RequestBody Complaint cmp){  Map<String,Long> res = new HashMap<>();  try {  Ticket tempticket = new Ticket();  Manager tempmgr = managerservice.getManagerForPin(cmp.getPin()); tempticket.setStatus("RAISED");  tempticket.setManager(tempmgr);  Ticket createdticket = ticketservice.createTicket(tempticket); complaintservice.addTicketToComplaint(cmp, createdticket); ticketservice.setComplaint(createdticket, cmp); res.put("status", 1L);  res.put("complaintId", cmp.getComplaintId()); res.put("ticketId", createdticket.getTicketId());  return new ResponseEntity<Map<String,Long>>(res,HttpStatus.OK);  }catch(Exception e) { res.put("status",0L);  return new  ResponseEntity<Map<String,Long>>(res,HttpStatus.EXPECTATION\_FAILED);  }  }  @PreAuthorize("hasAnyAuthority('ADMIN','CUSTOMER')")  @PostMapping("/addfeedback")  public ResponseEntity<Map<String,Long>> addFeedback(@RequestBody Complaint cmp){  Map<String,Long> res = new HashMap<>();  Long status = complaintservice.addFeedback(cmp.getComplaintId(), cmp.getFeedback());  res.put("status", status);  return new ResponseEntity<Map<String,Long>>(res,HttpStatus.OK);  }  @GetMapping("/allpin")  @PreAuthorize("hasAnyAuthority('ADMIN','ENGINEER','MANAGER','CUSTOMER')") public List<pin> getAllPins(){  return pinservice.getAllPin();  }  @PostMapping("/addpin")  @PreAuthorize("hasAuthority('ADMIN')")  public ResponseEntity<Map<String,Integer>> addpin(@RequestBody pin p){  Map<String,Integer> res = new HashMap<>(); int status = pinservice.addPin(p);  res.put("status", status);    return new ResponseEntity<Map<String,Integer>>(res,HttpStatus.OK);  }  @GetMapping("/managerforpin") @PreAuthorize("hasAuthority('ADMIN')")  public ResponseEntity<Manager> getManagerForpin(@RequestParam(name="pin") long pin){ |

|  |
| --- |
| Manager m = managerservice.getManagerForPin(new pin(pin)); return new ResponseEntity<Manager>(m,HttpStatus.OK);  }  } |
| **homeController.java**  package com.simplilearn.crs.controllers;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus; import org.springframework.http.HttpStatusCode; import org.springframework.http.ResponseEntity;  import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.web.bind.annotation.GetMapping; 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.simplilearn.crs.entities.Users; import com.simplilearn.crs.services.usersService;    @RestController  @RequestMapping("api/home")  public class homeController {    @Autowired  usersService service;  @Autowired  PasswordEncoder encoder;    @GetMapping("/")  public String welcome() {  return "Hi";  }  @PostMapping("/add")  public ResponseEntity<String> adduser(@RequestBody Users usr){ System.out.println("Method initiated, Receivedobject: "+usr); String password = usr.getPassword();  usr.setPassword(encoder.encode(password));  service.addUser(usr);  return new ResponseEntity<String>("Success!",HttpStatus.OK);  }  @GetMapping("/user")  public ResponseEntity<String> user(){  return new ResponseEntity<>("Welcome User!",HttpStatus.OK);  }    @GetMapping("/admin")  public ResponseEntity<String> admin(){  return new ResponseEntity<>("Welcome Admin!",HttpStatus.OK);  }  } |

|  |
| --- |
| **TicketController.java**  package com.simplilearn.crs.controllers;    import java.util.HashMap; import java.util.List;  import java.util.Map;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity;  import org.springframework.security.access.prepost.PreAuthorize; import org.springframework.security.core.Authentication; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.PatchMapping; import org.springframework.web.bind.annotation.RequestBody; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.RestController;    import com.simplilearn.crs.entities.Complaint; import com.simplilearn.crs.entities.Engineer; import com.simplilearn.crs.entities.Ticket; import com.simplilearn.crs.entities.Users; import com.simplilearn.crs.entities.pin; import com.simplilearn.crs.security.userSecurityObj; import com.simplilearn.crs.services.engineerService; import com.simplilearn.crs.services.ticketService; import com.simplilearn.crs.services.usersService;    @RestController  @RequestMapping("api/ticket")  public class TicketController {  @Autowired  ticketService tktservice;  @Autowired  engineerService engineerservice;  @Autowired  usersService usrService;    @PreAuthorize("hasAnyAuthority('ADMIN','MANAGER')")  @GetMapping("/all")  public List<Ticket> getAllTicket(){  return tktservice.getAllTickets();  }    @PreAuthorize("hasAnyAuthority('ADMIN','CUSTOMER','ENGINEER','MANAGER')")  @GetMapping("/ticketdata")  public Ticket getTicketData(@RequestParam(name="ticketId")Long ticketID) { return tktservice.getTicket(ticketID);  }  @PreAuthorize("hasAnyAuthority('ADMIN','CUSTOMER','ENGINEER','MANAGER')")  @GetMapping("/complaintdata")  public Complaint getComplaintData(@RequestParam(name="ticketId")Long ticketID) { return tktservice.getTicket(ticketID).getComplaint();  } |

|  |
| --- |
| @PatchMapping("/updatestatus")  @PreAuthorize("hasAnyAuthority('ADMIN','ENGINEER','MANAGER')")  public ResponseEntity<Map<String,Long>> statusUpdate (@RequestBody Ticket tkt){  Map<String,Long> res = new HashMap<>();  Long status = tktservice.updateStatus(tkt.getTicketId(), tkt.getStatus()); res.put("status", status);  return new ResponseEntity<Map<String,Long>>(res,HttpStatus.OK);  }  @PatchMapping("/updatecomment")  @PreAuthorize("hasAnyAuthority('ADMIN','ENGINEER','MANAGER')")  public ResponseEntity<Map<String,Long>> commentUpdate (@RequestBody Ticket tkt){  Map<String,Long> res = new HashMap<>();  Long status = tktservice.updateComment(tkt.getTicketId(), tkt.getComment()); res.put("status", status);  return new ResponseEntity<Map<String,Long>>(res,HttpStatus.OK);  }  @PatchMapping("/assignEngineer")  @PreAuthorize("hasAnyAuthority('ADMIN','MANAGER')")  public ResponseEntity<Map<String,Long>> assignEngineer(@RequestBody Ticket tkt){  Map<String,Long> res = new HashMap<>(); Long status = tktservice.assignEngineer(tkt); res.put("status", status);  return new ResponseEntity<Map<String,Long>>(res,HttpStatus.OK);  }  @GetMapping("/engineeralltickets")  @PreAuthorize("hasAnyAuthority('ADMIN','ENGINEER','MANAGER')")  public ResponseEntity<List<Ticket>> getEngineerTickets(@RequestParam(name="engineerid") Long engineerId){  List<Ticket> engrTickets = tktservice.getEngTickets(engineerId); return new ResponseEntity<List<Ticket>>(engrTickets,HttpStatus.OK);  }    @GetMapping("/engineeropentickets")  @PreAuthorize("hasAnyAuthority('ADMIN','ENGINEER','MANAGER')") public ResponseEntity<List<Ticket>> getEngineerOpenTickets(@RequestParam(name="engineerid") Long engineerId){ List<Ticket> engrTickets = tktservice.getEngOpenTickets(engineerId); return new ResponseEntity<List<Ticket>>(engrTickets,HttpStatus.OK);  }  @GetMapping("/manageropentickets")  @PreAuthorize("hasAnyAuthority('ADMIN','MANAGER')")  public ResponseEntity<List<Ticket>>  getManagerOpenTickets(@RequestParam(name="managerid") Long managerId){ List<Ticket> mgrTickets = tktservice.getManagerOpenTickets(managerId); return new ResponseEntity<List<Ticket>>(mgrTickets,HttpStatus.OK);  }  @GetMapping("/manageralltickets")  @PreAuthorize("hasAnyAuthority('ADMIN','MANAGER')")  public ResponseEntity<List<Ticket>> getManagerTickets(@RequestParam(name="managerid") Long managerId){  List<Ticket> mgrTickets = tktservice.getManagerTickets(managerId); return new ResponseEntity<List<Ticket>>(mgrTickets,HttpStatus.OK);  }  @GetMapping("/engineerforpin")  @PreAuthorize("hasAnyAuthority('ADMIN','MANAGER')") |

|  |
| --- |
| public ResponseEntity<List<Engineer>> getEngineerForpin(@RequestParam(name="pin") Long  Pin){  List<Engineer> location\_engineers = engineerservice.getEngineerForPin(new pin(Pin));  return new ResponseEntity<List<Engineer>>(location\_engineers,HttpStatus.OK);  }  @GetMapping("/getcurrentuser")  @PreAuthorize("hasAnyAuthority('ADMIN','MANAGER','ENGINEER','CUSTOMER')") public Users getCurrentUser(Authentication authentication) {  userSecurityObj t = (userSecurityObj) authentication.getPrincipal();  Users usr = usrService.findUser(t.getUsername());  return usr;  }  @PatchMapping("/assignmanager") @PreAuthorize("hasAuthority('ADMIN')")  public ResponseEntity<Map<String,Long>> assignManager(@RequestBody Ticket tkt){ Map<String,Long> status = new HashMap<>(); status.put("status", tktservice.assignManager(tkt)); return new ResponseEntity<>(status,HttpStatus.ACCEPTED);  }  } |
| **userController.java**  package com.simplilearn.crs.controllers;    import java.util.HashMap; import java.util.List;  import java.util.Map;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.HttpStatus; import org.springframework.http.HttpStatusCode; import org.springframework.http.ResponseEntity;  import org.springframework.security.access.prepost.PreAuthorize; import org.springframework.security.core.Authentication; import org.springframework.security.core.userdetails.UserDetails; import org.springframework.web.bind.annotation.DeleteMapping; import org.springframework.web.bind.annotation.GetMapping; 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.RequestParam; import org.springframework.web.bind.annotation.RestController;    import com.simplilearn.crs.entities.Customer; import com.simplilearn.crs.entities.Engineer; import com.simplilearn.crs.entities.Manager; import com.simplilearn.crs.entities.Users; import com.simplilearn.crs.repository.customerRepo; import com.simplilearn.crs.security.userSecurityObj; import com.simplilearn.crs.services.customerService; import com.simplilearn.crs.services.engineerService; import com.simplilearn.crs.services.managerService; import com.simplilearn.crs.services.usersService;    @RestController |

|  |
| --- |
| @RequestMapping("api/user")  @PreAuthorize("hasAuthority('ADMIN')") public class userController {    @Autowired  usersService usrService;  @Autowired  managerService mgrService;  @Autowired  engineerService engrService;  @Autowired  customerService cusService;      @GetMapping("/")  public String hanshake() {  return "Hello!";  }    @GetMapping("/all")  public ResponseEntity<List<Users>> getAllUsers() { List<Users> listOfUsers = usrService.findAllUsers();  ResponseEntity<List<Users>> res = new ResponseEntity<List<Users>>(listOfUsers,  HttpStatus.OK);  return res;  }      @PostMapping("/adduser")  public ResponseEntity<Map<String, Integer>> addUser(@RequestBody Users user) {  Map<String, Integer> status = new HashMap<>(); status.put("status", usrService.addUser(user));  return new ResponseEntity<Map<String, Integer>>(status, HttpStatus.OK);  }      @DeleteMapping("/deleteuser")  public ResponseEntity<Map<String, Integer>> deleteUser(@RequestParam("userid") long userid ) {  Map<String, Integer> status = new HashMap<>(); status.put("status", usrService.deleteUser(userid));  return new ResponseEntity<Map<String, Integer>>(status, HttpStatus.OK);  }      @PostMapping("/addmanager")  public ResponseEntity<Map<String, Integer>> addManager(@RequestBody Manager mgr)  {  Map<String, Integer> status = new HashMap<>(); status.put("status", mgrService.addManager(mgr));  return new ResponseEntity<Map<String, Integer>>(status, HttpStatus.OK);  }      @PostMapping("/addengineer") |

|  |
| --- |
| public ResponseEntity<Map<String, Integer>> addEngineer(@RequestBody Engineer eng)  {  Map<String, Integer> status = new HashMap<>(); status.put("status", engrService.addEngineer(eng));  return new ResponseEntity<Map<String, Integer>>(status, HttpStatus.OK);  }      @PostMapping("/addcustomer")  public ResponseEntity<Map<String, Integer>> addCustomer(@RequestBody Customer cus) {  Map<String, Integer> status = new HashMap<>(); status.put("status", cusService.addCustomer(cus));  return new ResponseEntity<Map<String, Integer>>(status, HttpStatus.OK);  }    @GetMapping("/allmanagers")  public ResponseEntity<List<Manager>> getAllManagers() { List<Manager> mgrs = mgrService.getAllManagers();  return new ResponseEntity<List<Manager>>(mgrs, HttpStatus.OK);  }    @GetMapping("/allengineers")  public ResponseEntity<List<Engineer>> getAllEngineer(){ List<Engineer> egrs = engrService.findAllEngineers();  return new ResponseEntity<List<Engineer>>(egrs,HttpStatus.OK);  }    @GetMapping("/allcustomers")  public ResponseEntity<List<Customer>> getAllCustomers(){ List<Customer> customers = cusService.getAllCustomers();  return new ResponseEntity<List<Customer>>(customers,HttpStatus.OK);  }      } |
| **Complaint.java**  package com.simplilearn.crs.entities;    import java.time.LocalDateTime;  import java.util.List;    import org.hibernate.annotations.CreationTimestamp; import org.hibernate.annotations.UpdateTimestamp;    import jakarta.persistence.Column; import jakarta.persistence.Entity; import jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType; import jakarta.persistence.Id; import jakarta.persistence.ManyToOne; import jakarta.persistence.OneToMany; import lombok.Data;    @Entity |

|  |
| --- |
| @Data  public class Complaint {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY) private long complaintId; private String name; private String address;  @ManyToOne  private pin pin; private long contactNo;  @ManyToOne  private Customer customer; private String subject; private String details; private String feedback;  @OneToMany(mappedBy = "complaint")  private List<Ticket> ticket;  @Column(updatable = false) @CreationTimestamp  private LocalDateTime createdAt;  @UpdateTimestamp  private LocalDateTime lastUpdated;    public void addTicket(Ticket tkt) { this.ticket.add(tkt);  }  } |
| **Customer.java**  package com.simplilearn.crs.entities;    import jakarta.persistence.Entity; import jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType; import jakarta.persistence.Id;  import jakarta.persistence.OneToOne;    @Entity  public class Customer extends Users {    private String address;  @OneToOne  private pin pin; private long contactNo;      } |
| **Engineer.java**  package com.simplilearn.crs.entities;    import java.io.Serializable; import java.util.List;    import jakarta.persistence.Column; import jakarta.persistence.Entity; import jakarta.persistence.GeneratedValue; |

|  |
| --- |
| import jakarta.persistence.GenerationType; import jakarta.persistence.Id;  import jakarta.persistence.ManyToMany; import lombok.Data;    @Entity @Data  public class Engineer extends Users {        /\*\*  \* \*/  private static final long serialVersionUID =2899903505215516240L;  @ManyToMany  private List<pin> locations;  private Long contactNo;      public void addPin(pin pin) {  this.locations.add(pin);  }      @Override  public String toString() {  return "Engineer [locations=" + locations + ", contactNo=" + contactNo + super.toString()  + ", getClass()=" + getClass() + "]";  }    } |
| **Manager.java**  package com.simplilearn.crs.entities;    import jakarta.persistence.Entity; import jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType; import jakarta.persistence.OneToOne; import lombok.Data;  import lombok.EqualsAndHashCode;    @Entity  @Data    public class Manager extends Users {    @OneToOne  private pin pin; private long contactNo;      } |

|  |
| --- |
| **Pin.java**  package com.simplilearn.crs.entities;    import java.io.Serializable;    import jakarta.persistence.Entity; import jakarta.persistence.Id;  import lombok.Data;    @Entity @Data  public class pin implements Serializable{    @Id    private long pin; public pin(long pin) { this.pin=pin;  }  public pin() {  // TODO Auto-generated constructor stub  }  public pin(Long pin) {  this.pin=pin;  }    } |
| **Ticket.java**  package com.simplilearn.crs.entities;    import java.time.LocalDateTime;    import org.hibernate.annotations.CreationTimestamp; import org.hibernate.annotations.UpdateTimestamp;    import com.fasterxml.jackson.annotation.JsonIgnore;    import jakarta.persistence.Column; import jakarta.persistence.Entity; import jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType; import jakarta.persistence.Id; import jakarta.persistence.ManyToOne; import jakarta.persistence.OneToMany; import jakarta.persistence.OneToOne; import lombok.Data;    @Entity  @Data  public class Ticket {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY) private long ticketId; private String status;  @JsonIgnore |

|  |
| --- |
| @ManyToOne  private Complaint complaint;  @ManyToOne  private Manager manager;  @ManyToOne  private Engineer engineer; private String comment; @Column(updatable = false) @CreationTimestamp  private LocalDateTime createdAt;  @UpdateTimestamp  private LocalDateTime updatedAt;  } |
| **Users.java**  package com.simplilearn.crs.entities;    import java.io.Serializable;    import jakarta.persistence.Column; import jakarta.persistence.Entity; import jakarta.persistence.GeneratedValue; import jakarta.persistence.GenerationType; import jakarta.persistence.Id; import jakarta.persistence.Inheritance; import jakarta.persistence.InheritanceType; import lombok.Data;    @Entity  @Data  @Inheritance(strategy = InheritanceType.JOINED) public class Users implements Serializable{  @Id  @GeneratedValue(strategy = GenerationType.SEQUENCE) private long userid; private String firstName; private String lastName; @Column(unique = true) private String username; private String Password; private String email; private String roles;  private boolean accountStatus;    } |
| **complaintRepo.java**  package com.simplilearn.crs.repository;    import java.util.List;    import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;    import com.simplilearn.crs.entities.Complaint; import com.simplilearn.crs.entities.Customer; |

|  |
| --- |
| @Repository  public interface complaintRepo extends JpaRepository<Complaint, Long> {    public List<Complaint> findByCustomer(Customer customer);  } |
| **customerRepo.java**  package com.simplilearn.crs.repository;    import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;    import com.simplilearn.crs.entities.Customer;    @Repository  public interface customerRepo extends JpaRepository<Customer, Long> {    } |
| **engineerRepo.java**  package com.simplilearn.crs.repository;    import java.util.List;    import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;    import com.simplilearn.crs.entities.Engineer; import com.simplilearn.crs.entities.pin;    @Repository  public interface engineerRepo extends JpaRepository<Engineer, Long> {    public List<Engineer> findByLocations(pin locations);  } |
| **managerRepo.java**  package com.simplilearn.crs.repository;    import java.util.List;    import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;    import com.simplilearn.crs.entities.Manager; import com.simplilearn.crs.entities.pin;    @Repository  public interface managerRepo extends JpaRepository<Manager, Long> {    public Manager findByPin(pin pin);    } |

|  |
| --- |
| **pinRepo.java**  package com.simplilearn.crs.repository;    import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;    import com.simplilearn.crs.entities.pin;    @Repository  public interface pinRepo extends JpaRepository<pin, Long> {    } |
| **ticketRepo.java**  package com.simplilearn.crs.repository;    import java.util.List;    import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.data.jpa.repository.Query; import org.springframework.stereotype.Repository;    import com.simplilearn.crs.entities.Engineer; import com.simplilearn.crs.entities.Manager; import com.simplilearn.crs.entities.Ticket;    @Repository  public interface ticketRepo extends JpaRepository<Ticket, Long> {    public List<Ticket> findByEngineer(Engineer engineer);  @Query("Select t from com.simplilearn.crs.entities.Ticket t where t.engineer=:engineer and not t.status ='CLOSED'")  public List<Ticket> findByEngineerOpen(Engineer engineer); public List<Ticket> findByManager(Manager manager);  @Query("Select t from com.simplilearn.crs.entities.Ticket t where t.manager=:manager and not t.status ='CLOSED'")  public List<Ticket> findByManagerOpen(Manager manager);  } |
| **usersRepo.java**  package com.simplilearn.crs.repository;    import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.stereotype.Repository;    import com.simplilearn.crs.entities.Users;    @Repository  public interface usersRepo extends JpaRepository<Users, Long> { public Users findByUsername(String Username);  } |
| **CustomUserDetailsService.java**  package com.simplilearn.crs.security;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.core.userdetails.UserDetails; import org.springframework.security.core.userdetails.UserDetailsService; |

|  |
| --- |
| import org.springframework.security.core.userdetails.UsernameNotFoundException; import org.springframework.stereotype.Component;    import com.simplilearn.crs.entities.Users; import com.simplilearn.crs.repository.usersRepo;  @Component  public class CustomUserDetailsService implements UserDetailsService{    @Autowired  usersRepo repo;    @Override  public UserDetails loadUserByUsername(String username) throws  UsernameNotFoundException {  System.out.println("Finding by username ----->"+username);  if(username!=null && username!="") { Users usr = repo.findByUsername(username);    if(usr!=null) {  userSecurityObj securityUser = new userSecurityObj(usr); System.out.println("Created user----------->"+securityUser); return securityUser;  }else {  System.out.println("User not Found!!");  throw new UsernameNotFoundException("User not present!");  }  }else {  System.out.println("Username not Found!!");  throw new UsernameNotFoundException("Username not present!");  }  }    } |
| **SecurityConfig.java**  package com.simplilearn.crs.security;  import static org.springframework.security.config.Customizer.withDefaults;    import java.io.IOException; import java.util.Arrays; import java.util.Collections; import java.util.List;      import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.http.HttpMethod;  import org.springframework.security.authentication.AuthenticationManager; import org.springframework.security.authentication.AuthenticationProvider; import org.springframework.security.authentication.dao.DaoAuthenticationProvider; import  org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfi guration; import  org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity; import org.springframework.security.config.annotation.web.builders.HttpSecurity; |

|  |
| --- |
| import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity; import org.springframework.security.config.http.SessionCreationPolicy; import org.springframework.security.core.Authentication; import org.springframework.security.core.AuthenticationException; import org.springframework.security.core.userdetails.UserDetailsService; import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.security.web.SecurityFilterChain; import org.springframework.security.web.authentication.AuthenticationFailureHandler; import org.springframework.security.web.authentication.AuthenticationSuccessHandler; import org.springframework.web.cors.CorsConfiguration; import org.springframework.web.cors.CorsConfigurationSource;  import org.springframework.web.cors.UrlBasedCorsConfigurationSource;    import jakarta.servlet.ServletException; import jakarta.servlet.http.Cookie; import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse;        @Configuration  @EnableWebSecurity  @EnableMethodSecurity(prePostEnabled = true) public class SecurityConfig {    @Bean  public SecurityFilterChain filterchain(HttpSecurity http) throws Exception {  http  .csrf((csrf)->csrf.disable())  .cors(cors -> {  cors.configurationSource(request -> {  CorsConfiguration corsConfiguration = new CorsConfiguration();    corsConfiguration.setAllowedOrigins(Collections.singletonList("http://localhost:4200")); corsConfiguration.setAllowedMethods(Arrays.asList("GET", "POST",  "PUT","PATCH", "DELETE"));  corsConfiguration.setAllowedHeaders(Collections.singletonList("\*")); corsConfiguration.setAllowCredentials(true);  return corsConfiguration;  });  })  .authorizeHttpRequests((authorize) -> authorize  .anyRequest().authenticated()  )  .authenticationProvider(authProvider())  .sessionManagement((sessmgmt)->  sessmgmt  .sessionCreationPolicy(SessionCreationPolicy.ALWAYS)  .maximumSessions(1)  )    .logout((logout) ->  logout.deleteCookies("JSESSIONID")  .invalidateHttpSession(true) |

|  |
| --- |
| .logoutUrl("/logout")  )  .formLogin((login)->{ login.loginProcessingUrl("/auth") .usernameParameter("username")  .passwordParameter("password")  .successHandler(successHandler())  .failureHandler(failureHandler());  });  return http.build();  }  private AuthenticationSuccessHandler successHandler() { return new AuthenticationSuccessHandler() {      @Override  public void onAuthenticationSuccess(HttpServletRequest request,  HttpServletResponse response,  Authentication authentication) throws IOException,  ServletException {  userSecurityObj usr = (userSecurityObj) authentication.getPrincipal();    // Serialize the roles as a string  String roles = authentication.getAuthorities().toString();    response.getWriter().append("{\"authentication\":\""+usr.getUsername()+ "\","  +"\"roles\":\""+roles+"\""  + "}");  response.setStatus(HttpServletResponse.SC\_OK);    }  };  }    private AuthenticationFailureHandler failureHandler() { return new AuthenticationFailureHandler() {    @Override  public void onAuthenticationFailure(HttpServletRequest request,  HttpServletResponse response,  AuthenticationException exception) throws IOException,  ServletException {    response.getWriter().append("Authentication failure"); response.setStatus(401);  }  };    }    @Bean  public UserDetailsService userDetailsService() {  return new CustomUserDetailsService();  } |

|  |
| --- |
| @Bean  public AuthenticationProvider authProvider() {  DaoAuthenticationProvider provider = new DaoAuthenticationProvider(); provider.setUserDetailsService(userDetailsService());  provider.setPasswordEncoder(encoder());  return provider;  }    @Bean  public PasswordEncoder encoder () { return new BCryptPasswordEncoder();  }    @Bean  public AuthenticationManager authManager(AuthenticationConfiguration config) throws Exception {  return config.getAuthenticationManager();  }  } |
| **userSecurityObj.java**  package com.simplilearn.crs.security;    import java.util.Arrays; import java.util.Collection; import java.util.List;  import java.util.stream.Collectors;    import org.springframework.security.core.GrantedAuthority;  import org.springframework.security.core.authority.SimpleGrantedAuthority; import org.springframework.security.core.userdetails.UserDetails;    import com.simplilearn.crs.entities.Users;    public class userSecurityObj implements UserDetails{ private Users usr; userSecurityObj(Users usr){  this.usr =usr;  }  @Override  public Collection<? extends GrantedAuthority> getAuthorities() {  String[] role = usr.getRoles().split(",");  List<GrantedAuthority> authorities =  Arrays.stream(role).map(SimpleGrantedAuthority::new).collect(Collectors.toList());  System.out.println(authorities);  return authorities;  }    @Override  public String getPassword() {  return usr.getPassword();  }    @Override  public String getUsername() { return usr.getUsername(); |

|  |
| --- |
| }    @Override  public boolean isAccountNonExpired() {  return usr.isAccountStatus();  }    @Override  public boolean isAccountNonLocked() {  return usr.isAccountStatus();  }    @Override  public boolean isCredentialsNonExpired() {  return usr.isAccountStatus();  }    @Override  public boolean isEnabled() {  return usr.isAccountStatus();  }      } |
| **complaintService.java**  package com.simplilearn.crs.services;    import java.util.List;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;    import com.simplilearn.crs.entities.Complaint; import com.simplilearn.crs.entities.Customer; import com.simplilearn.crs.entities.Ticket; import com.simplilearn.crs.repository.complaintRepo; import com.simplilearn.crs.repository.customerRepo;    @Service  public class complaintService {    @Autowired  complaintRepo repo;    @Autowired  customerRepo crepo;    public List<Complaint> getAllComplaints(){ return repo.findAll();  }    public Complaint createComplaint(Complaint cmp) { try {  return repo.save(cmp); |

|  |
| --- |
| }catch(Exception e) {  return null;  }  }    public int addTicketToComplaint(Complaint cmp, Ticket tkt) {  try {  cmp.addTicket(tkt); repo.save(cmp); return 1;  }catch(Exception e) {  return 0;  }  }  public List<Complaint> getAllCustomerComplaints(Long CustomerId){  Customer cus = crepo.findById(CustomerId).get();  return repo.findByCustomer(cus);  }  public Long addFeedback(Long cId, String feedback) {  try {  System.out.println("complaintid-->"+cId);  System.out.println("Feedback--------->"+feedback);  Complaint cmp =repo.findById(cId).get();  cmp.setFeedback(feedback); repo.save(cmp);  return 1L;  }catch(Exception e) {  System.out.println(e);  return 0L;  }  }  } |
| **customerService.java**  package com.simplilearn.crs.services;    import java.util.List;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.stereotype.Service;    import com.simplilearn.crs.entities.Customer; import com.simplilearn.crs.repository.customerRepo;    @Service  public class customerService {  @Autowired  customerRepo repo;  @Autowired  PasswordEncoder encoder;  public List<Customer> getAllCustomers() { return repo.findAll();  }    public int addCustomer(Customer cus) { |

|  |
| --- |
| try {  cus.setPassword(encoder.encode(cus.getPassword()));;  repo.save(cus); return 1;  } catch (Exception e) {  return 0;  }  }    public int deleteCustomer(Customer cus) {  try {  repo.delete(cus); return 1;  }catch(Exception e) {  return 0;  }  }  } |
| **engineerService.java**  package com.simplilearn.crs.services;    import java.util.List;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.stereotype.Service;    import com.simplilearn.crs.entities.Engineer; import com.simplilearn.crs.entities.pin; import com.simplilearn.crs.repository.engineerRepo; import com.simplilearn.crs.repository.pinRepo;    @Service  public class engineerService {  @Autowired  engineerRepo repo;    @Autowired  pinRepo prepo;    @Autowired  PasswordEncoder encoder;    public List<Engineer> findAllEngineers() { return repo.findAll();  }    public int addEngineer(Engineer eng) {  try {  System.out.println(eng);  eng.setPassword(encoder.encode(eng.getPassword()));  repo.save(eng); return 1;  }catch(Exception e) {  System.out.println(e); |

|  |
| --- |
| return 0;  }  }    public int deleteEngineer(Engineer eng) {  try {  repo.delete(eng); return 1;  }catch(Exception e) {  return 0;  }  }    public int allocatePin(Long pin, Engineer eng) {  try {  pin p = prepo.findById(pin).get();  eng.addPin(p); repo.save(eng); return 1;  }catch(Exception e) {  return 0;  }  }    public List<Engineer> getEngineerForPin(pin pin){  try {  return repo.findByLocations(pin);  }catch(Exception e) {  System.out.println(2);  return null;  }  }  } |
| **managerService.java**  package com.simplilearn.crs.services;    import java.util.List;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.stereotype.Service;    import com.simplilearn.crs.entities.Manager; import com.simplilearn.crs.entities.pin;  import com.simplilearn.crs.repository.managerRepo;    @Service  public class managerService {    @Autowired  managerRepo repo;    @Autowired  PasswordEncoder encoder; public List<Manager>getAllManagers() { |

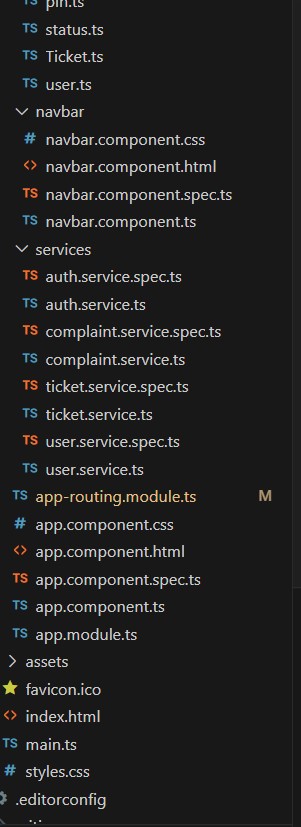
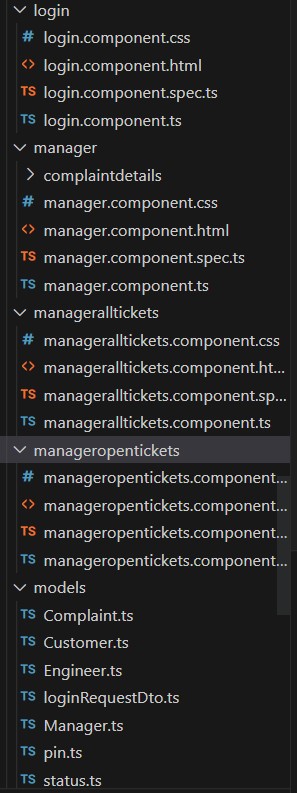
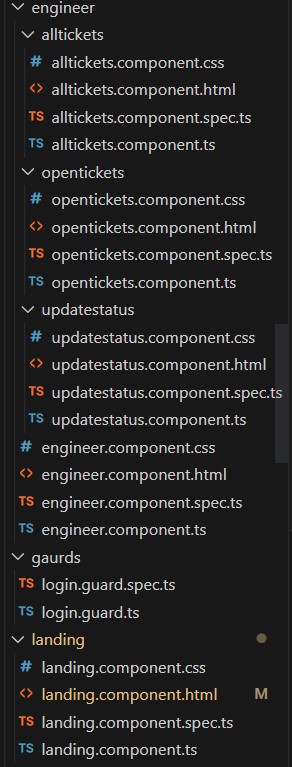
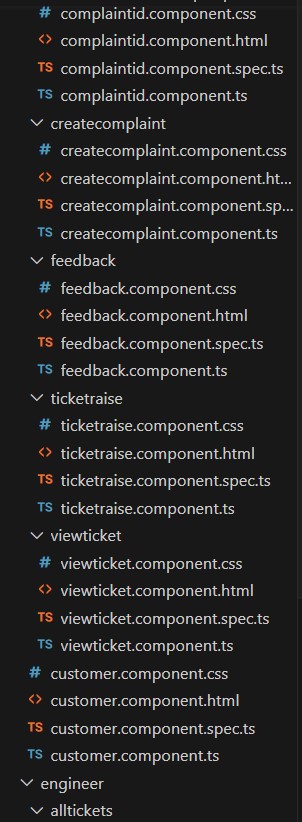
|  |
| --- |
| return repo.findAll();  }    public int addManager(Manager mgr) {  try {  mgr.setPassword(encoder.encode(mgr.getPassword()));;  repo.save(mgr); return 1;  }catch(Exception e) {  System.out.println(e);  return 0;  }  }    public int deleteManager(Manager mgr) {  try {  repo.delete(mgr); return 1;  }catch(Exception e) {  return 0;  }  }    public int setPin(Long managerId, pin pin) {  try {  Manager mgr = repo.findById(managerId).get(); mgr.setPin(pin); return 1;  }catch(Exception e) {  return 0;  }  }  public Manager getManagerForPin(pin pin) {  try {  return repo.findByPin(pin); }catch(Exception e) {  System.out.println(e);  return null;  }  }  } |
| **pinService.java**  package com.simplilearn.crs.services;    import java.util.List;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;    import com.simplilearn.crs.entities.pin;  import com.simplilearn.crs.repository.pinRepo;    @Service  public class pinService { |

|  |
| --- |
| @Autowired  private pinRepo repo;    public int addPin(pin pin) {  try {  repo.save(pin); return 1;  } catch (Exception e) {  return 0;  }  }    public List<pin> getAllPin() {  return repo.findAll();  }    public int deletePin(pin pin) {  try {  repo.delete(pin); return 1;  } catch (Exception e) {  return 0;  }  }  } |
| **ticketService.java**  package com.simplilearn.crs.services;    import java.util.List;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;    import com.simplilearn.crs.entities.Complaint; import com.simplilearn.crs.entities.Engineer; import com.simplilearn.crs.entities.Manager; import com.simplilearn.crs.entities.Ticket; import com.simplilearn.crs.repository.engineerRepo; import com.simplilearn.crs.repository.managerRepo; import com.simplilearn.crs.repository.ticketRepo;    @Service  public class ticketService {  @Autowired  ticketRepo repo;    @Autowired  engineerRepo erepo;    @Autowired  managerRepo mrepo;    public List<Ticket> getAllTickets(){ return repo.findAll();  } |

|  |
| --- |
| public Ticket createTicket(Ticket tkt) { return repo.save(tkt);  }    public int setComplaint(Ticket tkt,Complaint cmp) {  try {  tkt.setComplaint(cmp);  repo.save(tkt); return 1;  }catch(Exception e) {  return 0;  }  }  public Ticket getTicket(Long Id) {  return repo.findById(Id).get();  }  public Long updateStatus(Long id, String Status) {  try {  Ticket tkt = repo.findById(id).get();  tkt.setStatus(Status); repo.save(tkt);  return 1L;  }catch(Exception e) {  return 0L;  }    }  public Long updateComment(Long id, String Comment) {  try {  Ticket tkt = repo.findById(id).get(); tkt.setComment(Comment); repo.save(tkt); return 1L;  }catch(Exception e) {  return 0L;  }    }  public Long assignEngineer(Ticket tkt) {  try {  Ticket tk = repo.findById(tkt.getTicketId()).get();  Engineer eng = erepo.findById(tkt.getEngineer().getUserid()).get(); tk.setEngineer(eng);  tk.setStatus("ASSIGNED"); repo.save(tk); return 1L;  }catch(Exception e) {  System.out.println(e); return 0L;  }  }  public List<Ticket> getEngTickets(Long engineerId){  Engineer engineer = erepo.findById(engineerId).get(); return repo.findByEngineer(engineer); |

|  |
| --- |
| }  public List<Ticket> getEngOpenTickets(Long engineerId){ Engineer engineer = erepo.findById(engineerId).get(); return repo.findByEngineerOpen(engineer);  }  public List<Ticket> getManagerTickets(Long managerId){  Manager manager = mrepo.findById(managerId).get(); return repo.findByManager(manager);  }  public List<Ticket> getManagerOpenTickets(Long managerId){  Manager manager = mrepo.findById(managerId).get(); return repo.findByManagerOpen(manager);  }  public Long assignManager(Ticket tkt) {  try {  Manager m = mrepo.findById(tkt.getManager().getUserid()).get(); Ticket t = repo.findById(tkt.getTicketId()).get();  t.setManager(m); repo.save(t); return 1L;  } catch (Exception e) {  e.printStackTrace();  return 0L;  }  }    } |
| **usersService.java**  package com.simplilearn.crs.services;    import java.util.List;    import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;    import com.simplilearn.crs.entities.Users; import com.simplilearn.crs.repository.usersRepo;    @Service  public class usersService {    @Autowired  private usersRepo repo;    public int addUser(Users usr) { repo.save(usr);  return 1;  }    public int deleteUser(long usrid) {  Users usr = repo.findById(usrid).get(); repo.delete(usr); return 1;  } |

|  |
| --- |
| public int changeStatus(String username, boolean status) { Users usr = repo.findByUsername(username); usr.setAccountStatus(status);  return 1;    }  public Users findUser(String username) {  return repo.findByUsername(username);  }  public List<Users> findAllUsers(){ return repo.findAll();  }  } |
| **Frontend – Angular – UI Project Structure:** |



|  |
| --- |
| **Addpin.component.html**  <div class="card px-3 py-3 mx-3 my-3" >  <div class="row">  <div class="col" style="float: left;display: inline;"><h1>Add Pincode</h1></div>  <div class="col"><button class="btn btn-warning" style="float: right; display: inline;" (click)="closedialog()">X</button></div>  </div>  <label for="pin">Enter the pincode to add</label>  <input type="number" maxlength="6" id="pin" name="pin" class="form-control" [(ngModel)]="pin.pin">  <button class="btn btn-primary my-2" (click)="addpin()">Add pincode</button> </div> |
| **Addpin.component.ts**  import { Component } from '@angular/core';  import { MatDialogRef } from '@angular/material/dialog'; import { pin } from 'src/app/models/pin';  import { ComplaintService } from 'src/app/services/complaint.service';    @Component({  selector: 'app-addpin', templateUrl: './addpin.component.html', styleUrls: ['./addpin.component.css']  })  export class AddpinComponent { pin:pin={  pin:0  };  constructor(  private ref:MatDialogRef<AddpinComponent>, private compservice:ComplaintService  ){}  closedialog(){  this.ref.close();  }  addpin(){  this.compservice.addPin(this.pin).subscribe(x=>{ if(x.status==1){  alert("Pincode added successfully"); this.ref.close();  }  })  }  } |
| **Adduser.component.html**  <h1 style="display: inline;" mat-dialog-title>Add user</h1>  <div mat-dialog-actions style="display:inline;margin-left: 80%;"> <button class="btn btn-danger mx-4 my-4" style="float:right;" (click)="closeDialog()">X</button>  </div>  <div mat-dialog-content>  <label id="example-radio-group-label">Select the User category:</label>  <mat-radio-group aria-labelledby="example-radio-group-label" class="example-radio-group " [(ngModel)]="usertype"> |

|  |
| --- |
| <mat-radio-button class="example-radio-button" \*ngFor="let type of usertypes"  [value]="type"  style="display:inline;">  {{type}}  </mat-radio-button>  </mat-radio-group>    <div class="card my-4">  <form class="form-group py-4 px-4" #form="ngForm">  <div class="row">  <div class="col">  <label for="firstname">Enter First Name:</label>  <input type="text" id="firstname" name="firstname" class="form-control"  [(ngModel)]="usr.firstName"  required>  </div>  <div class="col">  <label for="lastname">Enter Last Name:</label>  <input type="text" id="lastname" name="lastname" class="form-control"  [(ngModel)]="usr.lastName"  required>  </div>  <div class="col">  <label for="username">Enter Username:</label>  <input type="text" id="username" name="username" class="form-control"  [(ngModel)]="usr.username"  required>  </div>  <div class="row">  <div class="col">  <label for="password">Choose Password:</label>  <input type="password" id="password" name="password" class="form-control"  [(ngModel)]="pass"  required  [ngStyle]="{'border-  color':form.touched?(pass==repass&&repass!=='')?'green':'red':''}">  </div>  <div class="col">  <label for="repassword">Re-enter Password:</label>  <input type="password" id="repassword" name="repassword" class="form-control" [(ngModel)]="repass" required  [ngStyle]="{'border-  color':form.touched?(pass==repass&&repass!=='')?'green':'red':''}">  </div>  <div class="col">  <label for="email">Enter e-mail:</label>  <input type="email" id="e-mail" name="email" class="form-control" required [(ngModel)]="usr.email">  </div>  </div>  <div \*ngIf="usertype=='CUSTOMER'">  <div class="row">  <div class="col">  <label for="address">Enter customer address:</label>  <input type="text" id="address" name="address" class="form-control" required |

|  |
| --- |
| [(ngModel)]="cus.address">  </div>  </div>  <div class="row">  <div class="col">  <label for="pin">Select customer Pin:</label>  <select class="form-control" name="pin" id="pin" required [(ngModel)]="cus.pin"> <option>Select Pin</option>  <option \*ngFor="let pin of pins" value="{{pin.pin}}">{{pin.pin}}</option>  </select>  </div>  <div class="col">  <label for="contact">Enter customer contact No: </label>  <input type="number" id="contact" name="contact" class="form-control" [(ngModel)]="cus.contactNo" required>  </div>  </div>  </div>  <div \*ngIf="usertype=='MANAGER'">    <div class="row">  <div class="col">  <label for="pin">Select Manager service area Pin:</label>  <select class="form-control" name="pin" id="pin" required  [(ngModel)]="mngr.pin">  <option>Select Pin</option>  <option \*ngFor="let pin of pins" value="{{pin.pin}}">{{pin.pin}}</option>  </select>  </div>  <div class="col">  <label for="contact">Enter Manager contact No:</label>  <input type="number" id="contact" name="contact" class="form-control" [(ngModel)]="mngr.contactNo" required>  </div>  </div>  </div>  <div \*ngIf="usertype=='ENGINEER'">    <div class="row">  <div class="col">  <label for="pin">Add Engineer service area Pin:</label>  <select class="form-control" name="pin" id="pin" [(ngModel)]="temppin" required>  <option>Select Pin</option>  <option \*ngFor="let pin of pins" value="{{pin.pin}}">{{pin.pin}}</option>  </select><button class="btn btn-sm btn-primary" \*ngIf="temppin" (click)="addpintoengineer(temppin)">Add</button>  <br>  <ol>  Engineer Service Locations:  <li \*ngFor="let pin of engr.locations">{{pin.pin}}</li>  </ol>  </div>  <div class="col">  <label for="contact">Enter Engineer contact No:</label> |

|  |
| --- |
| <input type="number" id="contact" name="contact" class="form-control" [(ngModel)]="engr.contactNo" required>  </div>  </div>  </div>  </div>  </form>  <br><br><br>  <button class="btn btn-primary mx-4 my-4" style="width:10%;" [disabled]="!form.valid"  (click)="adduser(usertype)">Add {{usertype}}</button>    </div>  </div> |
| **Adduser.component.ts**  import { Component } from '@angular/core';  import { MatDialogRef } from '@angular/material/dialog'; import { Router } from '@angular/router'; import { Customer } from 'src/app/models/Customer'; import { Engineer } from 'src/app/models/Engineer'; import { Manager } from 'src/app/models/Manager';  import { pin } from 'src/app/models/pin'; import { user } from 'src/app/models/user';  import { ComplaintService } from 'src/app/services/complaint.service'; import { UserService } from 'src/app/services/user.service';    @Component({  selector: 'app-adduser', templateUrl: './adduser.component.html', styleUrls: ['./adduser.component.css']  })  export class AdduserComponent {  usertype:string;  usertypes:string[] = ["CUSTOMER","MANAGER","ENGINEER"]; pins:pin[];  usr:user={userid:0,firstName:"",lastName:"",username:"",password:"", roles:"", email:"", accountStatus:true };  engr:Engineer = new Engineer(this.usr); mngr:Manager = new Manager(this.usr); cus:Customer =new Customer (this.usr); temppin:number; pass:string=""; repass:string="" constructor(  private ref:MatDialogRef<AdduserComponent>,  private compservice:ComplaintService,  private usrservice:UserService, private router:Router  ){    }  ngOnInit(){  this.compservice.getAllPin().subscribe(x=>this.pins=x); this.engr.locations=[];  }  closeDialog(){ |

|  |
| --- |
| this.ref.close();  }  addpintoengineer(p:number){ this.engr.locations.push(new pin(p)); this.temppin=0;  }  adduser(type:string){ if(type=="CUSTOMER"){  let c:Customer = new Customer(this.usr); c.address=this.cus.address;  c.pin=this.cus.pin;  c.contactNo=this.cus.contactNo; if(this.pass==this.repass){ c.password=this.pass;  c.roles="CUSTOMER"; this.usrservice.addCustomer(c).subscribe(x=>{ if(x.status==1){  alert("Customer added Successfully!"); this.closeDialog();  this.router.navigateByUrl("adminpanel/viewall")  }  });  }else{  alert("Password does not match!"); this.pass="";  this.repass="";  }  this.reset();  }else if(type=="MANAGER"){  let m:Manager = new Manager(this.usr); m.pin=this.mngr.pin;  m.contactNo=this.mngr.contactNo; if(this.pass==this.repass){ m.password=this.pass;  m.roles="MANAGER"; this.usrservice.addManager(m).subscribe(x=>{ if(x.status==1){  alert("Manager added Successfully!!"); this.closeDialog();  this.router.navigateByUrl("adminpanel/viewall")  }  });  }else{  alert("Password does not match!") this.pass="";  this.repass="";  }  this.reset();  }else if(type=="ENGINEER"){  let e:Engineer = new Engineer(this.usr); e.locations=this.engr.locations;  e.contactNo=this.engr.contactNo; if(this.pass==this.repass){ e.password=this.pass;  e.roles="ENGINEER"; |

|  |
| --- |
| this.usrservice.addEngineer(e).subscribe(x=>{    if(x.status==1){  alert("Engineer added Successfully!!"); this.closeDialog();  this.router.navigateByUrl("adminpanel/viewall")  }  });  }else{  alert("Password does not match!"); this.pass='';  this.repass='';  }  this.reset();  }  }    reset(){  this.usr={userid:0,firstName:"",lastName:"",username:"",password:"", roles:"", email:"", accountStatus:true };  this.engr=new Engineer(this.usr); this.engr.locations=[]; this.mngr= new Manager(this.usr); this.cus=new Customer (this.usr); this.pass=""; this.repass="";  this.temppin=0;  }  } |
| **Adminallcomplaints.component.html**  <div class="px-3 py-3">  <table class="table table-outlined table-striped">  <thead class="table table-dark">  <tr>  <th>Complaint Id</th>  <th>Customer Name</th>  <th>Customer Address</th>  <th>Pincode</th>  <th>Customer Contact</th>  <th>Complaint Subject</th>  <th>Complaint Details</th>  <th>Feedback</th>  <th>Tickets</th>  <th>Created At</th>  </tr>  </thead>  <tbody>  <tr \*ngFor="let comp of complaints">  <td>{{comp.complaintId}}</td>  <td>{{comp.name}}</td>  <td>{{comp.address}}</td>  <td>{{comp.pin.pin}}</td>  <td>{{comp.contactNo}}</td>  <td>{{comp.subject}}</td>  <td>{{comp.details}}</td> |

|  |
| --- |
| <td>{{comp.feedback?comp.feedback:"Not given"}}</td>  <td>  <ul \*ngFor="let tkt of comp.ticket" class="list-group list-group-horizontal-sm">  <li class="list-group-item">{{tkt.ticketId}}</li>  <li class="list-group-item"><b>{{tkt.status}}</b></li>  <li class="list-group-item"><button class="btn btn-warning btn-sm"  (click)="viewticket(tkt)"><i class="bi bi-eye-fill"></i></button></li>  </ul>  </td>  <td>{{comp.createdAt | date:"dd-MMM-yy hh:mm"}}</td>  </tr>  </tbody>  </table>  </div> |
| **Adminallcomplaints.component.ts** import { Component } from '@angular/core'; import { MatDialog } from '@angular/material/dialog';  import { ViewticketComponent } from 'src/app/customer/viewticket/viewticket.component'; import { complaint } from 'src/app/models/Complaint'; import { Ticket } from 'src/app/models/Ticket';  import { ComplaintService } from 'src/app/services/complaint.service';    @Component({  selector: 'app-adminallcomplaints', templateUrl: './adminallcomplaints.component.html', styleUrls: ['./adminallcomplaints.component.css']  })  export class AdminallcomplaintsComponent {  complaints:complaint[]; constructor(  private compservice:ComplaintService, private dialog:MatDialog  ){}  ngOnInit(){  this.compservice.getAllComplaints().subscribe(x=>this.complaints=x);  };      viewticket(tkt:Ticket){  this.dialog.open(ViewticketComponent,{width:'30%',data:tkt})  }  } |
| **Adminalltickets.component.html**  <table class="table table-striped table-outlined">  <thead class="table-dark">  <tr>  <th class="text-center">TicketID</th>  <th class="text-center">Status</th>  <th class="text-center">Date of creation</th>  <th class="text-center" >Last Updated</th>  <th class="text-center">Area Manager</th>  <th class="text-center">Complaint Details</th>  <th class="text-center">Assigned Engineer</th>  </tr>  </thead> |

|  |
| --- |
| <tbody>    <tr \*ngFor="let ticket of tickets">    <td class="text-center">{{ticket.ticketId}}</td>  <td [ngStyle]="{'background-color': (ticket.status=='CLOSED')?  '#00db3a':(ticket.status=='RAISED')? '#00b3ff':(ticket.status=='ASSIGNED')?'#f5c105':'#d9d1e3' }" class="text-center" style="color: white;"> {{ticket.status}} </td>  <td class="text-center">{{ticket.createdAt | date :"dd-MMM-yyyy,hh:mm:ss"}}</td>  <td class="text-center">{{ticket.updatedAt | date :"dd-MMM-yyyy,hh:mm:ss"}}</td> <td class="text-center"><button \*ngIf="ticket.manager==null" class='btn btn-warning' (click)='assignmanager(ticket)'>Assign Manager</button>{{(ticket.manager!==null)? ticket.manager.firstName:'' + " " }}{{ (ticket.manager!==null)? ticket.manager.lastName : '' }}</td>  <td class="text-center"><button class="btn btn-warning btn-sm"  (click)="complaintdetails(ticket)"><i class="bi bi-eye-fill"></i></button></td>  <td class="text-center"><button \*ngIf="ticket.engineer==null" class='btn btn-warning' (click)='assignengineer(ticket)'>Assign  Engineer</button>{{(ticket.engineer!==null)?ticket.engineer.firstName:"" + " " }}{{ (ticket.engineer!==null)? ticket.engineer.lastName :" " }}</td>    </tr>    </tbody>  </table> |
| **Adminalltickets.component.ts**  import { Component } from '@angular/core'; import { MatDialog } from '@angular/material/dialog'; import { ComplaintdetailsComponent } from  'src/app/manager/complaintdetails/complaintdetails.component';  import { Ticket } from 'src/app/models/Ticket';  import { TicketService } from 'src/app/services/ticket.service';  import { AssignmanagerComponent } from '../assignmanager/assignmanager.component'; import { AssignengineerComponent } from 'src/app/assignengineer/assignengineer.component';    @Component({  selector: 'app-adminalltickets', templateUrl: './adminalltickets.component.html', styleUrls: ['./adminalltickets.component.css']  })  export class AdminallticketsComponent { tickets:Ticket[]; constructor(  private tktService:TicketService, private dialog:MatDialog  ){  }  ngOnInit(){  this.tktService.getAllTickets().subscribe(x=>{this.tickets=x;});    }  complaintdetails(tkt:Ticket){  this.dialog.open(ComplaintdetailsComponent,{width:'40%',data:tkt.ticketId});  }  assignmanager(tkt:Ticket){ |

|  |
| --- |
| this.dialog.open(AssignmanagerComponent,{width:'40%',data:tkt});  }  assignengineer(tkt:Ticket){  this.dialog.open(AssignengineerComponent,{width:'40%',data:tkt});  }  } |
| **Allusers.component.html**  <button class="btn btn-primary" style="margin-left: 90%;" (click)="adduser()">Add User</button> <table class="table table-striped table-outlined">  <thead class="table-dark">  <tr>  <th>UserId</th>  <th>Full Name</th>  <th>Username</th>  <th>Email</th>  <th>Roles</th>  <th>Account Status</th>  </tr>  </thead>  <tbody>  <tr \*ngFor="let user of usrs">  <td>{{user.userid}}</td>  <td>{{user.firstName+' '+user.lastName}}</td>  <td>{{user.username}}</td>  <td>{{user.email}}</td>  <td>  {{user.roles}}  </td>  <td>{{user.accountStatus}}</td>  </tr>  </tbody>  </table> |
| **Allusers.component.ts**  import { Component } from '@angular/core'; import { MatDialog } from '@angular/material/dialog';  import { AdduserComponent } from '../adduser/adduser.component'; import { user } from 'src/app/models/user';  import { UserService } from 'src/app/services/user.service';    @Component({  selector: 'app-allusers', templateUrl: './allusers.component.html', styleUrls: ['./allusers.component.css']  })  export class AllusersComponent { usrs:user[];  constructor(private dialog:MatDialog, private usrService:UserService){    }  ngOnInit(){  this.usrService.getAllUsers().subscribe(x=>this.usrs=x);  }  adduser(){  this.dialog.open(AdduserComponent,{width:'80%',height:'75%'})  } |

|  |
| --- |
| } |
| **Assignmanager.component.html**  <div class="card px-3 py-3 mx-3 my-3">  <div class="row">  <div class="col"><h1 style="float: left;">Assign Manager</h1></div>  <div class="col"><button style="float: right;" (click)="closedialog()" class="btn btnwarning">X</button></div>  </div>  <table class="table table-outlined">  <tbody>  <tr>  <th>TicketID</th>  <td>{{data.ticketId}}</td>  </tr>  <tr>  <th>Pincode</th>  <td>{{comp.pin.pin}}</td>  </tr>  <tr>  <th>Select Manager</th>  <td>  <select class="form-control" name="manager" id="manager" #mgr>  <option \*ngFor="let mgr of managers" value={{mgr.userid}}>{{mgr.firstName+' '+mgr.lastName}} </option>    </select>  <button class="btn btn-warning" (click)="assignmanager(mgr.value)">Assign</button>  </td>  </tr>  </tbody>  </table>  </div> |
| **Assignmanager.component.ts**  import { Component ,Inject} from '@angular/core';  import { MAT\_DIALOG\_DATA, MatDialogRef } from '@angular/material/dialog'; import { complaint } from 'src/app/models/Complaint'; import { Manager } from 'src/app/models/Manager'; import { Ticket } from 'src/app/models/Ticket'; import { user } from 'src/app/models/user';  import { ComplaintService } from 'src/app/services/complaint.service'; import { TicketService } from 'src/app/services/ticket.service'; import { UserService } from 'src/app/services/user.service';    @Component({  selector: 'app-assignmanager', templateUrl: './assignmanager.component.html', styleUrls: ['./assignmanager.component.css']  })  export class AssignmanagerComponent { managers:Manager[]; comp:complaint; constructor(  @Inject(MAT\_DIALOG\_DATA) public data:Ticket, private ref:MatDialogRef<AssignmanagerComponent>, |

|  |
| --- |
| private userservice:UserService,  private tktservice:TicketService  ){}  ngOnInit(){  console.log(this.data);  this.userservice.getAllManagers().subscribe(x=>this.managers=x);  this.tktservice.getComplaintdataByTicketid(this.data.ticketId).subscribe(x=>this.comp=x);  }  closedialog(){  this.ref.close();  }    assignmanager(managerId:any){ let mgr:Manager = { userid:Number(managerId),  firstName:"", lastName:"", username:"", password:"", email:"",  roles:"",  accountStatus:true, contactNo:0, pin:{  pin:0  }  }  this.data.manager=mgr;  this.tktservice.assignmanager(this.data).subscribe(x=>{ if(x.status==1){  alert("Manager Assigned");  this.ref.close();  }  });  }  } |
| **Pincodedetails.component.html**  <div class="px-3 py-3 mx-3 my-3">  <div class="pin" style="width: 30%;display: inline;float: left;">  <button class="btn btn-primary my-3" (click)="addpin()">Add Pin</button> <table class="table table-outlined table-bordered"> <thead>  <tr>  <th >Available Pincodes</th>    </tr>  </thead>  <tbody>  <tr \*ngFor="let pin of pins">  <td>{{pin.pin}}</td>  </tr>  </tbody>  </table>  </div>  <div class="manager" style="width: 30%; float: right;"> |

|  |
| --- |
| <table class="table table-bordered">  <thead>  <tr>  <th colspan="2">List of Managers</th>  </tr>  </thead>  <tbody>  <tr \*ngFor="let mgr of managers">  <td>{{mgr.pin.pin}}</td>  <td>{{mgr.firstName+' '+mgr.lastName}}</td>  </tr>  </tbody>  </table> </div>  <div class="engineer" style="width: 30%; display: inline-block;margin-left: 4rem;">    <table class="table table-bordered">  <thead>  <tr>  <th colspan="2">List of Engineers</th>  </tr>  </thead>  <tbody>  <tr \*ngFor="let egr of engineers">  <td>{{egr.firstName+' '+egr.lastName}}</td>  <td><ul class="list-group">  <li \*ngFor="let pin of egr.locations" class="list-group-item">{{pin.pin}}</li>  </ul>  </td>  </tr>  </tbody>  </table>  </div>  </div> |
| **Pincodedetails.component.ts**  import { Component } from '@angular/core'; import { MatDialog } from '@angular/material/dialog'; import { Engineer } from 'src/app/models/Engineer'; import { Manager } from 'src/app/models/Manager';  import { pin } from 'src/app/models/pin';  import { ComplaintService } from 'src/app/services/complaint.service'; import { TicketService } from 'src/app/services/ticket.service'; import { UserService } from 'src/app/services/user.service'; import { AddpinComponent } from '../addpin/addpin.component';      @Component({  selector: 'app-pincodedetails', templateUrl: './pincodedetails.component.html', styleUrls: ['./pincodedetails.component.css']  })  export class PincodedetailsComponent { pins:pin[];  engineers:Engineer[]; |

|  |
| --- |
| managers:Manager[];    constructor( private userservice:UserService, private ticketservice:TicketService, private compservice:ComplaintService, private dialog:MatDialog  ){}    ngOnInit(){ this.userservice.getAllEngineers().subscribe(x=>this.engineers=x); this.userservice.getAllManagers().subscribe(x=>this.managers=x); this.compservice.getAllPin().subscribe(x=>this.pins=x);  }  addpin(){  this.dialog.open(AddpinComponent,{width:'40%',height:'40%'})  }  } |
| **Adminpanel.component.html**  <div class="card">  <h1 >Admin Dashboard</h1>  <div class="row mx-3 my-4">  <button class="btn btn-primary mx-2" routerLink="viewall" style="width: 10%;">View All users</button>  <button class="btn btn-primary " style="width: 10%;" routerLink="allcomplaints">View All Complaints</button>  <button class="btn btn-primary mx-2" style="width: 10%;" routerLink="alltickets">View All Tickets</button>  <button class="btn btn-primary " style="width: 20%;" routerLink="pincodedetails">Pincode, Area manager and Engieers</button>  </div>  </div>  <div class="card px-3 py-3">  <router-outlet></router-outlet> </div> |
| **Adminpanel.component.ts**  import { Component } from '@angular/core';    @Component({  selector: 'app-adminpanel', templateUrl: './adminpanel.component.html', styleUrls: ['./adminpanel.component.css']  })  export class AdminpanelComponent {    } |
| **Assignengineer.component.html**  <div class="card px-3 py-3 mx-3 my-3">  <div class="row">  <div class="col"><h1 style="float: left;">Assign Manager</h1></div>  <div class="col"><button style="float: right;" (click)="closedialog()" class="btn btnwarning">X</button></div>  </div>  <table class="table table-outlined"> <tbody> |

|  |
| --- |
| <tr>  <th>TicketID</th>  <td>{{data.ticketId}}</td>  </tr>  <tr>  <th>Pincode</th>  <td>{{comp.pin.pin}}</td>  </tr>  <tr>  <th>Select Engineer</th>  <td>  <select class="form-control" name="manager" id="manager" #egr>  <option \*ngFor="let egr of engineers" value={{egr.userid}}>{{egr.firstName+' '+egr.lastName}} </option>    </select>  <button class="btn btn-warning" (click)="assignEngineer(egr.value)">Assign</button>  </td>  </tr>  </tbody>  </table>  </div> |
| **Assignengineer.component.ts**  import { Component,Inject } from '@angular/core';  import { MAT\_DIALOG\_DATA, MatDialogRef } from '@angular/material/dialog'; import { Ticket } from '../models/Ticket'; import { complaint } from '../models/Complaint'; import { Engineer } from '../models/Engineer';  import { TicketService } from '../services/ticket.service';    @Component({  selector: 'app-assignengineer', templateUrl: './assignengineer.component.html', styleUrls: ['./assignengineer.component.css']  })  export class AssignengineerComponent { comp:complaint;  engineers:Engineer[];    constructor(  @Inject(MAT\_DIALOG\_DATA) public data:Ticket, private ref:MatDialogRef<AssignengineerComponent>, private ticketservice:TicketService  ){}  ngOnInit(){  this.ticketservice.getComplaintdataByTicketid(this.data.ticketId).subscribe(x=>{ this.comp=x;  this.ticketservice.getEngineerForPin(this.comp.pin.pin).subscribe(x=>{ this.engineers=x;  });  });  }  closedialog(){  this.ref.close();  } |

|  |
| --- |
| assignEngineer(eid:string){ let eng:Engineer={ userid:Number(eid),  firstName:"", lastName:"", username:"", password:"", email:"",  roles:"",  accountStatus:true, contactNo:0,  locations:[],  }  this.data.engineer=eng;  this.ticketservice.assignEngineer(this.data).subscribe(x=>{ if(x.status==1){  alert("Engineer Assigned");  this.closedialog();  }  });  }  } |
| **Allcomplaint.component.html**  <div class="px-3 py-3">  <table class="table table-outlined table-striped">  <thead class="table table-dark">  <tr>  <th>Complaint Id</th>  <th>Customer Name</th>  <th>Customer Address</th>  <th>Pincode</th>  <th>Customer Contact</th>  <th>Complaint Subject</th>  <th>Complaint Details</th>  <th>Feedback</th>  <th>Tickets</th>  <th>Created At</th>  </tr>  </thead>  <tbody>  <tr \*ngFor="let comp of complaints">  <td>{{comp.complaintId}}</td>  <td>{{comp.name}}</td>  <td>{{comp.address}}</td>  <td>{{comp.pin.pin}}</td>  <td>{{comp.contactNo}}</td>  <td>{{comp.subject}}</td> <td>{{comp.details}}</td>  <td>{{comp.feedback?comp.feedback:"Not given"}}<button \*ngIf="comp.feedback==null && checkTicketClosed(comp)" class="btn btn-warning" (click)="givefeedback(comp)"><i class="bi bi-pencil-square"></i></button></td>  <td>  <ul \*ngFor="let tkt of comp.ticket" class="list-group list-group-horizontal-sm">  <li class="list-group-item">{{tkt.ticketId}}</li>  <li class="list-group-item"><b>{{tkt.status}}</b></li> |

|  |
| --- |
| <li class="list-group-item"><button class="btn btn-warning btn-sm"  (click)="viewticket(tkt)"><i class="bi bi-eye-fill"></i></button></li>    </ul>  <button \*ngIf="checkTicketClosed(comp)" class="btn btn-warning"  (click)="raiseticket(comp)">Raise Ticket</button>  </td>  <td>{{comp.createdAt | date:"dd-MMM-yy hh:mm"}}</td>  </tr>  </tbody>  </table>  </div> |
| **Allcomplaint.component.ts**  import { Component } from '@angular/core'; import { MatDialog } from '@angular/material/dialog'; import { complaint } from 'src/app/models/Complaint'; import { Customer } from 'src/app/models/Customer';  import { Ticket } from 'src/app/models/Ticket';  import { AuthService } from 'src/app/services/auth.service'; import { ComplaintService } from 'src/app/services/complaint.service'; import { ViewticketComponent } from '../viewticket/viewticket.component'; import { FeedbackComponent } from '../feedback/feedback.component'; import { TicketraiseComponent } from '../ticketraise/ticketraise.component';    @Component({  selector: 'app-allcomplaint', templateUrl: './allcomplaint.component.html', styleUrls: ['./allcomplaint.component.css']  })  export class AllcomplaintComponent {  complaints: complaint[]; constructor(  private compservice: ComplaintService, private authservice: AuthService, private dialog: MatDialog  ) { }  ngOnInit() {  if (this.authservice.currentUser.roles.includes('CUSTOMER')) { this.authservice.getCurrentUser().subscribe(x => {  this.compservice.getCustomerComplaints(x.userid).subscribe(x => { this.complaints = x; console.log(x) });  });  } else {  alert("Bad Request!")  }  }  viewticket(tkt: Ticket) {  this.dialog.open(ViewticketComponent, { width: '30%', data: tkt })  }  givefeedback(c: complaint) {  this.dialog.open(FeedbackComponent, { data: c });  }    checkTicketClosed(c: complaint): boolean { |

|  |
| --- |
| let a = c.ticket.filter(x => {    if (x.status == "CLOSED" || x.status == "ESCALATED") {    return true;  } else {    return false;  }  });    if (a.length > 0) {    return true;  } else {    return false;  }  }  raiseticket(c: complaint) {  this.compservice.ticketReRaise(c).subscribe(x => { if (x.status == 1) {  this.dialog.open(TicketraiseComponent, { data: x });  } else {  alert("Server error!")  }  });  }  } |
| **Complaintid.component.html**  <div class="card px-3 py-3 mx-3 my-3">    <div class="row">  <div class="col"><h1 style="float: left;display: inline;">Complaint Registered</h1></div>  <div class="col"><button class="btn btn-danger" style="float: right;"  (click)="closedialog()">X</button></div>  </div>  <br>  <br>  <div>  <p>Your complaint is successfully registered.</p>  <p>Complaint reference number:ABCCR000{{data.complaintId}}</p>  <p>Ticket Id:{{data.ticketId}} is created against your complaint. The concerned area manager will shortly assign an engineer to resolve your issue. Keep track of the ticket status from customer dashboard.</p>  </div>  </div> |
| **Complaintid.component.ts**  import { Component, Inject } from '@angular/core';  import { MAT\_DIALOG\_DATA, MatDialogRef } from '@angular/material/dialog';    @Component({  selector: 'app-complaintid',  templateUrl: './complaintid.component.html', |

|  |
| --- |
| styleUrls: ['./complaintid.component.css']  })  export class ComplaintidComponent { constructor(  @Inject(MAT\_DIALOG\_DATA) public data: any, private ref:MatDialogRef<ComplaintidComponent>  ){}    closedialog(){  this.ref.close();  }  } |
| **Createcomplaint.component.html**  <div class="card mx-4 my-4 px-4 py-4">  <div>  <div style="display: inline;float: left;">  <h1>Raise complaint</h1>  </div>  <button class="btn btn-sm btn-danger" style="float: right;" (click)="closedialog()">X</button> </div>    <form (ngSubmit)="createcomplaint()">  <div class="row">  <div class="col">  <label for="cusname">Enter your name</label>  <input type="text" id="cusname" name="cusname" class="form-control"  [(ngModel)]="complaint.name" required>  </div>  <div class="col">  <label for="address">Enter your address</label>  <input type="text" id="address" name="address" class="form-control"  [(ngModel)]="complaint.address" required>  </div>  </div>  <div class="row">  <div class="col">  <label for="pin">Select Pin</label>  <select class="form-control" name="pin" id="pin" required  [(ngModel)]="complaint.pin.pin" required>  <option>Select Pin</option>  <option \*ngFor="let pin of pins" value="{{pin.pin}}">{{pin.pin}}</option>  </select>  </div>  <div class="col">  <label for="contact">Enter your contact No </label>  <input type="number" id="contact" name="contact" class="form-control" required [(ngModel)]="complaint.contactNo" required>  </div>  </div>  <br><br>  <div class="row">  <div class="col">  <label for="subject">Select Subject</label>  <select class="form-control" name="subject" id="subject" required [(ngModel)]="complaint.subject"> |

|  |
| --- |
| <option>Select complaint subject</option>  <option \*ngFor="let sub of subjects" value="{{sub}}">{{sub}}</option>  </select>  </div>  <div class="col">  <label for="details">Enter complaint details</label>  <input type="text" id="details" name="details" class="form-control"  [(ngModel)]="complaint.details" required>  </div>  </div>  <br><br>  <button class="btn btn-warning" type="submit">Submit</button> </form>  </div> |
| **Createcomplaint.component.ts**  import { Component } from '@angular/core';  import { MatDialog, MatDialogRef } from '@angular/material/dialog'; import { complaint } from 'src/app/models/Complaint'; import { Customer } from 'src/app/models/Customer';  import { pin } from 'src/app/models/pin';  import { AuthService } from 'src/app/services/auth.service';  import { ComplaintService } from 'src/app/services/complaint.service';  import { ComplaintidComponent } from '../complaintid/complaintid.component';    @Component({  selector: 'app-createcomplaint', templateUrl: './createcomplaint.component.html', styleUrls: ['./createcomplaint.component.css']  })  export class CreatecomplaintComponent { complaint:complaint = new complaint();  subjects:string[]= ["Cannot make a call but receive a call" , "Can make calls, but cannot receive calls","Cannot make or receive calls"]; pins:pin[]; cus:Customer; constructor(  private ref:MatDialogRef<CreatecomplaintComponent>,  private compservice:ComplaintService, private authservice:AuthService, private dialog:MatDialog  ){}  ngOnInit(){  this.compservice.getAllPin().subscribe(x=>this.pins=x);  this.authservice.getCurrentUser().subscribe(x=>{this.cus=new Customer(x)}); this.complaint.pin=new pin(0);  }  closedialog(){  this.ref.close();  }    createcomplaint(){ if(this.cus){  this.complaint.customer=this.cus;  this.compservice.raiseComplaint(this.complaint).subscribe(x=>{ if(x.status==1){ |

|  |
| --- |
| this.dialog.open(ComplaintidComponent,{data:x});  } });  this.ref.close();  }  }  } |
| **Feedback.component.html**  <div class="card px-3 py-3 mx-3 my-3">  <div>  <div style="display: inline;float: left;">  <h1>Give Feedback</h1>  </div>  <button class="btn btn-sm btn-danger" style="float: right;" (click)="closedialog()">X</button>  </div>  <div class="my-3">      <input type="text" name="feedback" id="feedback" class="form-control my-2" [(ngModel)]="feedback">  <button class="btn btn-primary" (click)="givefeedback()">Submit</button>  </div>  </div> |
| **Feedback.component.ts**  import { Component,Inject } from '@angular/core';  import { MAT\_DIALOG\_DATA, MatDialogRef } from '@angular/material/dialog'; import { complaint } from 'src/app/models/Complaint'; import { Ticket } from 'src/app/models/Ticket';  import { ComplaintService } from 'src/app/services/complaint.service'; import { TicketService } from 'src/app/services/ticket.service';    @Component({  selector: 'app-feedback', templateUrl: './feedback.component.html', styleUrls: ['./feedback.component.css']  })  export class FeedbackComponent {  feedback:string=""; constructor(  @Inject(MAT\_DIALOG\_DATA) private data:complaint, private ref:MatDialogRef<FeedbackComponent>, private compservice:ComplaintService){}    closedialog(){  this.ref.close();  }  givefeedback(){  this.data.feedback=this.feedback;  this.compservice.postFeedback(this.data).subscribe(x=>{ if(x.status==1){  alert("Feedback successfully posted!!");  } });  this.closedialog();  } |

|  |
| --- |
| } |
| **Ticketraise.component.html**  <div class="card px-5 py-5">  <p class="text-lg">Ticket successfully raised again the complaint! <br>  Complaint Reference Number:ABCCR000{{data.complaintId}} <br> Raised Ticket Number:{{data.ticketId}}  </p>  </div> |
| **Ticketraise.component.ts**  import { Component,Inject } from '@angular/core';  import { MAT\_DIALOG\_DATA } from '@angular/material/dialog';    @Component({  selector: 'app-ticketraise', templateUrl: './ticketraise.component.html', styleUrls: ['./ticketraise.component.css']  })  export class TicketraiseComponent {  constructor(@Inject(MAT\_DIALOG\_DATA) public data:any){}  } |
| **Viewticket.component.html**  <div class="card px-3 py-3 mx-3 my-3">  <div class="row">  <div class="col">  <h1 style="float: left;display: inline;">Ticket Details</h1> </div>  <div class="col">  <button class="btn btn-warning" style="float: right;" (click)="closedialog()">X</button>  </div>  </div>  <table class="table table-bordered">  <tbody>  <tr>  <th>Ticket ID</th>  <td>{{ticket.ticketId}}</td>  </tr>  <tr>  <th>Status</th>  <td> <button disabled class="btn" [ngStyle]="{'background-  color':(ticket.status=='RAISED')?'#F29727':(ticket.status=='CLOSED')?'#22A699':'#B3C890'}" style="border: none;color: black;">{{ticket.status}}</button></td>  </tr>  <tr>  <th>Area Manager</th>  <td>{{ticket.manager==null?'Not Present':ticket.manager.firstName}} {{' '}}{{ ticket.manager==null?'':ticket.manager.lastName}}</td>  </tr>  <tr>  <th>Assigned Engineer</th>  <td>{{ticket.engineer==null?'Not Assigned':ticket.engineer.firstName }}{{' '}}{{ ticket.engineer==null?'':ticket.engineer.lastName}}</td>  </tr>  <tr> |

|  |
| --- |
| <th>Ticket Created at</th>  <td>{{ticket.createdAt | date:"dd-MMM-yy hh:mm"}}</td>  </tr>  <tr>  <th>Ticket last updated at</th>  <td>{{ticket.updatedAt | date: "dd-MMM-yy hh:mm"}}</td>  </tr>  </tbody>  </table>  </div> |
| **Viewticket.component.ts**  import { Component, Inject } from '@angular/core';  import { MAT\_DIALOG\_DATA, MatDialogRef } from '@angular/material/dialog'; import { Ticket } from 'src/app/models/Ticket';    @Component({  selector: 'app-viewticket', templateUrl: './viewticket.component.html', styleUrls: ['./viewticket.component.css']  })  export class ViewticketComponent { constructor(  @Inject(MAT\_DIALOG\_DATA) public ticket: Ticket, private ref:MatDialogRef<ViewticketComponent>  ){}  closedialog(){  this.ref.close();  }  } |
| **Customer.component.html**  <div class="card">  <div class="row">  <div class="col px-4 py-3">  <span>  <h1>Customer Dashboard</h1>  </span>  </div>  <div class="col px-3 py-3">  <button class="btn btn-primary mx-2" routerLink="createcomplaint" style="float: right;"  (click)="createcomplaint()">Create a  complaint</button>  </div>  </div>      <div class="card px-4 py-3">  <router-outlet></router-outlet>  </div>  </div> |
| **Customer.component.ts**  import { Component } from '@angular/core'; import { MatDialog } from '@angular/material/dialog';  import { CreatecomplaintComponent } from './createcomplaint/createcomplaint.component';    @Component({ |

|  |
| --- |
| selector: 'app-customer', templateUrl: './customer.component.html', styleUrls: ['./customer.component.css']  })  export class CustomerComponent {    constructor(  private dialog:MatDialog,  ){}  createcomplaint(){  this.dialog.open(CreatecomplaintComponent,{width:'70%'});  }  } |
| **Alltickets.component.html**  <table class="table table-striped table-outlined">  <thead class="table-dark">  <tr>  <th class="text-center">TicketID</th>  <th class="text-center">Status</th>  <th class="text-center">Date of creation</th>  <th class="text-center" >Last Updated</th>  <th class="text-center">Area Manager</th>  <th class="text-center">Complaint Details</th>  <th class="text-center">Assigned Engineer</th>  </tr>  </thead>  <tbody>    <tr \*ngFor="let ticket of tickets">    <td class="text-center">{{ticket.ticketId}}</td>  <td [ngStyle]="{'background-color': (ticket.status=='CLOSED')?  '#00db3a':(ticket.status=='RAISED')? '#00b3ff':(ticket.status=='ASSIGNED')?'#f5c105':'#d9d1e3' }" class="text-center" style="color: white;"> {{ticket.status}} </td>  <td class="text-center">{{ticket.createdAt | date :"dd-MMM-yyyy,hh:mm:ss"}}</td>  <td class="text-center">{{ticket.updatedAt | date :"dd-MMM-yyyy,hh:mm:ss"}}</td>  <td class="text-center">{{(ticket.manager!==null)? ticket.manager.firstName:"Not  Assigned" + " " }}{{ (ticket.manager!==null)? ticket.manager.lastName : " " }}</td>  <td class="text-center"><button class="btn btn-warning btn-sm"  (click)="complaintdetails(ticket)"><i class="bi bi-eye-fill"></i></button></td>  <td class="text-center">{{(ticket.engineer!==null)?ticket.engineer.firstName:"Not  Assigned" + " " }}{{ (ticket.engineer!==null)? ticket.engineer.lastName :" " }}</td>    </tr>    </tbody>  </table> |
| **Alltickets.component.ts**  import { Component } from '@angular/core'; import { MatDialog } from '@angular/material/dialog'; import { ComplaintdetailsComponent } from  'src/app/manager/complaintdetails/complaintdetails.component'; import { Engineer } from 'src/app/models/Engineer'; import { Ticket } from 'src/app/models/Ticket';  import { AuthService } from 'src/app/services/auth.service'; |

|  |
| --- |
| import { TicketService } from 'src/app/services/ticket.service';    @Component({  selector: 'app-alltickets', templateUrl: './alltickets.component.html', styleUrls: ['./alltickets.component.css']  })  export class AllticketsComponent {  tickets:Ticket[] constructor( private tktservice:TicketService, private authservice:AuthService, private dialog:MatDialog  ){}  ngOnInit(){  this.authservice.getCurrentUser().subscribe(x=>{  this.tktservice.getAllEngineerTickets(x.userid).subscribe(x=>{ this.tickets=x;  })  });    }  complaintdetails(tkt:Ticket){  this.dialog.open(ComplaintdetailsComponent,{width:'40%',data:tkt.ticketId});  }  } |
| **Opentickets.component.html**  <table class="table table-striped table-outlined">  <thead class="table-dark">  <tr>  <th class="text-center">TicketID</th>  <th class="text-center">Status</th>  <th class="text-center">Date of creation</th>  <th class="text-center" >Last Updated</th>  <th class="text-center">Area Manager</th>  <th class="text-center">Complaint Details</th>  <th class="text-center">Assigned Engineer</th>  </tr>  </thead>  <tbody>    <tr \*ngFor="let ticket of tickets">    <td class="text-center">{{ticket.ticketId}}</td>  <td [ngStyle]="{'background-color': (ticket.status=='CLOSED')?  '#00db3a':(ticket.status=='RAISED')? '#00b3ff':(ticket.status=='ASSIGNED')?'#f5c105':'#d9d1e3' }" class="text-center" style="color: white;"> {{ticket.status}} <button  \*ngIf="ticket.status!=='CLOSED'" (click)="updatestatus(ticket)" class="btn btn-danger"><i class="bi bi-pencil-square"></i></button></td>  <td class="text-center">{{ticket.createdAt | date :"dd-MMM-yyyy,hh:mm:ss"}}</td>  <td class="text-center">{{ticket.updatedAt | date :"dd-MMM-yyyy,hh:mm:ss"}}</td>  <td class="text-center">{{(ticket.manager!==null)? ticket.manager.firstName:"Not  Assigned" + " " }}{{ (ticket.manager!==null)? ticket.manager.lastName : " " }}</td>  <td class="text-center"><button class="btn btn-warning btn-sm"  (click)="complaintdetails(ticket)"><i class="bi bi-eye-fill"></i></button></td> |

|  |
| --- |
| <td class="text-center">{{(ticket.engineer!==null)?ticket.engineer.firstName:"Not  Assigned" + " " }}{{ (ticket.engineer!==null)? ticket.engineer.lastName :" " }}</td>    </tr>    </tbody>  </table> |
| **Opentickets.component.ts**  import { Component } from '@angular/core'; import { MatDialog } from '@angular/material/dialog'; import { ComplaintdetailsComponent } from  'src/app/manager/complaintdetails/complaintdetails.component';  import { Ticket } from 'src/app/models/Ticket'; import { AuthService } from 'src/app/services/auth.service'; import { TicketService } from 'src/app/services/ticket.service';  import { UpdatestatusComponent } from '../updatestatus/updatestatus.component';    @Component({  selector: 'app-opentickets', templateUrl: './opentickets.component.html', styleUrls: ['./opentickets.component.css']  })  export class OpenticketsComponent { tickets:Ticket[] constructor( private tktservice:TicketService, private authservice:AuthService, private dialog:MatDialog  ){}  ngOnInit(){  this.authservice.getCurrentUser().subscribe(x=>{  this.tktservice.getEngineerOpenTickets(x.userid).subscribe(x=>{ this.tickets=x;  })  });    }  complaintdetails(tkt:Ticket){  this.dialog.open(ComplaintdetailsComponent,{width:'40%',data:tkt.ticketId});  }  updatestatus(tkt:Ticket){  this.dialog.open(UpdatestatusComponent,{width:'20%',data:tkt});  }  } |
| **Updatestatus.component.html**  <div class="card mx-3 my-3 px-3 py-3">  <div class="row">  <div class="col"><h1 style="float: left;">Update Status</h1></div>  <div class="col"><button class="btn btn-danger" style="float: right;"  (click)="closedialog()">X</button></div> </div>  <label for="status">Select Status</label>  <select name="status" id="status" class="form-control" [(ngModel)]="tempstatus">  <option value="WIP">Work In Progress(WIP)</option>  <option value="ESCALATED">ESCALATED</option> |

|  |
| --- |
| <option value="CLOSED">CLOSED</option>  </select>  <button class="btn btn-primary my-3" (click)="updatestatus()">Update</button> </div> |
| **Updatestatus.component.ts**  import { Component,Inject } from '@angular/core';  import { MAT\_DIALOG\_DATA, MatDialogRef } from '@angular/material/dialog'; import { Ticket } from 'src/app/models/Ticket';  import { TicketService } from 'src/app/services/ticket.service';    @Component({  selector: 'app-updatestatus', templateUrl: './updatestatus.component.html', styleUrls: ['./updatestatus.component.css']  })  export class UpdatestatusComponent {  tempstatus:string=""; constructor(  @Inject(MAT\_DIALOG\_DATA)private data:Ticket, private ref:MatDialogRef<UpdatestatusComponent>, private tktservice:TicketService){}    updatestatus(){  this.data.status=this.tempstatus;  this.tktservice.updateStatus(this.data).subscribe(x=>{ if(x.status==1){  alert("Status updated!!");  } });  this.closedialog();  }    closedialog(){  this.ref.close();  }  } |
| **Engineer.component.html**  <div class="card" >  <h1 >Engineer Dashboard</h1>  <div class="row">  <button class="btn btn-primary mx-3" style="width:10%;float:left;display: inline;" routerLink="alltickets">View All Tickets</button>  <button class="btn btn-primary mx-3" style="width:10%;float:left;display: inline;" routerLink="opentickets">View Open Tickets</button> </div>  </div>  <div class="card my-4">  <router-outlet></router-outlet> </div> |
| **Engineer.component.ts**  import { Component } from '@angular/core';    @Component({  selector: 'app-engineer',  templateUrl: './engineer.component.html', |

|  |
| --- |
| styleUrls: ['./engineer.component.css']  })  export class EngineerComponent {    } |
| **Login.gaurd.ts** import { CanActivateFn, Router } from '@angular/router'; import { AuthService } from '../services/auth.service'; import { inject } from '@angular/core';    export const loginGuard: CanActivateFn = (route, state) => { const authservice = inject(AuthService); const router = inject(Router); if(authservice.currentUser.authentication){ console.log("Login status checking.......true"); return true;  }else{  console.log("Login status checking.......false"); router.navigateByUrl("login");  return false;  }  }; |
| **Landing.component.html**  <app-navbar></app-navbar>  <router-outlet></router-outlet> |
| **Landing.component.ts**  import { Component } from '@angular/core'; import { AuthService } from '../services/auth.service'; import { user } from '../models/user'; import { Router } from '@angular/router';    @Component({  selector: 'app-landing', templateUrl: './landing.component.html', styleUrls: ['./landing.component.css']  })  export class LandingComponent { usr:any= { authentication:false,  username:"guest",  roles:[]  };  constructor(private auth:AuthService,  private router:Router){} ngOnInit(){  this.usr = this.auth.currentUser; if(this.usr.roles.includes("MANAGER")){ this.router.navigateByUrl('manager');  }else if(this.usr.roles.includes("CUSTOMER")){ this.router.navigateByUrl('customer'); }else if(this.usr.roles.includes("ENGINEER")){ this.router.navigateByUrl('engineer'); }else if(this.usr.roles.includes("ADMIN")){ this.router.navigateByUrl('adminpanel');  } |

|  |
| --- |
| }  } |
| **Login.component.html**  <app-navbar></app-navbar>    <div class="container">  <h1 class="mt-4 text-center"><b>COMPLAINT REDRESSAL SYSTEM</b></h1> <div class="row">  <div class="col-md-6 offset-md-3">  <form class="" #userform="ngForm">    <h2 >Log in</h2>  <p>  <label for="username" class="">Username</label>  <input type="text" id="username" name="username" class="form-control" placeholder="Username" required autofocus [(ngModel)]="loginreq.username">  </p> <p>  <label for="password" class="">Password</label>  <input type="password" id="password" name="password" class="form-control" placeholder="Password" required [(ngModel)]="loginreq.password">  </p>  <button class="btn btn-lg btn-primary btn-block" type="submit" [disabled]="userform.invalid" (click)="signin()">Sign in</button> </form>  </div>  </div>  </div> |
| **Login.component.ts**  import { Component } from '@angular/core'; import { request } from '../models/loginRequestDto'; import { AuthService } from '../services/auth.service';  import { MatSnackBar, MatSnackBarHorizontalPosition, MatSnackBarVerticalPosition } from  '@angular/material/snack-bar';  import { Router } from '@angular/router';      @Component({  selector: 'app-login', templateUrl: './login.component.html', styleUrls: ['./login.component.css']  })  export class LoginComponent { loginreq:request = new request();  horizontalPosition: MatSnackBarHorizontalPosition = 'end'; verticalPosition: MatSnackBarVerticalPosition = 'top'; constructor( private authservice:AuthService, private snackbar:MatSnackBar,  private router: Router    ){}    signin(){ |

|  |
| --- |
| this.authservice.login(this.loginreq).subscribe(res=>{ const rolesString = res.roles.replace(/[\[\]"]+/g, '');  const rolesArray = rolesString.split(',').map((role: string) => role.trim());  this.authservice.currentUser.authentication=true; this.authservice.currentUser.username=res.authentication; this.authservice.currentUser.roles=rolesArray; this.snackbar.open("Authentication Successful!",  "Dismiss",{duration:3000,horizontalPosition:this.horizontalPosition, verticalPosition: this.verticalPosition,}) this.loginreq = new request(); this.router.navigate(['/']);  },  (err)=>{ this.snackbar.open("Authentication Failure!", "Dismiss",{duration:3000,horizontalPosition: this.horizontalPosition,  verticalPosition: this.verticalPosition,}) this.loginreq = new request();  }  );    }    } |
| **Complaintdetails.component.html**  <div class="card px-3 py-3 mx-3 my-3">  <div class="row">  <div class="col"><h1 style="float: left;display: inline;">Complaint Details</h1></div>  <div class="col"><button class="btn btn-warning" style="float: right;"  (click)="closedialog()">X</button></div>  </div>  <table class="table table-bordered">  <tbody>  <tr>  <th>Complaint Reference Number</th>  <th>ABCCR000{{complaint.complaintId}}</th>  </tr>  <tr>  <th>Customer Name</th>  <td>{{complaint.name}}</td>  </tr>  <tr>  <th>Customer Address</th>  <td>{{complaint.address}}</td>  </tr>  <tr>  <th>Pincode</th>  <td>{{complaint.pin.pin}}</td>  </tr>  <tr>  <th>Customer Contact Number</th>  <td>{{complaint.contactNo}}</td>  </tr>  <br>  <tr>  <th>Complaint Subject</th> |

|  |
| --- |
| <td>{{complaint.subject}}</td>  </tr>  <tr>  <th>Complaint Details</th>  <td>{{complaint.details}}</td>  </tr>  <tr>  <th>Complaint Feedback</th>  <td>{{complaint.feedback==null?'Not Given':complaint.feedback}}</td>  </tr>  <tr>  <th>Complaint raised date</th>  <td>{{complaint.createdAt | date:"dd-MMM-yy hh:mm"}}</td>  </tr>  </tbody>  </table>  </div> |
| **Complaintdetails.component.ts**  import { Component, Inject } from '@angular/core';  import { MAT\_DIALOG\_DATA, MatDialogRef } from '@angular/material/dialog'; import { complaint } from 'src/app/models/Complaint'; import { TicketService } from 'src/app/services/ticket.service';    @Component({  selector: 'app-complaintdetails', templateUrl: './complaintdetails.component.html', styleUrls: ['./complaintdetails.component.css']  })  export class ComplaintdetailsComponent { complaint:complaint; constructor(  @Inject(MAT\_DIALOG\_DATA) private data:any, private ref:MatDialogRef<ComplaintdetailsComponent>, private ticketservice:TicketService ){console.log('passed data',data)} ngOnInit(){  this.ticketservice.getComplaintdataByTicketid(this.data).subscribe(x=>this.complaint=x);  }  closedialog(){  this.ref.close();  }  } |
| **Manager.component.html**  <div class="card" >  <h1 >Manager Dashboard</h1>  <div class="row">  <button class="btn btn-primary mx-3" style="width:10%;float:left;display: inline;" routerLink="managerall">View All Tickets</button>  <button class="btn btn-primary mx-3" style="width:10%;float:left;display: inline;" routerLink="manageropen">View Open Tickets</button> </div>  </div>  <div class="card my-4">  <router-outlet></router-outlet> </div> |

|  |
| --- |
| **Manager.component.ts** import { Component } from '@angular/core'; import { Engineer } from '../models/Engineer'; import { Ticket } from '../models/Ticket';  import { TicketService } from '../services/ticket.service';    @Component({  selector: 'app-manager', templateUrl: './manager.component.html', styleUrls: ['./manager.component.css']  })  export class ManagerComponent {    } |
| **Manageralltickets.component.html**  <table class="table table-striped table-outlined">  <thead class="table-dark">  <tr>  <th class="text-center">TicketID</th>  <th class="text-center">Status</th>  <th class="text-center">Date of creation</th>  <th class="text-center" >Last Updated</th>  <th class="text-center">Area Manager</th>  <th class="text-center">Complaint Details</th>  <th class="text-center">Assigned Engineer</th>  </tr>  </thead>  <tbody>    <tr \*ngFor="let ticket of tickets">    <td class="text-center">{{ticket.ticketId}}</td>  <td [ngStyle]="{'background-color': (ticket.status=='CLOSED')?  '#00db3a':(ticket.status=='RAISED')? '#00b3ff':(ticket.status=='ASSIGNED')?'#f5c105':'#d9d1e3' }" class="text-center" style="color: white;"> {{ticket.status}} </td>  <td class="text-center">{{ticket.createdAt | date :"dd-MMM-yyyy,hh:mm:ss"}}</td>  <td class="text-center">{{ticket.updatedAt | date :"dd-MMM-yyyy,hh:mm:ss"}}</td>  <td class="text-center">{{(ticket.manager!==null)? ticket.manager.firstName:"Not  Assigned" + " " }}{{ (ticket.manager!==null)? ticket.manager.lastName : " " }}</td>  <td class="text-center"><button class="btn btn-warning btn-sm"  (click)="complaintdetails(ticket)"><i class="bi bi-eye-fill"></i></button></td>  <td class="text-center">{{(ticket.engineer!==null)?ticket.engineer.firstName:"Not  Assigned" + " " }}{{ (ticket.engineer!==null)? ticket.engineer.lastName :" " }}</td>    </tr>    </tbody>  </table> |
| **Manageralltickets.component.ts** import { Component } from '@angular/core'; import { Ticket } from '../models/Ticket'; import { TicketService } from '../services/ticket.service'; import { MatDialog } from '@angular/material/dialog'; import { ComplaintdetailsComponent } from  '../manager/complaintdetails/complaintdetails.component'; |

|  |
| --- |
| @Component({  selector: 'app-manageralltickets', templateUrl: './manageralltickets.component.html', styleUrls: ['./manageralltickets.component.css']  })  export class ManagerallticketsComponent { tickets:Ticket[]; constructor(  private tktService:TicketService, private dialog:MatDialog  ){  }  ngOnInit(){  this.tktService.getAllTickets().subscribe(x=>{this.tickets=x;});    }  complaintdetails(tkt:Ticket){  this.dialog.open(ComplaintdetailsComponent,{width:'40%',data:tkt.ticketId});  }  } |
| **Manageropentickets.component.html**  <table class="table table-striped table-outlined">  <thead class="table-dark">  <tr>  <th class="text-center">TicketID</th>  <th class="text-center">Status</th>  <th class="text-center">Date of creation</th>  <th class="text-center" >Last Updated</th>  <th class="text-center">Area Manager</th>  <th class="text-center">Complaint Details</th>  <th class="text-center">Assigned Engineer</th>  </tr>  </thead>  <tbody>    <tr \*ngFor="let ticket of tickets">    <td class="text-center">{{ticket.ticketId}}</td>  <td [ngStyle]="{'background-color': (ticket.status=='CLOSED')?  '#00db3a':(ticket.status=='RAISED')? '#00b3ff':(ticket.status=='ASSIGNED')?'#f5c105':'#d9d1e3' }" class="text-center" style="color: white;"> {{ticket.status}} </td>  <td class="text-center">{{ticket.createdAt | date :"dd-MMM-yyyy,hh:mm:ss"}}</td>  <td class="text-center">{{ticket.updatedAt | date :"dd-MMM-yyyy,hh:mm:ss"}}</td>  <td class="text-center">{{(ticket.manager!==null)? ticket.manager.firstName:"Not  Assigned" + " " }}{{ (ticket.manager!==null)? ticket.manager.lastName : " " }}</td>  <td class="text-center"><button class="btn btn-warning btn-sm"  (click)="complaintdetails(ticket)"><i class="bi bi-eye-fill"></i></button></td>  <td class="text-center"><button \*ngIf="ticket.engineer==null" class='btn btn-warning' (click)='assignengineer(ticket)'>Assign  Engineer</button>{{(ticket.engineer!==null)?ticket.engineer.firstName:"" + " " }}{{ (ticket.engineer!==null)? ticket.engineer.lastName :" " }}</td>    </tr> |

|  |
| --- |
| </tbody>  </table> |
| **Manageropentickets.component.ts** import { Component } from '@angular/core'; import { AuthService } from '../services/auth.service'; import { TicketService } from '../services/ticket.service'; import { Ticket } from '../models/Ticket'; import { user } from '../models/user'; import { Manager } from '../models/Manager'; import { MatDialog } from '@angular/material/dialog'; import { ComplaintdetailsComponent } from  '../manager/complaintdetails/complaintdetails.component';  import { AssignengineerComponent } from '../assignengineer/assignengineer.component';    @Component({  selector: 'app-manageropentickets', templateUrl: './manageropentickets.component.html', styleUrls: ['./manageropentickets.component.css']  })  export class ManageropenticketsComponent { usr:user; tickets:Ticket[];  constructor(private authservice:AuthService, private tktservice:TicketService, private dialog:MatDialog){    }  ngOnInit(){  this.authservice.getCurrentUser().subscribe(d=>{ this.usr=d;  this.tktservice.getManagerOpenTickets(this.usr.userid).subscribe(x=>this.tickets=x);  });    }  complaintdetails(tkt:Ticket){  this.dialog.open(ComplaintdetailsComponent,{width:'40%',data:tkt.ticketId});  }  assignengineer(tkt:Ticket){  this.dialog.open(AssignengineerComponent,{width:'40%',data:tkt});  }  } |
| **Complaint.ts**  import { Customer } from "./Customer";  import { Ticket } from "./Ticket";  import { pin } from "./pin";    export class complaint{ complaintId:number;  name:string; address:string;  pin:pin;  contactNo:number; customer:Customer; subject:string; |

|  |
| --- |
| details:string; feedback:string; ticket:Ticket[]; createdAt:Date;  lastUpdated:Date;  } |
| **Customer.ts**  import { pin } from "./pin";  import { user } from "./user";    export class Customer extends user {  address:string;  pin:pin;  contactNo:number; constructor(usr:user){  super(usr);  }  } |
| **Engineer.ts** import { pin } from "./pin";  import { user } from "./user";    export class Engineer extends user{  locations:pin[]; contactNo:number; constructor(usr:user){  super(usr);  }  } |
| **loginrequestDto.ts** export class request{ username:string;  password:string;  } |
| **Manager.ts**  import { pin } from "./pin";  import { user } from "./user";    export class Manager extends user{ pin:pin;  contactNo:number; constructor(usr:user){  super(usr);  }  } |
| **Pin.ts**  export class pin{ pin:number;  constructor(pin:number){ this.pin=pin;  }  } |
| **Status.ts**  export class status{ |

|  |
| --- |
| status:number;  } |
| **Ticket.ts** import { complaint } from "./Complaint"; import { Engineer } from "./Engineer"; import { Manager } from "./Manager";    export class Ticket{ ticketId:number; status:string; complaint:complaint; manager:Manager; engineer:Engineer; comment:string; createdAt:Date;  updatedAt:Date;  } |
| **User.ts** export class user { userid:number; firstName:string; lastName:string; username:string; password:string; email:string;  roles:string;  accountStatus:boolean; constructor(usr:user){ this.userid=usr.userid; this.firstName=usr.firstName; this.lastName=usr.lastName; this.username=usr.username; this.password=usr.password;  this.email=usr.email; this.roles=usr.roles;  this.accountStatus=usr.accountStatus;  }  } |
| **Navbar.component.html**  <nav class="navbar navbar-light bg-light justify-content-between">  <a class="navbar-brand" routerLink="">ABC Telecom Ltd.</a>    <form class="form-inline">  <span>  Welcome {{currentUser.username}}!!  </span>    <button class="btn btn-warning my-2 my-sm-0" \*ngIf="!currentUser.authentication" routerLink="login">Login</button>  <button class="btn btn-warning my-2 my-sm-0" \*ngIf="currentUser.authentication" (click)="logout()">Log out</button>  </form>  </nav> |

|  |
| --- |
| **Navbar.component.ts**  import { Component } from '@angular/core'; import { AuthService } from '../services/auth.service'; import { Router } from '@angular/router';    @Component({  selector: 'app-navbar', templateUrl: './navbar.component.html', styleUrls: ['./navbar.component.css']  })  export class NavbarComponent {  currentUser:{  authentication:boolean, username:string, roles:string[]  }= {  authentication:false,  username:"guest",  roles:[]  };  constructor(  private authservice:AuthService, private router:Router){  }  ngOnInit(){ if(this.authservice.currentUser.username!='guest'){ this.currentUser = this.authservice.currentUser;    }  }  logout(){  this.authservice.logout();  this.currentUser = this.authservice.currentUser; this.router.navigateByUrl("/");  }  } |
| **Auth.service.ts**  import { HttpClient, HttpHeaders } from '@angular/common/http'; import { Injectable } from '@angular/core'; import { request } from '../models/loginRequestDto'; import { Observable } from 'rxjs';  import { MatSnackBar, MatSnackBarHorizontalPosition, MatSnackBarVerticalPosition } from  '@angular/material/snack-bar';  import { user } from '../models/user';    @Injectable({  providedIn: 'root'  })  export class AuthService {  private baseUrl:string = "http://localhost:8080/" horizontalPosition: MatSnackBarHorizontalPosition = 'end'; verticalPosition: MatSnackBarVerticalPosition = 'top'; currentUser:{  authentication:boolean, |

|  |
| --- |
| username:string, roles:string[]  } = {  authentication:false,  username:"guest",  roles:[]  }    constructor(private http:HttpClient, private snackbar:MatSnackBar,) { }    login(req:request):Observable<any>{ const body = new FormData(); body.append('username', req.username); body.append('password', req.password); const headers = new HttpHeaders();  headers.append('Content-Type', 'multipart/form-data');  return this.http.post(this.baseUrl+"auth",body,{headers,withCredentials: true});  }    logout():Observable<any>{ this.currentUser = { authentication:false,  username:"guest",  roles:[]  }  this.snackbar.open("Logged Out!",  "Dismiss",{duration:3000,horizontalPosition:this.horizontalPosition, verticalPosition: this.verticalPosition,}) return this.http.get(this.baseUrl+"logout");  }    hasRole(role: string): boolean {  return this.currentUser.roles.includes(role);  }    hasAnyRole(roles: string[]): boolean {  return this.currentUser.roles.some(r => roles.includes(r));  }  isLoggedIn():boolean{  return this.currentUser.authentication;  }  getCurrentUser():Observable<user>{ return  this.http.get<user>("http://localhost:8080/api/ticket/"+'getcurrentuser',{withCredentials:true});  }  } |
| **Complaint.service.ts**  import { HttpClient, HttpParams } from '@angular/common/http'; import { Injectable } from '@angular/core'; import { complaint } from '../models/Complaint'; import { Observable } from 'rxjs'; import { status } from '../models/status'; import { pin } from '../models/pin';  import { Manager } from '../models/Manager'; |

|  |
| --- |
| @Injectable({  providedIn: 'root'  })  export class ComplaintService {  baseUrl:string = 'http://localhost:8080/api/complaint/' constructor(private http:HttpClient) { }  raiseComplaint(cmp:complaint):Observable<any>{  return this.http.post<any>(this.baseUrl+'raise',cmp,{withCredentials:true});  }  getAllComplaints():Observable<complaint[]>{  return this.http.get<complaint[]>(this.baseUrl+'all',{withCredentials:true});  }  getCustomerComplaints(custId:number):Observable<complaint[]>{ let params:HttpParams = new HttpParams().set("cid",custId); return  this.http.get<complaint[]>(this.baseUrl+'cuscomplaints',{params:params,withCredentials:true});  }  ticketReRaise(cmp:complaint):Observable<any>{  return this.http.post<any>(this.baseUrl+'ticketreraise',cmp,{withCredentials:true});  }  postFeedback(cmp:complaint):Observable<status>{  return this.http.post<status>(this.baseUrl+'addfeedback',cmp,{withCredentials:true});  }  getAllPin():Observable<pin[]>{  return this.http.get<pin[]>(this.baseUrl+"allpin",{withCredentials:true});  }  addPin(p:pin):Observable<status>{  return this.http.post<status>(this.baseUrl+"addpin",p,{withCredentials:true});  }  getManagerForPin(pin:number):Observable<Manager>{ let params = new HttpParams().set("pin",pin); return  this.http.get<Manager>(this.baseUrl+'managerforpin',{params:params,withCredentials:true})  }  } |
| **Ticket.service.ts**  import { HttpClient, HttpParams } from '@angular/common/http'; import { Injectable } from '@angular/core'; import { Observable } from 'rxjs'; import { Ticket } from '../models/Ticket'; import { complaint } from '../models/Complaint'; import { status } from '../models/status'; import { Engineer } from '../models/Engineer'; import { user } from '../models/user';    @Injectable({  providedIn: 'root'  })  export class TicketService {  baseUrl: string = 'http://localhost:8080/api/ticket/' constructor(private http: HttpClient) { }    getAllTickets(): Observable<Ticket[]> {  return this.http.get<Ticket[]>(this.baseUrl + 'all', { withCredentials: true }); |

|  |
| --- |
| }  getTicketById(ticketId: number): Observable<Ticket> {  let param: HttpParams = new HttpParams().set("ticketId", ticketId); return this.http.get<Ticket>(this.baseUrl + 'ticketdata', { params: param });  }  getComplaintdataByTicketid(ticketId: number): Observable<complaint> { let param: HttpParams = new HttpParams().set("ticketId", ticketId);  return this.http.get<complaint>(this.baseUrl + 'complaintdata', { params: param, withCredentials: true });  }  updateStatus(tkt: Ticket): Observable<status> {  return this.http.patch<status>(this.baseUrl + 'updatestatus', tkt,{withCredentials:true});  }  updateComment(tkt: Ticket): Observable<status> {  return this.http.patch<status>(this.baseUrl + 'updatecomment', tkt);  }  assignEngineer(tkt: Ticket): Observable<status> {  return this.http.patch<status>(this.baseUrl + 'assignEngineer', tkt,{withCredentials:true});  }  getAllEngineerTickets(engineerID: number): Observable<Ticket[]> { let param: HttpParams = new HttpParams().set("engineerid", engineerID); return this.http.get<Ticket[]>(this.baseUrl + 'engineeralltickets', { params: param ,withCredentials:true});  }  getEngineerOpenTickets(engId: number): Observable<Ticket[]> { let param: HttpParams = new HttpParams().set("engineerid", engId);  return this.http.get<Ticket[]>(this.baseUrl + 'engineeropentickets', { params: param ,withCredentials:true});  }  getManagerOpenTickets(mgrId: number): Observable<Ticket[]> { let param: HttpParams = new HttpParams().set("managerid", mgrId);  return this.http.get<Ticket[]>(this.baseUrl + 'manageropentickets', { params: param, withCredentials: true });  }  getAllManagerTickets(mgrId: number): Observable<Ticket[]> { let param: HttpParams = new HttpParams().set("managerid", mgrId);  return this.http.get<Ticket[]>(this.baseUrl + 'manageralltickets', { params: param });  }  getEngineerForPin(pincode: number): Observable<Engineer[]> { let param: HttpParams = new HttpParams().set("pin", pincode); return this.http.get<Engineer[]>(this.baseUrl + 'engineerforpin', { params:  param,withCredentials:true });  }  assignmanager(t: Ticket): Observable<status> {  return this.http.patch<status>(this.baseUrl + 'assignmanager', t, { withCredentials: true });  }  } |
| **User.service.ts**  import { HttpClient, HttpClientModule, HttpParams } from '@angular/common/http'; import { Injectable } from '@angular/core'; import { user } from '../models/user'; import { Observable } from 'rxjs'; import { status } from '../models/status'; import { Manager } from '../models/Manager'; import { Engineer } from '../models/Engineer'; |

|  |
| --- |
| import { Customer } from '../models/Customer';    @Injectable({  providedIn: 'root'  })  export class UserService {  baseUrl:string='http://localhost:8080/api/user/' constructor(private http:HttpClient) { }    getAllUsers():Observable<user[]>{  return this.http.get<user[]>(this.baseUrl +'all', {withCredentials:true});  }    adduser(usr:user):Observable<status>{  return this.http.post<status>(this.baseUrl+'adduser',usr);  }  deleteuser(usrid:number):Observable<status>{  let params:HttpParams = new HttpParams().set("userid",usrid)  return this.http.delete<status>(this.baseUrl+'deleteuser', {params:params});  }  addManager(mgr:Manager):Observable<status>{  return this.http.post<status>(this.baseUrl+'addmanager',mgr,{withCredentials:true});  }  addEngineer(egr:Engineer):Observable<status>{ console.log("Service Method called !!");  return this.http.post<status>(this.baseUrl+'addengineer',egr,{withCredentials:true});  }  addCustomer(cus:Customer){  return this.http.post<status>(this.baseUrl+'addcustomer',cus,{withCredentials:true});  }  getAllManagers():Observable<Manager[]>{  return this.http.get<Manager[]>(this.baseUrl+'allmanagers',{withCredentials:true});  }  getAllEngineers():Observable<Engineer[]>{  return this.http.get<Engineer[]>(this.baseUrl+'allengineers',{withCredentials:true});  }  getAllCustomers():Observable<Customer[]>{  return this.http.get<Customer[]>(this.baseUrl+'allcustomers');  }  } |
| **App-routing.module.ts**  import { NgModule } from '@angular/core';  import { RouterModule, Routes } from '@angular/router'; import { LoginComponent } from './login/login.component'; import { LandingComponent } from './landing/landing.component';  import { loginGuard } from './gaurds/login.guard';  import { ManagerallticketsComponent } from './manageralltickets/manageralltickets.component'; import { ManageropenticketsComponent } from  './manageropentickets/manageropentickets.component'; import { ManagerComponent } from './manager/manager.component'; import { CustomerComponent } from './customer/customer.component'; import { AdminpanelComponent } from './adminpanel/adminpanel.component'; import { EngineerComponent } from './engineer/engineer.component'; import { AllusersComponent } from './adminpanel/allusers/allusers.component'; |

|  |
| --- |
| import { CreatecomplaintComponent } from  './customer/createcomplaint/createcomplaint.component';  import { AllcomplaintComponent } from './customer/allcomplaint/allcomplaint.component'; import { AdminallcomplaintsComponent } from  './adminpanel/adminallcomplaints/adminallcomplaints.component'; import { AdminallticketsComponent } from  './adminpanel/adminalltickets/adminalltickets.component'; import { PincodedetailsComponent } from  './adminpanel/pincodedetails/pincodedetails.component';  import { AllticketsComponent } from './engineer/alltickets/alltickets.component';  import { OpenticketsComponent } from './engineer/opentickets/opentickets.component';    const routes: Routes = [  {path:'login',component:LoginComponent},  {path:'',component:LandingComponent,canActivate:[loginGuard],children:[  {path:'manager',component:ManagerComponent,children:[  {path:'managerall', component:ManagerallticketsComponent},  {path:'manageropen', component:ManageropenticketsComponent}  ]},  {path:'customer',component:CustomerComponent,children:[  {path:'allcomplaints',component:AllcomplaintComponent},  {path:'\*\*',redirectTo:'allcomplaints',pathMatch:'full'}  ]},  {path:'engineer',component:EngineerComponent,children:[  {path:'alltickets',component:AllticketsComponent},  {path:'opentickets',component:OpenticketsComponent}  ]},  {path:'adminpanel', component:AdminpanelComponent,children:[  {path:'viewall', component:AllusersComponent},  {path:'allcomplaints',component:AdminallcomplaintsComponent},  {path:'alltickets',component:AdminallticketsComponent},  {path:'pincodedetails',component:PincodedetailsComponent}  ]}  ]},  {path:'\*\*',redirectTo:'',pathMatch:'full'}    ];    @NgModule({  imports: [RouterModule.forRoot(routes)], exports: [RouterModule]  })  export class AppRoutingModule { } |
| **App.component.html**  <div class="container">  <router-outlet></router-outlet> </div> |
| **App.component.ts**  import { Component } from '@angular/core';    @Component({  selector: 'app-root',  templateUrl: './app.component.html', styleUrls: ['./app.component.css']  }) |

|  |
| --- |
| export class AppComponent {  title = 'ComplaintRedressalSystem-ui';  } |
| **App.module.ts**  import { NgModule } from '@angular/core'; import { BrowserModule } from '@angular/platform-browser'; import {HttpClientModule} from '@angular/common/http'; import { AppRoutingModule } from './app-routing.module'; import { AppComponent } from './app.component'; import { LoginComponent } from './login/login.component'; import { NavbarComponent } from './navbar/navbar.component'; import { FormsModule } from '@angular/forms';  import {MatSnackBarModule} from '@angular/material/snack-bar';  import {BrowserAnimationsModule} from '@angular/platform-browser/animations'; import { CookieService } from 'ngx-cookie-service'; import { LandingComponent } from './landing/landing.component'; import { CustomerComponent } from './customer/customer.component'; import { ManagerComponent } from './manager/manager.component'; import { EngineerComponent } from './engineer/engineer.component'; import { AdminpanelComponent } from './adminpanel/adminpanel.component';  import { ManagerallticketsComponent } from './manageralltickets/manageralltickets.component'; import { ManageropenticketsComponent } from  './manageropentickets/manageropentickets.component'; import { AllusersComponent } from './adminpanel/allusers/allusers.component'; import { AdduserComponent } from './adminpanel/adduser/adduser.component'; import {MatDialogModule}from '@angular/material/dialog'; import {MatRadioModule} from '@angular/material/radio'; import { CreatecomplaintComponent } from  './customer/createcomplaint/createcomplaint.component'; import { AllcomplaintComponent } from './customer/allcomplaint/allcomplaint.component'; import { ViewticketComponent } from './customer/viewticket/viewticket.component'; import { ComplaintidComponent } from './customer/complaintid/complaintid.component'; import { ComplaintdetailsComponent } from  './manager/complaintdetails/complaintdetails.component'; import { AdminallcomplaintsComponent } from  './adminpanel/adminallcomplaints/adminallcomplaints.component'; import { AdminallticketsComponent } from  './adminpanel/adminalltickets/adminalltickets.component'; import { AssignmanagerComponent } from  './adminpanel/assignmanager/assignmanager.component';  import { AssignengineerComponent } from './assignengineer/assignengineer.component'; import { PincodedetailsComponent } from  './adminpanel/pincodedetails/pincodedetails.component';  import { AddpinComponent } from './adminpanel/addpin/addpin.component'; import { AllticketsComponent } from './engineer/alltickets/alltickets.component'; import { OpenticketsComponent } from './engineer/opentickets/opentickets.component'; import { UpdatestatusComponent } from './engineer/updatestatus/updatestatus.component'; import { FeedbackComponent } from './customer/feedback/feedback.component'; import { TicketraiseComponent } from './customer/ticketraise/ticketraise.component';      @NgModule({  declarations: [  AppComponent,  LoginComponent, |
| NavbarComponent,  LandingComponent,  CustomerComponent,  ManagerComponent,  EngineerComponent,  AdminpanelComponent,  ManagerallticketsComponent,  ManageropenticketsComponent,  AllusersComponent,  AdduserComponent,  CreatecomplaintComponent,  AllcomplaintComponent,  ViewticketComponent,  ComplaintidComponent,  ComplaintdetailsComponent,  AdminallcomplaintsComponent,  AdminallticketsComponent,  AssignmanagerComponent, AssignengineerComponent,  PincodedetailsComponent,  AddpinComponent,  AllticketsComponent,  OpenticketsComponent,  UpdatestatusComponent,  FeedbackComponent,  TicketraiseComponent    ],  imports: [  BrowserModule,  AppRoutingModule,  HttpClientModule,  FormsModule,  MatSnackBarModule,  BrowserAnimationsModule,  MatDialogModule,  MatRadioModule  ],  providers: [CookieService],  bootstrap: [AppComponent]  })  export class AppModule { } |