Complaint Redressal (Source Code)

Developed By :*G Rajya Lakshmi Suvarna Himavalli*

Back -End

ComplaintRedressalApplication.java

```
package com.simplilearn.finalphase2;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

@SpringBootApplication
//@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class })
//@EnableAutoConfiguration(exclude = {DataSourceAutoConfiguration.class })
public class ComplaintRedressalApplication {

	public static void main(String[] args) {
		SpringApplication.run(ComplaintRedressalApplication.class, args);
	}

}
```

CorsConfiguration.java

```
package com.simplilearn.finalphase2.configuration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
@Component
public class CorsConfiguration {

	private static final String GET = "GET";
	private static final String POST = "POST";
```

```
	private static final String DELETE = "DELETE";
	private static final String PUT = "PUT";

	@Bean
	public WebMvcConfigurer corsConfigurer() {
		return new WebMvcConfigurer() {
			@Override
			public void addCorsMappings(CorsRegistry registry) {
				registry.addMapping("/**")
				.allowedMethods(GET,PUT,POST,DELETE)
				.allowedHeaders("*")
				.allowedOriginPatterns("*")
				.allowCredentials(true);
			}

		};
	}

}
```

JwtAuthenticationEntryPoint.java

```
package com.simplilearn.finalphase2.configuration;

import java.io.IOException;


import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

@Component
public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {

	@Override
	public void commence(HttpServletRequest request, HttpServletResponse response,
			AuthenticationException authException) throws IOException,
ServletException {
		//sending msg that access is unauthorized
```

```
response.sendError(HttpServletResponse.SC_UNAUTHORIZED,"Unauthorized");

        }

}
```

JwtRequestFilter.java

```
package com.simplilearn.finalphase2.configuration;

import java.io.IOException;


import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Lazy;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import com.simplilearn.finalphase2.service.JwtService;
import com.simplilearn.finalphase2.util.JwtUtil;

import io.jsonwebtoken.ExpiredJwtException;

@Component
public class JwtRequestFilter extends OncePerRequestFilter {

        public static String CURRENT_USER = "";

        @Autowired
        private JwtUtil jwtutil;

        @Lazy
```

```
	@Autowired
	private JwtService jwtservice;

	@Override
	protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
response,
			FilterChain filterChain)
			throws ServletException, IOException {
		final String header = request.getHeader("Authorization");

		String jwtToken = null;
		String userName = null;
		if (header != null && header.startsWith("Bearer ")) {
			jwtToken = header.substring(7);

			try {

				userName = jwtutil.getUserNameFromToken(jwtToken);
				CURRENT_USER = userName;

			} catch (IllegalArgumentException e) {
				System.out.println("Unable to get JWT token");
			} catch (ExpiredJwtException e) {
				System.out.println("Jwt token is expired");
			}

		} else {
			System.out.println("Jwt token does not start with bearer");
		}

		if (userName != null &&
SecurityContextHolder.getContext().getAuthentication() == null) {

			UserDetails userDetails = jwtservice.loadUserByUsername(userName);

			if (jwtutil.ValidateToken(jwtToken, userDetails)) {
				UsernamePasswordAuthenticationToken
usernamePasswordAuthenticationToken =
						new
UsernamePasswordAuthenticationToken(userDetails,
								null, userDetails.getAuthorities());

				usernamePasswordAuthenticationToken.setDetails
				(new WebAuthenticationDetailsSource().buildDetails(request));
```

```
SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationTok
en);
                        }

                }

                filterChain.doFilter(request, response);
        }

}
/*
 * This class wll retrieve the header token
 * once we get the token we validate all deired details and pass the requests*/
```

WebSecurityConfiguration.java

```
package com.simplilearn.finalphase2.configuration;

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

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Lazy;
import org.springframework.http.HttpHeaders;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationMana
gerBuilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodS
ecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAd
apter;
import org.springframework.security.config.http.SessionCreationPolicy;
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.authentication.UsernamePasswordAuthenticationFilter;
import org.springframework.stereotype.Component;

import com.simplilearn.finalphase2.service.JwtService;


@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
@Component
public class WebSecurityConfiguration extends WebSecurityConfigurerAdapter {
	@Autowired
	private JwtAuthenticationEntryPoint jwtauthenticationentrypoint;

	//@Lazy
	@Autowired
	private JwtRequestFilter jwtrequestfilter;

	//@Lazy
	@Autowired
	private UserDetailsService jwtservice;

	@Bean
	@Override
	public AuthenticationManager authenticationManagerBean() throws Exception {
		return super.authenticationManagerBean();
	}

	@Override
	protected void configure(HttpSecurity httpsecurity) throws Exception {
		httpsecurity.cors();
		httpsecurity.csrf().disable().authorizeRequests()
		.antMatchers("/authenticate","/registerNewCustomer")
		.permitAll().antMatchers(HttpHeaders.ALLOW)
				.permitAll().anyRequest().authenticated()
				.and()
				.exceptionHandling()

.authenticationEntryPoint(jwtauthenticationentrypoint).and().sessionManagement()
				.sessionCreationPolicy(SessionCreationPolicy.STATELESS);

		httpsecurity.addFilterBefore(jwtrequestfilter,
UsernamePasswordAuthenticationFilter.class);
```

```
	}

	@Bean
	public PasswordEncoder passwordEncoder() {
		return new BCryptPasswordEncoder();
	}

	@Autowired
	public void configureGlobal(AuthenticationManagerBuilder
authenticatnManagerBuilder) throws Exception {

authenticatnManagerBuilder.userDetailsService(jwtservice).passwordEncoder(passwordEnco
der());
	}

}

/*
 * All these annotations will take care of role based authentication like
 * specifically for admin n for user dont authenticate end point specified
 * inside antmatchers here-- .authorizeRequests().antMatchers("").permitAll()
 *
 * .sessionCreationPolicy(SessionCreationPolicy.STATELESS)-->for user security purpose this
 * will not store the user password in session
 **/
```

ComplaintController.java

```
package com.simplilearn.finalphase2.controller;

import java.util.List;


import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Pageable;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
```

```
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.simplilearn.finalphase2.entity.Complaint;
import com.simplilearn.finalphase2.service.ComplaintService;

@RestController
@CrossOrigin
public class ComplaintController {
	@Autowired
	private ComplaintService complaintService;

	//only customer can register complaint
	@PreAuthorize("hasRole('Customer')")
	@PostMapping(value= {"/add/new/complaint"})
	public Complaint addNewComplaint(@RequestBody Complaint complaint) {
		return complaintService.addNewComplaint(complaint);

	}

	//only admin can delete any complaint if as required
	@PreAuthorize("hasRole('Admin')")
	@DeleteMapping(value= {"/delete/complaint/{complaintId}"})
	public void deleteComplaint(@PathVariable ("complaintId") Integer complaintId) {
		complaintService.deleteComplaint(complaintId);
	}

	//only admin and manager can get all complaints list
	@PreAuthorize("hasAnyRole('Admin','Manager','Customer')")
	@GetMapping(value= {"/get/complaint/list"})
	public List<Complaint> getAllComplaints() {
		List<Complaint> complaints = complaintService.getAllComplaint();
//		System.out.println("Size of complaints list id: "+ complaints.size());

		return complaints;
	}

	@PreAuthorize("hasAnyRole('Engineer')")
	@GetMapping(value= {"/get/complaint/list/engineer"})
	public List<Complaint> getAllComplaintsEngg() {
		return complaintService.getAllComplaintsForEngineer();
	}
```

```
	@PreAuthorize("hasAnyRole('Engineer')")
	@GetMapping({"/markAsResolved/{complaintId}"})
	public void markAsResolved(@PathVariable(name = "complaintId")Integer
complaintId ) {
		complaintService.markAsResolved(complaintId);
	}


	@PreAuthorize("hasAnyRole('Engineer')")
	@GetMapping({"/markAsWip/{complaintId}"})
	public void markAsWip(@PathVariable(name = "complaintId")Integer complaintId ) {
		complaintService.markAsWip(complaintId);
	}

	@PreAuthorize("hasAnyRole('Engineer')")
	@GetMapping({"/markAsInReview/{complaintId}"})
	public void markAsInReview(@PathVariable(name = "complaintId")Integer
complaintId ) {
		complaintService.markAsInReview(complaintId);
	}

	@GetMapping("/getComplaintById/{complaintId}")
	public Complaint getComplaintById(@PathVariable Integer complaintId) {
		return complaintService.getComplaintById(complaintId);

	}

	@PreAuthorize("hasRole('Customer')")
	@GetMapping({"/get/mycomplaints"})
	public List<Complaint> getMyComplaintDetails() {
		return complaintService.getMyComplaintDetails();
	}

}
```

JwtController.java

```
package com.simplilearn.finalphase2.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
```

```
import com.simplilearn.finalphase2.entity.JwtRequest;
import com.simplilearn.finalphase2.entity.JwtResponse;
import com.simplilearn.finalphase2.service.JwtService;

@RestController
@CrossOrigin
public class JwtController {
	@Autowired
	private JwtService jwtservice;

	@PostMapping({"/authenticate"})
	public JwtResponse CreateJwtToken(@RequestBody JwtRequest jwtrequest)throws
Exception {
		return jwtservice.createJwtToken(jwtrequest);
	}

}
```

RoleController.java

```
package com.simplilearn.finalphase2.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.simplilearn.finalphase2.entity.Role;
import com.simplilearn.finalphase2.service.RoleService;

@RestController
public class RoleController {
	@Autowired
	private RoleService roleservice;

	@PreAuthorize("hasRole('Admin')")
	@PostMapping({"/createNewRole"})
	public Role createNewRole(@RequestBody Role role) {
		return roleservice.createNewRole(role);

	}
}
```

UserController.java

```
package com.simplilearn.finalphase2.controller;

import java.util.List;

import javax.annotation.PostConstruct;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.simplilearn.finalphase2.dao.UserDao;
import com.simplilearn.finalphase2.entity.User;
import com.simplilearn.finalphase2.service.UserService;

@RestController
public class UserController {
	@Autowired
	private UserService userservice;

	@Autowired
	private UserDao userDao;

	@PostConstruct//will run each the program starts
	public void initRolesAndUsers() {
		userservice.initRolesAndUser();
	}

	@PostMapping({"/registerNewCustomer"})
	public User RegisterNewUser(@RequestBody User customer) {
		return userservice.RegisterNewUser(customer);
	}
```

```
	//for admin to register new engineer
	@PreAuthorize("hasRole('Admin')")
	@PostMapping({"/registerNewEngineer"})
	public User RegisterNewEngineer(@RequestBody User engg) {
		return userservice.RegisterNewEngineer(engg);
	}

	//for admin to register new manager
	@PreAuthorize("hasRole('Admin')")
	@PostMapping({"/registerNewManager"})
	public User RegisterNewManager(@RequestBody User manager) {
		return userservice.RegisterNewManager(manager);
	}

	@GetMapping({"/forAdmin"})
	@PreAuthorize("hasRole('Admin')") //this will give access only to users
having role of Admin
	public String forAdmin() {
		return "This URL is only accessible to admin";
	}

	@GetMapping({"/forManager"})
	@PreAuthorize("hasRole('Manager')")
	public String forManager() {
		return "This URL is only accessible to Manager";
	}

	@GetMapping({"/forCustomer"})
	@PreAuthorize("hasRole('Customer')")
	public String forCustomer() {
		return "This URL is only accessible to Customer";
	}

	@GetMapping({"/forEngineer"})
	@PreAuthorize("hasRole('Engineer')")
	public String forEngineer() {
		return "This URL is only accessible to Engineer";
	}

	@PreAuthorize("hasRole('Admin')")
```

```
	@GetMapping("/allUser")
	public List<User> getAllUsers() {
		return (List<User>) userDao.findAll();
	}

	@PreAuthorize("hasRole('Admin')")
	@DeleteMapping("/deleteUser/{userName}")
	public void deleteUser(@PathVariable String userName) {
		userservice.deleteUser(userName);
	}



}
```

ComplaintDao.java

```
package com.simplilearn.finalphase2.dao;

import org.springframework.data.repository.CrudRepository;


import java.util.List;

import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Repository;
import com.simplilearn.finalphase2.entity.Complaint;
import com.simplilearn.finalphase2.entity.User;

@Repository
public interface ComplaintDao extends CrudRepository<Complaint, Integer>{

	public List<Complaint> findAll(Pageable pageable);

	public List<Complaint> findByCustomer(User user);

}
```

RoleDao.java

```
package com.simplilearn.finalphase2.dao;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.simplilearn.finalphase2.entity.Role;

@Repository
public interface RoleDao extends CrudRepository<Role, String>{

}
```

UserDao.java

```
package com.simplilearn.finalphase2.dao;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.simplilearn.finalphase2.entity.User;
@Repository
public interface UserDao extends CrudRepository<User, String> {

}
```

Complaint.java

```
package com.simplilearn.finalphase2.entity;

import java.util.List;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
```

```
import javax.persistence.OneToOne;


@Entity
public class Complaint {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private int complaintId;
	private String category;
	private String heading;
	private String details;
	private String address;
	private String pincode;
	private String fullname;
	private String complaintStatus;
	@OneToOne
	private User customer;


//	@ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
//	@JoinTable(name = "customer_complaint", joinColumns = { @JoinColumn(name =
"complaint_id") },
//	inverseJoinColumns = { @JoinColumn(name = "customer_id") })
//	private Set<User> customer;


	public User getCustomer() {
		return customer;
	}


	public void setCustomer(User customer) {
		this.customer = customer;
	}


	public int getComplaintId() {
		return complaintId;
	}


	public void setComplaintId(int complaintId) {
		this.complaintId = complaintId;
	}
```

```
public String getCategory() {
	return category;
}


public void setCategory(String category) {
	this.category = category;
}


public String getHeading() {
	return heading;
}


public void setHeading(String heading) {
	this.heading = heading;
}


public String getDetails() {
	return details;
}


public void setDetails(String details) {
	this.details = details;
}


public String getAddress() {
	return address;
}


public void setAddress(String address) {
	this.address = address;
}


public String getPincode() {
	return pincode;
```

```
	}


	public void setPincode(String pincode) {
		this.pincode = pincode;
	}


	public String getFullname() {
		return fullname;
	}


	public void setFullname(String fullname) {
		this.fullname = fullname;
	}




//	public Set<User> getCustomer() {
//		return customer;
//	}
//
//
//	public void setCustomer(Set<User> customer) {
//		this.customer = customer;
//	}



	public String getComplaintStatus() {
		return complaintStatus;
	}


	public void setComplaintStatus(String complaintStatus) {
		this.complaintStatus = complaintStatus;
	}


	public Complaint() {
```

```
		super();
		// TODO Auto-generated constructor stub
	}


	public Complaint(String category, String heading, String details, String address, String
pincode, String fullname,
			String complaintStatus, User customer) {
		super();
		this.category = category;
		this.heading = heading;
		this.details = details;
		this.address = address;
		this.pincode = pincode;
		this.fullname = fullname;
		this.complaintStatus = complaintStatus;
		this.customer = customer;
	}


}
```

ComplaintCategory.java

```
package com.simplilearn.finalphase2.entity;




import javax.persistence.Entity;

import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

import javax.persistence.OneToOne;

@Entity
public class ComplaintCategory {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private int categoryId;
	private String BroadBand;
	private String PrePaid;
	private String PostPaid;
```

```
	private String PlansDetails;
	@OneToOne
	private Complaint complaint;
	public String getBroadBand() {
		return BroadBand;
	}
	public void setBroadBand(String broadBand) {
		BroadBand = broadBand;
	}
	public String getPrePaid() {
		return PrePaid;
	}
	public void setPrePaid(String prePaid) {
		PrePaid = prePaid;
	}
	public String getPostPaid() {
		return PostPaid;
	}
	public void setPostPaid(String postPaid) {
		PostPaid = postPaid;
	}
	public String getPlansDetails() {
		return PlansDetails;
	}
	public void setPlansDetails(String plansDetails) {
		PlansDetails = plansDetails;
	}
	public Complaint getComplaint() {
		return complaint;
	}
	public void setComplaint(Complaint complaint) {
		this.complaint = complaint;
	}
	public ComplaintCategory(String broadBand, String prePaid, String postPaid, String
plansDetails,
			Complaint complaint) {
		super();
		BroadBand = broadBand;
		PrePaid = prePaid;
		PostPaid = postPaid;
		PlansDetails = plansDetails;
		this.complaint = complaint;
	}
	public int getCategoryId() {
```

```
		return categoryId;
	}
	public void setCategoryId(int categoryId) {
		this.categoryId = categoryId;
	}
}
```

JwtRequest.java

```
package com.simplilearn.finalphase2.entity;
public class JwtRequest {
	private String userName;
	private String userPassword;
	public String getUserName() {
		return userName;
	}
	public void setUserName(String userName) {
		this.userName = userName;
	}
	public String getUserPassword() {
		return userPassword;
	}
	public void setUserPassword(String userPassword) {
		this.userPassword = userPassword;
	}


}
```

JwtResponse.java

```
package com.simplilearn.finalphase2.entity;
public class JwtResponse {
	private User user;
	private String jwtToken;
	public JwtResponse(User user, String jwtToken) {
		super();
		this.user = user;
		this.jwtToken = jwtToken;
	}
	public User getUser() {
		return user;
	}
	public void setUser(User user) {
		this.user = user;
```

```
	}
	public String getJwtToken() {
		return jwtToken;
	}
	public void setJwtToken(String jwtToken) {
		this.jwtToken = jwtToken;
	}
}
```

Role.java

```
package com.simplilearn.finalphase2.entity;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Role {

	@Id
	private String roleName;
	private String roleDescription;
	public String getRoleName() {
		return roleName;
	}
	public void setRoleName(String roleName) {
		this.roleName = roleName;
	}
	public String getRoleDescription() {
		return roleDescription;
	}
	public void setRoleDescription(String roleDescription) {
		this.roleDescription = roleDescription;
	}


}
```

User.java

```
package com.simplilearn.finalphase2.entity;

import java.util.Set;
```

```
import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name="users")
public class User {

	@Id
	private String userName;
	private String fullName;
	private String userpassword;

	// many uses may have many different roles
	/*
	 * By itself create a third table named USER_ROLE that will have user and its
	 * associated role details this third table will have user_id and associated
	 * role_id
	 */
	@ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
	@JoinTable(name = "USER_ROLE", joinColumns = { @JoinColumn(name =
"USER_ID") }, inverseJoinColumns = {
				@JoinColumn(name = "ROLE_ID") })
	private Set<Role> role;

	public String getUserName() {
		return userName;
	}

	public void setUserName(String userName) {
		this.userName = userName;
	}

	public String getFullName() {
		return fullName;
	}

	public void setFullName(String fullName) {
```

```
		this.fullName = fullName;
	}

	public String getUserpassword() {
		return userpassword;
	}

	public void setUserpassword(String userpassword) {
		this.userpassword = userpassword;
	}

	public Set<Role> getRole() {
		return role;
	}

	public void setRole(Set<Role> role) {
		this.role = role;
	}



}
```

ComplaintService.java

```
package com.simplilearn.finalphase2.service;


import java.util.List;


import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;

import com.simplilearn.finalphase2.configuration.JwtRequestFilter;
import com.simplilearn.finalphase2.dao.ComplaintDao;
import com.simplilearn.finalphase2.dao.UserDao;
import com.simplilearn.finalphase2.entity.Complaint;
import com.simplilearn.finalphase2.entity.User;

@Service
```

```
public class ComplaintService {

	@Autowired
	public ComplaintDao complaintDao;

	@Autowired
	public UserDao userDao;

	private static final String COMPLAINT_RAISED = "Raised";

	//mark status as resolved
	public void markAsResolved(Integer complaintId) {
		Complaint complaint = complaintDao.findById(complaintId).get();

		if (complaint != null) {
			complaint.setComplaintStatus("Resolved");
			complaintDao.save(complaint);
		}else {
			complaint.setComplaintStatus("Error");
			complaintDao.save(complaint);
		}
	}


	//mark status as in progress
	public void markAsWip(Integer complaintId) {
		Complaint complaint = complaintDao.findById(complaintId).get();

		if (complaint != null) {
			complaint.setComplaintStatus("Work in progress");
			complaintDao.save(complaint);
		}else {
			complaint.setComplaintStatus("Error");
			complaintDao.save(complaint);
		}
	}


	//mark status as in review
	public void markAsInReview(Integer complaintId) {
		Complaint complaint = complaintDao.findById(complaintId).get();

		if (complaint != null) {
			complaint.setComplaintStatus("In Review");
```

```
                complaintDao.save(complaint);
        }else {
                complaint.setComplaintStatus("Error");
                complaintDao.save(complaint);
        }
}



//adding new complaint feature for customer
public Complaint addNewComplaint(Complaint c) {
        c.setComplaintStatus(COMPLAINT_RAISED);
        return complaintDao.save(c);
}

//getting all complaints list
public List<Complaint> getAllComplaint(){
        return (List<Complaint>)complaintDao.findAll();
}

public List<Complaint> getAllComplaintsForEngineer(){
        Complaint c = new Complaint();
        String pin = c.getPincode();
        if(pin.equals("110025")) {
                return (List<Complaint>)complaintDao.findAll();
        }
        else {
                return null;
        }

}

//deleting complaint
public void deleteComplaint(Integer complaintId) {
        complaintDao.deleteById(complaintId);

}

//getting compaints by Id
public Complaint getComplaintById(Integer complaintId) {
        return complaintDao.findById(complaintId).get();
}

public List<Complaint> getMyComplaintDetails() {
```

```
		String currentUser = JwtRequestFilter.CURRENT_USER;// will get the current
username
		User user = userDao.findById(currentUser).get(); // by this we will get all the
userdetails

		return complaintDao.findByCustomer(user);
		// this method described inside orderDetailDao will return list of orders for
		// that particular user
	}
}
```

JwtService.java

```
package com.simplilearn.finalphase2.service;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Lazy;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.DisabledException;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
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.Service;

import com.simplilearn.finalphase2.dao.UserDao;
import com.simplilearn.finalphase2.entity.JwtRequest;
import com.simplilearn.finalphase2.entity.JwtResponse;
import com.simplilearn.finalphase2.entity.User;
import com.simplilearn.finalphase2.util.JwtUtil;

@Service
public class JwtService implements UserDetailsService {
	@Autowired
	private UserDao userdao;

	//@Lazy
	@Autowired
	private JwtUtil jwtutil;
```

```
	@Autowired
	private AuthenticationManager authenticationmanager;

	public JwtResponse createJwtToken(JwtRequest jwtRequest) throws Exception {
		String userName = jwtRequest.getUserName();
		String userPassword = jwtRequest.getUserPassword();
		authenticate(userName, userPassword);

		final UserDetails userDetails = loadUserByUsername(userName);

		String newGeneratedToken = jwtutil.generateToken(userDetails);
		User user= userdao.findById(userName).get();

		return new JwtResponse(user, newGeneratedToken);
	}

	@Override
	public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {

		User user = userdao.findById(username).get();
		if (user != null) {
			return new
org.springframework.security.core.userdetails.User(user.getUserName(),
					user.getUserpassword(),
					getAuthorities(user));
		}
		else
		{
			throw new UsernameNotFoundException("username is not valid");
		}
	}

	private Set getAuthorities(User user) {
		Set authorities = new HashSet();

		user.getRole().forEach(role -> {
			authorities.add(new SimpleGrantedAuthority("ROLE_" +
role.getRoleName()));
		});
		return authorities;
	}
```

```
	private void authenticate(String userName, String userPassword) throws Exception {

		try {
			authenticationmanager.authenticate(new
UsernamePasswordAuthenticationToken(userName, userPassword));
		} catch (DisabledException e) {
			throw new Exception("user is disabled");
		} catch (BadCredentialsException e) {
			throw new Exception("Bad credentials from user");
		}

	}

}
```

RoleService.java

```
package com.simplilearn.finalphase2.service;

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

import com.simplilearn.finalphase2.dao.RoleDao;
import com.simplilearn.finalphase2.entity.Role;

@Service
public class RoleService {
	@Autowired //(required=true)
	private RoleDao roledao;

	public Role createNewRole(Role role) {
		return roledao.save(role);//will save this role and return the information saved.

	}

}
```

UserService.java

```
package com.simplilearn.finalphase2.service;

import java.util.HashSet;
import java.util.Set;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import com.simplilearn.finalphase2.dao.RoleDao;
import com.simplilearn.finalphase2.dao.UserDao;
import com.simplilearn.finalphase2.entity.Role;
import com.simplilearn.finalphase2.entity.User;

@Service
public class UserService {
	@Autowired //(required=true)
	private UserDao userdao;

	@Autowired //(required=true)
	private RoleDao roledao;

	@Autowired
	private PasswordEncoder passwordEncoder;


	public void deleteUser(String userName) {
		userdao.deleteById(userName);
	}

	public User RegisterNewUser(User user) {
		Role role = roledao.findById("Customer").get();

		Set<Role> roles = new HashSet<>();
		roles.add(role);
		user.setRole(roles);

		user.setUserpassword(getEncodedPassword(user.getUserpassword()));

		return userdao.save(user);
	}

	public User RegisterNewEngineer(User engg) {
		Role role = roledao.findById("Engineer").get();

		Set<Role> roles = new HashSet<>();
		roles.add(role);
		engg.setRole(roles);
```

```
		engg.setUserpassword(getEncodedPassword(engg.getUserpassword()));

		return userdao.save(engg);
	}

	public User RegisterNewManager(User manager) {
		Role role = roledao.findById("Manager").get();

		Set<Role> roles = new HashSet<>();
		roles.add(role);
		manager.setRole(roles);


manager.setUserpassword(getEncodedPassword(manager.getUserpassword()));

		return userdao.save(manager);
	}

	public void initRolesAndUser() {
		Role adminRole = new Role();
		adminRole.setRoleName("Admin");
		adminRole.setRoleDescription("Administrative role for the website");
		roledao.save(adminRole);

		Role customerRole = new Role();
		customerRole.setRoleName("Customer");
		customerRole.setRoleDescription("Default role for customers");
		roledao.save(customerRole);

		Role managerRole = new Role();
		managerRole.setRoleName("Manager");
		managerRole.setRoleDescription("Assigning complaints to suitable engineer to
resolve it");
		roledao.save(managerRole);

		Role engineerRole = new Role();
		engineerRole.setRoleName("Engineer");
		engineerRole.setRoleDescription("Resolve the complaints and update the
status");
		roledao.save(engineerRole);

		User adminUser = new User();
		adminUser.setFullName("admin");
		adminUser.setUserName("admin123");
```

```
		adminUser.setUserpassword(getEncodedPassword("pwd@123"));
		Set<Role> adminRoles = new HashSet<>();
		adminRoles.add(adminRole);
		adminUser.setRole(adminRoles);
		userdao.save(adminUser);

		User managerUser = new User();
		managerUser.setFullName("ABC Manager1");
		managerUser.setUserName("manager1");
		managerUser.setUserpassword(getEncodedPassword("manager1@pwd"));;
		Set<Role> managerRoles = new HashSet<>();
		managerRoles.add(managerRole);
		managerUser.setRole(managerRoles);
		userdao.save(managerUser);
	}

	public String getEncodedPassword(String password) {
		return passwordEncoder.encode(password);
	}
}

/*
 * adminUser.setUserpassword("pwd@123");
 * This has to store encrypted password*/
```

JwtUtil.java

```
package com.simplilearn.finalphase2.util;

import java.util.Date;

import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

@Component
public class JwtUtil {
```

```
	private static final String SECRET_KEY = "my_movie_plan";
	private static final int TOKEN_VALIDITY = 3600 * 5;

	public String getUserNameFromToken(String token) {

		return getClaimFromToken(token,Claims::getSubject);

	}

	private <T> T getClaimFromToken(String token,Function<Claims, T> claimResolver) {

		final Claims claims = getAllClaimsFromToken(token);
		return claimResolver.apply(claims);
	}

	private Claims getAllClaimsFromToken(String token) {
		return
Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token).getBody();
	}

	public boolean ValidateToken(String token,UserDetails userDetails) {
		String userName = getUserNameFromToken(token);
		return (userName.equals(userDetails.getUsername()) &&
!isTokenExpired(token));
	}

	private boolean isTokenExpired(String token) {
		final Date expDate = getExpirationDateFromToken(token);
		return expDate.before(new Date());
	}


	private Date getExpirationDateFromToken(String token) {
		return getClaimFromToken(token,Claims::getExpiration);
	}

	public String generateToken(UserDetails userDetails) {
		Map<String, Object> claims = new HashMap<>();

		return Jwts.builder()
				.setClaims(claims)
				.setSubject(userDetails.getUsername())
				.setIssuedAt(new Date(System.currentTimeMillis()))
```

```
                                .setExpiration(new Date(System.currentTimeMillis()+
TOKEN_VALIDITY * 1000))
                                .signWith(SignatureAlgorithm.HS512, SECRET_KEY)
                                .compact();


        }
}


/*
 * secret key has to be pass from this
 * return Jwts.parser().setSigningKey(null)
 * that can secret can be hard coded...fro better security it can be stored in db.
 *
 * getClaimFromToken--> higher order function of java as take a function as argument
 * all the snippet is the part the functional programming
 *
 * return expDate.before(new Date());
 * If expiration date is not before the current date this will false and the token is not expired in
that case
*/
```

application.properties

```
#configure database
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/complaint_redressal_db1
spring.datasource.username=root
spring.datasource.password=Shivavalli@1
#spring.datasource.platform=mysql
spring.jpa.hibernate.ddl-auto = update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
spring.main.allow-circular-references=true
#above line will allow the circular references to different beans
#spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAuto
Configuration
spring.jpa.generate-ddl=true
#Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.properties.hibernate.format_sql=true
```

Front-End

Admin-home.component.html

```
<div class="container">
    <div class="card p-5 text-center">
        <h1>Welcome Admin!!</h1>
    </div>
</div>
```

Admin-home.component.ts

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

@Component({
  selector: 'app-admin-home',
  templateUrl: './admin-home.component.html',
  styleUrls: ['./admin-home.component.css']
})
export class AdminHomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

All-complaints.component.html

```
<div class="row">
    <div class="col-md-12">


            <div class="card-header"><h1>Complaint Details:</h1></div>


        <div class="card-body" style="background-color: aliceblue;">
            <table class="table table-info">
                <tr style="text-align: center;">
```

```
                    <th style="padding: 5px;">ID</th>
                    <th style="padding: 5px;">Full Name</th>
                    <th style="padding: 5px;">Heading</th>
                    <th style="padding: 5px;">Details</th>
                    <!-- <th style="padding: 5px;">Director</th>
                    <th style="padding: 5px;">Language</th> -->
                    <th style="padding: 10px;">Address</th>
                    <th style="padding: 5px;">Pincode</th>
                    <th style="padding: 5px;">complaintStatus</th>
                </tr>

                <tr style="text-align: center;" *ngFor="let c of
complaintdetails">
                    <td style="padding: 5px;">{{c.complaintId}}</td>
                    <td style="padding: 5px;">{{c.fullname}}</td>
                    <td style="padding: 5px;">{{c.heading}}</td>
                    <td style="padding: 5px;">{{c.details}}</td>
                    <td style="padding: 5px;">{{c.address}}</td>
                    <!-- <td style="padding: 5px;">{{mt.director}}</td>
                    <td style="padding: 5px;">{{mt.language}}</td> -->
                    <td style="padding: 5px;">{{c.pincode}}</td>
                    <td style="padding: 5px;">{{c.complaintStatus}}</td>
                </tr>
            </table>

        </div>
    </div>
</div>
```

All-complaints.component.ts

```
import { HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { map } from 'rxjs/operators';
import { Complaint } from '../complaint';
import { ComplaintService } from '../complaint.service';

@Component({
  selector: 'app-allcomplaints',
```

```
  templateUrl: './allcomplaints.component.html',
  styleUrls: ['./allcomplaints.component.css']
})
export class AllcomplaintsComponent implements OnInit {

  pageNumber:number = 0;
  complaintdetails = [];
  constructor(
    private complaintService:ComplaintService,
    private router:Router
  ) { }

  ngOnInit(): void {
    this.getAllComplaints();
  }

  getAllComplaints()
  {
    this.complaintService.getComplaintList()
    .subscribe(
      (resp: Complaint[]) => {
        console.log(resp);
        resp.forEach(c => this.complaintdetails.push(c));
      },
      (error: HttpErrorResponse) => {
        console.log(error);
      }
    );
  }

}
```

All Complaints manager.html

```
<div class="row">
    <div class="col-md-12">


            <div class="card-header"><h1>Complaint Details:</h1></div>
```

```
        <div class="card-body" style="background-color: aliceblue;">
            <table class="table table-info">
                <tr style="text-align: center;">
                    <th style="padding: 5px;">ID</th>
                    <th style="padding: 5px;">Full Name</th>
                    <th style="padding: 5px;">Heading</th>
                    <th style="padding: 5px;">Details</th>
                    <!-- <th style="padding: 5px;">Director</th>
                    <th style="padding: 5px;">Language</th> -->
                    <th style="padding: 10px;">Address</th>
                    <th style="padding: 5px;">Pincode</th>
                    <th style="padding: 5px;">Assigned To</th>
                    <th style="padding: 5px;">complaintStatus</th>
                </tr>

                <tr style="text-align: center;" *ngFor="let c of
complaintdetails">
                    <td style="padding: 5px;">{{c.complaintId}}</td>
                    <td style="padding: 5px;">{{c.fullname}}</td>
                    <td style="padding: 5px;">{{c.heading}}</td>
                    <td style="padding: 5px;">{{c.details}}</td>
                    <td style="padding: 5px;">{{c.address}}</td>
                    <!-- <td style="padding: 5px;">{{mt.director}}</td>
                    <td style="padding: 5px;">{{mt.language}}</td> -->
                    <td style="padding: 5px;">{{c.pincode}}</td>
                    <td style="padding: 5px;" *ngIf="c.pincode ===
'110025'">Ankita Soni</td>
                    <td style="padding: 5px;" *ngIf="c.pincode ===
'110021'">Deepak Shrivastav</td>
                    <td style="padding: 5px;" *ngIf="c.pincode ===
'110022'">Asima Tirkey</td>
                    <td style="padding: 5px;">{{c.complaintStatus}}</td>
                </tr>
            </table>

        </div>
    </div>
</div>
```

All Complaints Manager.ts:

```
import { HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Complaint } from '../complaint';
import { ComplaintService } from '../complaint.service';

@Component({
  selector: 'app-allcomplaints-manager',
  templateUrl: './allcomplaints-manager.component.html',
  styleUrls: ['./allcomplaints-manager.component.css']
})
export class AllcomplaintsManagerComponent implements OnInit {

  asignedTo:string;
  complaintdetails:any|Complaint[];
  constructor(
    private complaintService:ComplaintService,
    private router:Router
  ) { }

  ngOnInit(): void {
    this.getAllComplaints();

  }

  getAllComplaints()
  {
    this.complaintService.getComplaintList()
    .subscribe(
      (resp: Complaint[]) => {
        console.log(resp);
        resp.forEach(c => this.complaintdetails.push(c));
      },
      (error: HttpErrorResponse) => {
        console.log(error);
      }
    );
  }
```

```
  wip(complaintId){
    this.complaintService.markAsWip(complaintId);
    this.getAllComplaints();
  }
  Reviewed(complaintId){
    this.complaintService.markAsInReview(complaintId);
    this.getAllComplaints();
  }
  Resolved(complaintId){
    this.complaintService.markAsResolved(complaintId);
    this.getAllComplaints();
  }
}
```

Complaint Engineer.html:

```
<div class="row">
    <div class="col-md-12">

        <div class="card-header">
            <h1>Complaint Details:</h1>
        </div>



        <div class="card-body" style="background-color: aliceblue;">
            <table class="table table-info">
                <tr style="text-align: center;">
                    <th style="padding: 5px;">ID</th>
                    <th style="padding: 5px;">Full Name</th>
                    <th style="padding: 5px;">Heading</th>
                    <th style="padding: 5px;">Details</th>
                    <!-- <th style="padding: 5px;">Director</th>
                    <th style="padding: 5px;">Language</th> -->
                    <th style="padding: 10px;">Address</th>
                    <th style="padding: 5px;">Pincode</th>
                    <th style="padding: 5px;">complaintStatus</th>
                    <th style="padding: 5px;">Operation</th>
                </tr>
```

```
                <tr style="text-align: center;" *ngFor="let c of
complaintdetails">
                    <ng-container *ngIf="c.pincode === '110025'">
                        <td style="padding: 5px;">{{c.complaintId}}</td>
                        <td style="padding: 5px;">{{c.fullname}}</td>
                        <td style="padding: 5px;">{{c.heading}}</td>
                        <td style="padding: 5px;">{{c.details}}</td>
                        <td style="padding: 5px;">{{c.address}}</td>
                        <!-- <td style="padding:
5px;">{{mt.director}}</td>
        <td style="padding: 5px;">{{mt.language}}</td> -->
                        <td style="padding: 5px;">{{c.pincode}}</td>
                        <td style="padding:
5px;">{{c.complaintStatus}}</td>
                        <td style="padding: 5px;">
                            <button mat-raised-button color="primary"
class="mb-2"
                                (click)="wip(c.complaintId)">WIP</button>
                            <button mat-raised-button color="primary"
class="mb-2"

(click)="Reviewed(c.complaintId)">Reviewed</button>
                            <button mat-raised-button color="primary"
class="ml-2"

(click)="Resolved(c.complaintId)">Resolved</button>
                        </td>

                    </ng-container>
                </tr>


            </table>

        </div>
    </div>
</div>
```

Complaint Engineer.ts:

```
import { HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { Complaint } from '../complaint';
import { ComplaintService } from '../complaint.service';

@Component({
  selector: 'app-complaint-engineer',
  templateUrl: './complaint-engineer.component.html',
  styleUrls: ['./complaint-engineer.component.css']
})
export class ComplaintEngineerComponent implements OnInit {

  complaintdetails = [];
  constructor(
    private complaintService:ComplaintService
  ) { }

  ngOnInit(): void {
    this.getAllComplaints();
  }

  getAllComplaints()
  {
    this.complaintService.getComplaintList()
    .subscribe(
      (resp: Complaint[]) => {
        console.log(resp);
        resp.forEach(c => this.complaintdetails.push(c));
      },
      (error: HttpErrorResponse) => {
        console.log(error);
      }
    );
  }

  wip(complaintId){
    this.complaintService.markAsWip(complaintId);
    this.getAllComplaints();
  }
```

```
  Reviewed(complaintId){
    this.complaintService.markAsInReview(complaintId);
    this.getAllComplaints();
  }
  Resolved(complaintId){
    this.complaintService.markAsResolved(complaintId);
    this.getAllComplaints();
  }

}
```

Complaint Register.html:

```
<div class="container mt-5">
    <div class="card p-5" style="background-color:#012169;">
        <div class="row">
            <div class="col-12">
        <form #comaplaintForm="ngForm"
(ngSubmit)="registerComplaint(comaplaintForm)" >

            <mat-form-field appearance="outline" class="w-100">
                <mat-label>Full Name: </mat-label>
                <input matInput placeholder="fullname"
[(ngModel)]="complaintDetails.fullname" name="fullname" style="color:
whitesmoke;">
            </mat-form-field>

            <!--Enter complaint category-->
            <!-- <mat-form-field appearance="fill">
                <mat-label>Category</mat-label>
                <mat-select [formControl]="Category" multiple>
                  <mat-select-trigger>
                    {{category.value?.[0] || ''}}
                    <span *ngIf="(category.value?.length || 0) > 1"
class="example-additional-selection">
                      (+{{(category.value?.length || 0) - 1}}
{{category.value?.length === 2 ? 'other' : 'others'}})
                    </span>
                  </mat-select-trigger>
                  <mat-option *ngFor="let topping of toppingList"
[value]="topping">{{category}}</mat-option>
```

```
                </mat-select>
              </mat-form-field>
               -->


            <mat-form-field appearance="outline" class="w-100">
                <mat-label>Complaint Title: </mat-label>
                <input matInput placeholder="Complaint Title"
[(ngModel)]="complaintDetails.heading" name="heading" style="color:
whitesmoke;">
            </mat-form-field>

            <mat-form-field appearance="outline" class="w-100">
                <mat-label>Complaint Details: </mat-label>
                <textarea  type="text" matInput placeholder="Complaint
Details" [(ngModel)]="complaintDetails.details" name="details"
style="color: whitesmoke;"></textarea>
            </mat-form-field>

            <mat-form-field appearance="outline" class="w-100">
                <mat-label>Address: </mat-label>
                <input  matInput placeholder="Address"
[(ngModel)]="complaintDetails.address" name="address" style="color:
whitesmoke;" required>
            </mat-form-field>

            <mat-form-field appearance="outline" class="w-100">
                <mat-label>Pincode: </mat-label>
                <input  matInput placeholder="Pincode"
[(ngModel)]="complaintDetails.pincode" name="pincode" style="color:
whitesmoke;" required>
            </mat-form-field>

            <div align="centre">
                <button type="submit" mat-raised-button
color="accent">Register Complaint</button>
             </div>
        </form>
    </div>
</div>
```

```
    </div>
</div>
```

Complaint Register Component.ts:

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { ActivatedRoute, Router } from '@angular/router';
import { Category } from '../category.model';
import { Complaint } from '../complaint';
import { ComplaintService } from '../complaint.service';
import { Role } from '../role';
import { User } from '../user';


@Component({
  selector: 'app-complaint-register',
  templateUrl: './complaint-register.component.html',
  styleUrls: ['./complaint-register.component.css']
})

export class ComplaintRegisterComponent implements OnInit {

  userDetails: any| User[];
  customer:User;
  role:Role
  category: Category[];

  complaintDetails: Complaint = {
    category: '',
    heading: '',
    details: '',
    address: '',
    pincode: '',
    fullname: '',
    complaintStatus: '',
    customer: new User
  }

  constructor(
```

```
    private complaintService:ComplaintService,
    private router:Router,
    private activatedRoute:ActivatedRoute
  ) { }

  ngOnInit(): void {
    // this.userDetails =
this.activatedRoute.snapshot.data['complaintDetails'];

    // this.userDetails.forEach(
    //   x =>this.complaintDetails.customer.push(
    //     {complaintId : x.complaintId}//quantity default value given 1
    //     )
    //   );

      console.log(this.userDetails);
      console.log(this.complaintDetails);
  }

  public registerComplaint(comaplaintForm:NgForm){
    console.log(comaplaintForm.value);
    this.complaintService.regNewComplaint(comaplaintForm.value).subscribe(
      (resp) => {
        console.log(resp);
        this.router.navigate(['/customerHome']);
      },
      (error) =>{
        console.log(error);
      }
    )
  }

}
```

Customer Home Component.html:

```
<div class="container">
    <div class="card p-5 text-center">
        <h1>Welcome!!</h1>
        <h2>Any complaints?</h2>
```

```
        <a routerLink="/complaintReg" style="color: red;"><h2>Raise your
complaints here...</h2></a>
    </div>
</div>
```

Customer Home Component.ts:

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

@Component({
  selector: 'app-customer-home',
  templateUrl: './customer-home.component.html',
  styleUrls: ['./customer-home.component.css']
})
export class CustomerHomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Customer Register.html:

```
<div class="container">
    <div class="card p-20 mt-5 mb-5" style="background-color: #002E4B;">
        <form #userRegister = "ngForm" (ngSubmit)="register(userRegister)"
>

            <div class="p-5">

                <mat-form-field appearance="outline" class="w-100 p-2" >
                    <mat-label >
                        User Full Name:
                    </mat-label>
                    <input matInput ngModel name="fullName" id="fullName"
placeholder="User Full Name" style="color: whitesmoke;">
                    <!-- <mat-hint>Full Name</mat-hint> -->
                </mat-form-field>

                <mat-form-field appearance="outline" class="w-100 p-2">
```

```
                    <mat-label>
                        Username
                    </mat-label>
                    <input matInput ngModel name="userName" id="userName"
placeholder="userName" required style="color: whitesmoke;">
                    <!-- <mat-hint>Username</mat-hint> -->
                </mat-form-field>

                <mat-form-field appearance="outline" class="w-100 p-2">
                    <mat-label>
                        Password
                    </mat-label>
                    <input type="password" matInput ngModel
name="userpassword" id="userpassword" placeholder="userpassword" required
style="color: whitesmoke;">
                    <!-- <mat-hint>Password</mat-hint> -->
                </mat-form-field>

            </div>

            <div class="text-center">
                <button type="submit" mat-raised-button color="primary"
class="mt-2 mb-5 w-50" >
                    Register
                </button>
            </div>

        </form>
    </div>
</div>
```

Customer Register.ts:

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';
import { UserService } from '../user.service';

@Component({
  selector: 'app-customer-reg',
  templateUrl: './customer-reg.component.html',
```

```
  styleUrls: ['./customer-reg.component.css']
})
export class CustomerRegComponent implements OnInit {

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

  ngOnInit(): void {
  }

  register(userRegister:NgForm){
    console.log(userRegister.value);
    this.userservice.customerRegister(userRegister.value).subscribe(
      (response) => {
        console.log(response);
        this.router.navigate(['/login']);
      },
      (error) => {
        console.log(error);
      }
    );
  }

}
```

Engineer Home Component.html:

```
<div class="container">
    <div class="card p-5 text-center">
        <h1>Welcome Ankita!!</h1>
        <h2>Check Complaints Assigned to you...</h2>

    </div>
</div>
```

Engineer Home Component.ts:

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

@Component({
```

```
  selector: 'app-engineer-home',
  templateUrl: './engineer-home.component.html',
  styleUrls: ['./engineer-home.component.css']
})
export class EngineerHomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Header Component.html:

```
<mat-toolbar color="primary">
    <mat-toolbar-row >
        <span ><img src="/assets/telecom-icon2.jpg" style="width: 80px;
height: 60px; cursor: pointer;" class="img-fluid rounded-lg"
routerLink=""></span>
      <span class="ml-3"routerLink="" style="cursor: pointer;color:
black;" >ABC Telecom</span>
      <span class="example-spacer"></span>
      <button mat-raised-button color="accent" *ngIf="! isLoggedIn()"
routerLink="/login">Login</button>
      <button mat-raised-button color="warn" class="ml-2"
*ngIf="isLoggedIn()" (click)="logout()">Logout</button>
    </mat-toolbar-row>


      <mat-toolbar-row color="primary" >
        <button mat-raised-button color="accent" *ngIf="isCustomer()"
routerLink="/customerHome">Dashboard</button>
        <button mat-raised-button color="accent" *ngIf="isCustomer()"
routerLink="/complaintReg" class="ml-2">Register Complaint</button>
        <button mat-raised-button color="accent" *ngIf="isCustomer()"
routerLink="/mycomplaints" class="ml-2">Previous complaints</button>
        <button mat-raised-button color="accent" *ngIf="isAdmin()"
routerLink="/adminHome">Dashboard</button>
        <button mat-raised-button color="accent" *ngIf="isAdmin()"
routerLink="/allcomplaints" class="ml-2" >All complaints</button>
```

```
        <button mat-raised-button color="accent" *ngIf="isAdmin()"
routerLink="/userList" class="ml-2">All Users</button>
        <button mat-raised-button color="accent" *ngIf="isManager()"
routerLink="/managerHome">Dashboard</button>
        <button mat-raised-button color="accent" *ngIf="isManager()"
routerLink="/allComplaintsManager" class="ml-2" >All complaints</button>
        <button mat-raised-button color="accent" *ngIf="isEngineer()"
routerLink="/complaintEngg" class="ml-2">Assigned Complaints</button>
      </mat-toolbar-row>
      </mat-toolbar>
```

Header Component.ts:

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { UserAuthService } from '../user-auth.service';
import { UserService } from '../user.service';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent implements OnInit {

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

  ngOnInit(): void {
  }

  public isLoggedIn(){
    return this.userAuthService.isLoggedIn();
  }

  public logout(){
    this.userAuthService.clear();
    this.router.navigate(['']);
```

```
  }

  public isAdmin(){
   return this.userAuthService.isAdmin();
  }

  public isCustomer(){
    return this.userAuthService.isCustomer();
  }

  public isManager(){
    return this.userAuthService.isManager();
  }

  public isEngineer(){
    return this.userAuthService.isEngineer();
  }

}
```

Home Component.html:

```
<div class="container">
    <p class="mt-3">
        ABC Telecom is India's foremost and truly-integrated
telecommunications service provider, with a corporate customer base of
over 40,000 Indian and multinational corporations, including small and
medium enterprises, and close to 300 of the finest enterprise and carrier
companies globally.
    </p>

<p class="mt-2">
    ABC Telecom touches 90 per cent of the country's population, supported
by an OFC network of over 190,000 km. The Company also has a significant
tower infrastructure that caters to other telecommunications operators.
</p>

<p class="mt-2" >
    Spread across India's top 29 cities, ABC Telecom' data solutions for
Homes and Small Enterprises cater to buildings connected to the Reliance
network. Reliance Communications also owns the world’s largest private
```

under-sea cable system under its subsidiary Global Cloud Xchange, which offers a comprehensive portfolio of solutions customized for Carriers, Enterprises and New Media Companies globally.

    GCX's under-sea cable system spans more than 67,000 route km, which, seamlessly integrated with Reliance Communications' 200,000 route km of domestic optic fiber backbone, provides a robust Global Service Delivery Platform for Enterprises. GCX is uniquely equipped to support businesses through the deployment of Next-Generation Enterprise solutions across its Cloud Delivery Networks. Since its inception in December 2002, Reliance Communications has been revolutionizing the way the world communicates, impacting not just lifestyles, but the very lives of its customers-globally

For further information, please visit: www.abctele.co.in

```
<div class="row mt-2">
    <div class="column">
      <a><img src="/assets/telecom-1.jpg" alt="Snow"
style="width:100%;height: 200px;"></a>
      <p class="text-center"><b>Career</b></p>
    </div>
    <div class="column">
      <a><img src="/assets/telecom-2.jpg" alt="Forest"
style="width:100%;height: 200px;"></a>
      <p class="text-center"><b>Our Offices</b></p>
    </div>
    <div class="column">
        <a><img src="/assets/telecom-3.jpg" alt="Mountains"
style="width:100%;height: 200px;"></a>
      <p class="text-center"><b>Our Vision</b></p>
    </div>
  </div>

</div>
```

Home Component.ts:

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Login Component.html:

```
<div class="container mt-5">
    <div clas="card" style="padding: 50px;padding-right: 300px;
padding-left: 300px;background-color:#F0FFFF">

        <form  #loginForm="ngForm" (ngSubmit)="login(loginForm)">
            <div class="mb-2 text-center" style="text-align:
center;color:red;font-style: italic;font-size: larger;">
                {{msg}}
            </div>
            <input type="text" ngModel name="userName" id="userName"
placeholder="Enter username"
               style="padding: 15px;" class="form-control mb-4 ml-5 ml-5">
            <input type="password" ngModel name="userPassword"
id="userPassword" placeholder="Enter username" style="padding: 15px;"
                class="form-control mb-4 ml-5 mr-5">
                <div class=" ml-5 mr-5"  style="padding:
15px;background-color: #F0FFFF">
                    <input type="submit" value="Login" class="btn
btn-outline-primary form-control rounded-2 ml-10 mr-10 mb-2 " />
                    <input type="reset" value="Reset" class="btn
btn-outline-warning form-control rounded-2 ml-10 mr-10" />
                </div>
```

```
        </form>
    </div>
    <!-- outside card -->
    <div class="text-center">
        <button mat-raised-button type="button" color="primary"
class="mt-2 mb-5" (click)="registerUser()" >Create customer
account</button>
    </div>

</div>

<!-- input type="text" name id will be to keep same as mentioned in
backend class jwtRequest/(The class which responsible to take username and
password for further backend processing) to make the integration automatic
-->
```

Login Component.ts:

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';
import { UserAuthService } from '../user-auth.service';
import { UserService } from '../user.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  msg:any;
  constructor(
    private userService:UserService,
    private userAuthSrvice:UserAuthService,
    private router:Router
  ) { }

  ngOnInit(): void {
  }
  login(loginForm:NgForm){
```

```
    this.userService.login(loginForm.value).subscribe((response:any)=>{
      console.log(response);//this response contains the entered userdata
      // console.log(response.jwtToken);
      // console.log(response.user.role);
      this.userAuthSrvice.setRoles(response.user.role);
      this.userAuthSrvice.setToken(response.jwtToken);

      const role = response.user.role[0].roleName;//the value of role=
Admin or User is present inside
      //user-->role-->roleName
      if(role =='Admin'){
        this.router.navigate(['/adminHome']);
      }
      else if(role =='Customer'){
        this.router.navigate(['/customerHome']);
      }
      else if(role =='Manager'){
        this.router.navigate(['/managerHome']);
      }
      else if(role =='Engineer'){
        this.router.navigate(['/engineerHome']);
      }
      else{
        this.msg = "Entered Credentials are incorrect";
      }
    },
    (error)=>{
      this.msg = "Entered Credentials are incorrect, Please try again!!";
      console.log(error);
    });
    console.log("Form is submitted");
    // console.log(loginForm.value); //This will print data values input
in the form and print the console
  }

  registerUser(){
    this.router.navigate(['/customerReg']);
  }

}
```

Manager Home Component.html:

```
<div class="container">
    <div class="card p-5 text-center">
        <h1>Welcome Manager!!</h1>
        <h2>Assign complaints </h2>

    </div>
</div>
```

Manager Home Component.ts:

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

@Component({
  selector: 'app-manager-home',
  templateUrl: './manager-home.component.html',
  styleUrls: ['./manager-home.component.css']
})
export class ManagerHomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

My Complaints Component.html:

```
<div class="container mt-5 mb-2">
    <div class="row">
        <div class="col-md-12">
            <div class="card"style="background-color: aliceblue;">
                <div class="card-header"><h1>Complaint Details:</h1></div>
            </div>
            <div class="card-body" style="background-color: aliceblue;">
                <table class="table table-info">
                    <tr style="text-align: center;">
                        <th style="padding: 5px;">Heading</th>
                        <th style="padding: 5px;">Details</th>
```

```
                        <!-- <th style="padding: 5px;">Movie Summary</th>
-->
                        <th style="padding: 5px;">Address</th>
                        <th style="padding: 10px;">Pincode</th>
                        <th style="padding: 5px;">Status</th>

                    </tr>

                    <tr  *ngFor="let c of mycomplaints" style="text-align:
center;">
                        <ng-container *ngIf="c.pincode === '110025'">
                        <td style="padding: 8px;">{{c.heading}}</td>
                        <td style="padding: 8px;">{{c.details}}</td>
                        <!-- <td style="padding:
8px;">{{c.movieTicket.description}}</td> -->
                        <td style="padding: 8px;">{{c.address}}</td>
                        <td style="padding: 8px;">{{c.pincode}}</td>
                        <td style="padding:
8px;">{{c.complaintStatus}}</td>
                        </ng-container>
                    </tr>
                </table>
            </div>
        </div>
    </div>
</div>
```

My Complaints Component.ts:

```
import { Component, OnInit } from '@angular/core';
import { Complaint } from '../complaint';
import { ComplaintService } from '../complaint.service';

@Component({
  selector: 'app-mycomplaints',
  templateUrl: './mycomplaints.component.html',
  styleUrls: ['./mycomplaints.component.css']
})
export class MycomplaintsComponent implements OnInit {
```

```
  mycomplaints:Complaint[];
  constructor(
    private complaintService:ComplaintService
  ) { }

  ngOnInit(): void {
    this.getMyComplaints();
  }

  getMyComplaints(){
    this.complaintService.getComplaintList().subscribe(
      (resp: Complaint[]) => {
        console.log(resp);
        this.mycomplaints = resp;
      },
      (err) => {
        console.log(err);
        }

    )

  }

}
```

User List Component .html:

```
<div class="row">
    <div class="col-md-12">

        <div class="card"style="background-color: aliceblue;">
            <div class="card-header"><h1>List of active users:</h1></div>
        </div>
        <div class="card-body" style="background-color: aliceblue;">
            <table class="table table-info">
                <tr style="text-align: center;">
                    <th style="padding: 5px;">Full Name</th>
                    <th style="padding: 5px;">Username</th>
                    <!-- <th style="padding: 5px;">Role</th> -->
                    <!-- <th style="padding: 5px;">Role Description</th>
-->
```

```
                    <th style="padding: 5px;">Password</th>
                    <th style="padding: 5px;">Delete</th>
                </tr>

                <tr style="text-align: center;" *ngFor="let u of
userinfo">
                    <td style="padding: 5px;">{{u.fullName}}</td>
                    <td style="padding: 5px;">{{u.userName}}</td>
                    <!-- <td style="padding: 5px;">{{u.roleName}}</td> -->
                    <!-- <td style="padding:
5px;">{{u.role.roleDescription}}</td>  -->
                    <td style="padding: 5px;">{{u.userpassword}}</td>
                    <td>
                      <button mat-icon-button aria-label="Delete icon"
(click)="deleteUser(u.userName)"  color="warn">
                        <mat-icon>delete</mat-icon>
                      </button>
                    </td>
                </tr>
            </table>
        </div>
    </div>
</div>
```

User List Component.ts:

```
import { HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { Role } from '../role';
import { User } from '../user';
import { UserService } from '../user.service';

@Component({
  selector: 'app-user-list',
  templateUrl: './user-list.component.html',
  styleUrls: ['./user-list.component.css']
})
export class UserListComponent implements OnInit {

  userinfo: any|Role| User[];
```

```
  constructor(
    private userService:UserService,
  ) { }

  ngOnInit(): void {
    this.getAllUsers();
  }
  getAllUsers(){
    this.userService.getAllUsers().subscribe(
      (resp) =>{
        this.userinfo = resp;
        console.log(resp);
      },
      (err) => {
        console.log(err);
      }
    //   data=>{
    //   console.log(data);
    //   this.userinfo = data;
    // }
    );
  }

  public deleteUser(userName){
    console.log(userName);
    this.userService.deleteUser(userName).subscribe((resp)=>{
      console.log(resp);
      this.getAllUsers();
    },
    (error:HttpErrorResponse)=>{
      console.log(error);
    })
  }

}
```

App Routing module.ts:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AdminHomeComponent } from './admin-home/admin-home.component';
```

```
import { AllcomplaintsManagerComponent } from
'./allcomplaints-manager/allcomplaints-manager.component';
import { AllcomplaintsComponent } from
'./allcomplaints/allcomplaints.component';
import { AuthGuard } from './auth.guard';
import { ComplaintEngineerComponent } from
'./complaint-engineer/complaint-engineer.component';
import { ComplaintRegisterComponent } from
'./complaint-register/complaint-register.component';
import { ComplaintResolverService } from './complaint-resolver.service';
import { CustomerHomeComponent } from
'./customer-home/customer-home.component';
import { CustomerRegComponent } from
'./customer-reg/customer-reg.component';
import { EngineerHomeComponent } from
'./engineer-home/engineer-home.component';
import { HomeComponent } from './home/home.component';
import { LoginComponent } from './login/login.component';
import { ManagerHomeComponent } from
'./manager-home/manager-home.component';
import { MycomplaintsComponent } from
'./mycomplaints/mycomplaints.component';
import { UserListComponent } from './user-list/user-list.component';

const routes: Routes = [
  {path:'',component:HomeComponent},
  {path:'login',component:LoginComponent},
  {path:'customerReg',component:CustomerRegComponent},
  {path:'adminHome',component:AdminHomeComponent,canActivate:[AuthGuard],
data:{roles:['Admin']}},

{path:'engineerHome',component:EngineerHomeComponent,canActivate:[AuthGuar
d], data:{roles:['Engineer']}},

{path:'managerHome',component:ManagerHomeComponent,canActivate:[AuthGuard]
, data:{roles:['Manager']}},

{path:'customerHome',component:CustomerHomeComponent,canActivate:[AuthGuar
d], data:{roles:['Customer']}},
```

```
{path:'complaintReg',component:ComplaintRegisterComponent,canActivate:[AuthGuard], data:{roles:['Customer']}},

{path:'mycomplaints',component:MycomplaintsComponent,canActivate:[AuthGuard], data:{roles:['Customer']} },

{path:'allcomplaints',component:AllcomplaintsComponent,canActivate:[AuthGuard], data:{roles:['Admin']}},

{path:'userList',component:UserListComponent,canActivate:[AuthGuard],data:{roles:['Admin']}},

{path:'allComplaintsManager',component:AllcomplaintsManagerComponent,canActivate:[AuthGuard], data:{roles:['Manager']}},

{path:'complaintEngg',component:ComplaintEngineerComponent,canActivate:[AuthGuard], data:{roles:['Engineer']}}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

App Component.html:

```
<div class="bg">

  <div>
    <app-header></app-header>
  </div>
  <div >
    <router-outlet></router-outlet>
  </div>

  <footer class="bg-dark text-center text-white">
    <!-- Copyright -->
    <div class="text-center p-3" style="background-color:
rgb(169,169,169)">
```

```
      © 2002 Copyright: ABC Telecom || Developed By: Khushboo Sharma ||
FSD JG Nov 2022 Cohort 3
      <!-- <a class="text-white"
href="https://mdbootstrap.com/">MDBootstrap.com</a> -->
    </div>
    <!-- Copyright -->
  </footer>

</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 = 'ComplaintRedressal';
}
```

App Module.ts:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { RouterModule } from '@angular/router';
import{MatToolbarModule} from '@angular/material/toolbar';
import {MatButtonModule} from '@angular/material/button';
import {MatFormFieldModule} from '@angular/material/form-field';
import {MatInputModule} from '@angular/material/input';
import {MatGridListModule} from '@angular/material/grid-list';
import {MatTableModule} from '@angular/material/table';
import {MatIconModule} from '@angular/material/icon';
import {MatDialogModule} from '@angular/material/dialog';
import {MatButtonToggleModule} from '@angular/material/button-toggle';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
```

```
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
import { BrowserAnimationsModule } from
'@angular/platform-browser/animations';
import { HttpClientModule } from '@angular/common/http';
import { HomeComponent } from './home/home.component';
import { LoginComponent } from './login/login.component';
import { HeaderComponent } from './header/header.component';
import { AdminHomeComponent } from './admin-home/admin-home.component';
import { EngineerHomeComponent } from
'./engineer-home/engineer-home.component';
import { ManagerHomeComponent } from
'./manager-home/manager-home.component';
import { CustomerHomeComponent } from
'./customer-home/customer-home.component';
import { ComplaintRegisterComponent } from
'./complaint-register/complaint-register.component';
import { AuthGuard } from './auth.guard';
import { HTTP_INTERCEPTORS } from '@angular/common/http';
import { AuthInterceptor } from './auth.interceptor';
import { CustomerRegComponent } from
'./customer-reg/customer-reg.component';
import { MycomplaintsComponent } from
'./mycomplaints/mycomplaints.component';
import { AllcomplaintsComponent } from
'./allcomplaints/allcomplaints.component';
import { UserListComponent } from './user-list/user-list.component';
import { AllcomplaintsManagerComponent } from
'./allcomplaints-manager/allcomplaints-manager.component';
import { ComplaintEngineerComponent } from
'./complaint-engineer/complaint-engineer.component';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    LoginComponent,
    HeaderComponent,
    AdminHomeComponent,
    EngineerHomeComponent,
    ManagerHomeComponent,
```

```
    CustomerHomeComponent,
    ComplaintRegisterComponent,
    CustomerRegComponent,
    MycomplaintsComponent,
    AllcomplaintsComponent,
    UserListComponent,
    AllcomplaintsManagerComponent,
    ComplaintEngineerComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    NgbModule,
    BrowserAnimationsModule,
    FormsModule,
    MatToolbarModule,
    MatFormFieldModule,
    MatButtonModule,
    MatInputModule,
    MatGridListModule,
    MatTableModule,
    MatIconModule,
    MatDialogModule,
    MatButtonToggleModule,
    RouterModule,
    HttpClientModule
  ],
  providers: [
    AuthGuard,
    {
      provide: HTTP_INTERCEPTORS,
      useClass: AuthInterceptor,
      multi: true
    }
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Auth Guard.ts:

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot,
UrlTree, Router } from '@angular/router';
import { Observable } from 'rxjs';
import { UserAuthService } from './user-auth.service';
import { UserService } from './user.service';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {

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

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> |
Promise<boolean | UrlTree> | boolean | UrlTree {
    if (this.userAuthService.getToken() !== null) {
      const role = route.data['roles'] as Array<string>;

      if (role) {
        const match = this.userService.roleMatch(role);

        if (match) {
          return true;
        } else {
          this.router.navigate(['/forbidden'])
          return false;
        }

      }

    }
```

```
    this.router.navigate(['/login']);
    return false;
  }

}
```

Auth Interceptor.ts:

```
import { HttpErrorResponse, HttpEvent, HttpHandler, HttpInterceptor,
HttpRequest } from "@angular/common/http";
import { Router } from "@angular/router";
import { Observable, throwError } from "rxjs";
import { UserAuthService } from "./user-auth.service";
import { catchError} from 'rxjs/operators';
import { Injectable } from "@angular/core";

@Injectable()
export class AuthInterceptor implements HttpInterceptor{

    constructor(private userAuthService:UserAuthService,
        private router:Router){}

        //this will create a header with jwt token and send it to the
baceknd
    intercept(req: HttpRequest<any>, next: HttpHandler):
Observable<HttpEvent<any>> {
        // throw new Error("Method not implemented.")
        if(req.headers.get('No-Auth') === 'True'){
            return next.handle(req.clone());
        }

        const token = this.userAuthService.getToken();//getToken is ins
user.auth.ts which gets token from local storage and passs it to
interceptor, if we do not enter anything it will send nothing

        //req = this.addToken(req,token); this addToken function will add
token value in the header even if it is nul hence we need to add a null
condition as well.

        if(token){
            req = this.addToken(req,token);
```

```
        }
        //with thid if conditn it will addToken only when some token value
is present.

        console.log(req);// will print req in console

        //through following code we are going to backend taking the token
within
        return next.handle(req).pipe(
            catchError(
                (err:HttpErrorResponse) => {
                    console.log(err.status);
                    if (err.status == 401) {
                        this.router.navigate(['/login'])
                    }else if(err.status == 403){
                        this.router.navigate(['/forbidden']);
                    }
                    return throwError("Something went wrong");
                }
            )
        );
    }

    private addToken(request:HttpRequest<any>, token:string){
        return request.clone(
            {
                setHeaders:{
                    Authorization : `Bearer ${token}`
                }
            }
        );
    }

}
//check whether it contains header like NO_AUTH --> we will not add
jwtToken and it will return as it is
```

Category Model.ts:

```
import { Complaint } from "./complaint";

export class Category{
```

```
    BroadBand:string;
    PrePaid:string;
    PostPaid:string;
    PlansDetails:string;
    complaint:Complaint;
}
```

Complaint Resolver Service.ts:

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, Resolve, RouterStateSnapshot } from
'@angular/router';
import { Observable, of } from 'rxjs';
import { map } from 'rxjs/operators';
import { Complaint } from './complaint';
import { ComplaintService } from './complaint.service';


@Injectable({
    providedIn: 'root'
  })

  export class ComplaintResolverService implements Resolve<Complaint>{

    constructor(
        private complaintService:ComplaintService
      ) { }

    resolve(route: ActivatedRouteSnapshot, state:
RouterStateSnapshot):Complaint[]| Observable
<Complaint[]>|Promise<Complaint[]> |any {
        //throw new Error('Method not implemented.');
        const id = route.paramMap.get("id");
        return this.complaintService.getComplaintById(id)
        .pipe(
          map(
            (X : Complaint[], i) => X.map((complaint :  Complaint) =>
(complaint))
          )
        );
      }}
```

Complaint Service.ts:

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Complaint } from './complaint';

@Injectable({
  providedIn: 'root'
})
export class ComplaintService {

  PATH_API = "http://localhost:8080";

  constructor(
    private httpClient:HttpClient
  ) { }

public regNewComplaint(registerData){
  return
this.httpClient.post(this.PATH_API+'/add/new/complaint',registerData);
}

public deleteComplaint(complaintId){
  return
this.httpClient.delete(this.PATH_API+'/delete/complaint/'+complaintId);
}

public getComplaintList():Observable<Complaint[]>{
  return
this.httpClient.get<Complaint[]>(this.PATH_API+'/get/complaint/list');
}

public markAsResolved(complaintId){
  return
this.httpClient.get(this.PATH_API+'/markAsResolved/'+complaintId);
}

public markAsWip(complaintId){
  return this.httpClient.get(this.PATH_API+'/markAsWip/'+complaintId);
}
```

```
public markAsInReview(complaintId){
  return
this.httpClient.get(this.PATH_API+'/markAsInReview/'+complaintId);
}

public getComplaintById(complaintId){
  return
this.httpClient.get(this.PATH_API+'/getComplaintById/'+complaintId);
}

public getMyComplaints(): Observable<Complaint[]>{
  return
this.httpClient.get<Complaint[]>(this.PATH_API+'/get/mycomplaints');
}

public getComplaintListEngg():Observable<Complaint[]>{
  return
this.httpClient.get<Complaint[]>(this.PATH_API+'/get/complaint/list/engine
er');
}

}
```

Complaint.ts:

```
import { User } from "./user";

export class Complaint {

        category: string;
        heading: string;
        details:string;
        address:string;
        pincode:string;
        fullname:string;
        complaintStatus:string;
        customer:User;

}
```

Role.ts:

```
export class Role {

    roleName:string;
    roleDescription:string;
}
```

User Auth Service.ts:

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

@Injectable({
  providedIn: 'root'
})
export class UserAuthService {

  constructor() { }
  public setRoles(roles:[]){
    //array type roles [] has to be convert into string to store it into data storage that can be done by JSON.stringify
    localStorage.setItem("roles", JSON.stringify(roles));
  }

  //localStorage.getItem("roles")-->return string, but to get array type return use JSON.parse
  public getRoles():[]{
    return JSON.parse(localStorage.getItem("roles"));
  }

  public setToken(jwtToken:string){
    localStorage.setItem("jwtToken", jwtToken);
  }

  public getToken() : string{
    return localStorage.getItem("jwtToken");
  }

  public clear(){
    localStorage.clear();
  }
```

```
  public isLoggedIn(){
    return this.getRoles() && this.getToken();
  }

  public isAdmin(){
    const roles: any[] = this.getRoles();
    return roles[0].roleName === 'Admin';
  }

  public isCustomer(){
    const roles: any[] = this.getRoles();
    return roles[0].roleName === 'Customer';
  }

  public isManager(){
    const roles: any[] = this.getRoles();
    return roles[0].roleName === 'Manager';
  }

  public isEngineer(){
    const roles: any[] = this.getRoles();
    return roles[0].roleName === 'Engineer';
  }

}
```

User Service.ts:

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { User } from './user';
import { UserAuthService } from './user-auth.service';

@Injectable({
  providedIn: 'root'
})
export class UserService {

  PATH_API = "http://localhost:8080";
  requestHeader = new HttpHeaders(
    { "No-Auth": "True" }
```

```
  );

  constructor(
    private httpClient: HttpClient,
    private userAuthService: UserAuthService
  ) { }

  public getAllUsers(){
    return this.httpClient.get(this.PATH_API + "/allUser");
  }

  //this.httpClient.post(here url to api be called)
  public login(LoginData) {
    return this.httpClient.post(this.PATH_API + "/authenticate",
LoginData, { headers: this.requestHeader });
  }

  public forCustomer(){
    return this.httpClient.get(this.PATH_API + '/forCustomer',
{responseType:'text'});
  }

  public forAdmin(){
    return this.httpClient.get(this.PATH_API + '/forAdmin',
{responseType:'text'});
  }

  public forManager(){
    return this.httpClient.get(this.PATH_API + '/forManager',
{responseType:'text'});
  }

  public forEngineer(){
    return this.httpClient.get(this.PATH_API + '/forEngineer',
{responseType:'text'});
  }

  //roleMatch(allowedRoles) allowedRoles(roles specified to user in db)-->
will be matched with actual roles-->stored in datastorage if
matched-->return isMatch true
```

```
  public roleMatch(allowedRoles): boolean {
    let isMatch = false;
    const userRoles: any = this.userAuthService.getRoles();

    if (userRoles != null && userRoles) {
      for (let i = 0; i < userRoles.length; i++) {
        for (let j = 0; j < allowedRoles.length; j++) {

          if (userRoles[i].roleName == allowedRoles[j]) {
            isMatch = true;
            return isMatch;
          }
          else {
            return isMatch;
          }

        }
      }
    }

  }

  //for new customer registration open to all
  public customerRegister(registerData){
    return this.httpClient.post(this.PATH_API+'/registerNewCustomer',
registerData);
  }

  //for new engineer registration only allowed fro admin
  public engineerRegister(registerData){
    return this.httpClient.post(this.PATH_API+'/registerNewEngineer',
registerData);
  }

  //for new manager registration only allowed for admin
  public managerRegister(registerData){
    return this.httpClient.post(this.PATH_API+'/registerNewManager',
registerData);
  }
```

```
  public deleteUser(userName:string){
    return this.httpClient.delete(this.PATH_API+'/deleteUser/'+userName);
  }


// public getUserByUserName(userName:string){
//   return this.httpClient.get<User>(this.PATH_API+'/getUserById/'+
userName);
// }

}
```

User.ts:

```
import { Role } from "./role";

export class User {
    userName:string;
    fullName:string;
    userpassword:string;
    role:Role;
}
```

Index.html:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>ComplaintRedressal</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link
href="https://fonts.googleapis.com/css?family=Roboto:300,400,500&display=s
wap" rel="stylesheet">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
</head>
<body class="mat-typography">
  <app-root></app-root>
</body>
</html>
```