Project Code

Project Backend Code

```
package com.crs;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class Phase5ComplaintRedressalSystemApplication {
	public static void main(String[] args) {
		SpringApplication.run(Phase5ComplaintRedressalSystemApplication.class,
args);
	}
}
```

```
package com.crs.config;

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 AuthEntryPoint implements AuthenticationEntryPoint{
```

```
	@Override
	public void commence(HttpServletRequest request, HttpServletResponse response,
				AuthenticationException authException) throws IOException,
ServletException {
		response.sendError(HttpServletResponse.SC_UNAUTHORIZED,
"Unauthorized");
	}

}
```

```
package com.crs.config;

import java.io.IOException;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;

import com.fasterxml.jackson.core.JacksonException;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.DeserializationContext;
import com.fasterxml.jackson.databind.JsonDeserializer;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

public class CustomAuthorityDeserializer extends JsonDeserializer<Object>{

	@Override
```

```
        public Object deserialize(JsonParser jp, DeserializationContext ctxt) throws
IOException, JacksonException {
                ObjectMapper mapper = (ObjectMapper) jp.getCodec();
      JsonNode jsonNode = mapper.readTree(jp);
      List<GrantedAuthority> grantedAuthorities = new LinkedList<>();

      Iterator<JsonNode> elements = jsonNode.elements();
      while (elements.hasNext()) {
         JsonNode next = elements.next();
         JsonNode authority = next.get("authority");
         grantedAuthorities.add(new SimpleGrantedAuthority(authority.asText()));
      }
      return grantedAuthorities;
         }
}
```

```
package com.crs.config;

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.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.crs.service.UserDetailService;

import io.jsonwebtoken.ExpiredJwtException;

@Component
public class JwtAuthFilter extends OncePerRequestFilter{
	@Autowired
	private UserDetailService userDetailService;

	@Autowired
	private JwtUtil jwtUtil;

	@Override
	protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
			throws ServletException, IOException {

		final String requestTokenHeader = request.getHeader("Authorization");
		String username = null;
		String jwtToken = null;

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

			try {
				username = this.jwtUtil.extractUsername(jwtToken);

			}catch(ExpiredJwtException e) {
				e.printStackTrace();
				System.out.println("Token Expired!");
```

```
                }catch(Exception e) {
                        e.printStackTrace();
                }
        }else {
                System.out.println("Invalid token! Not starting from bearer string!");
        }
        // validated


        if(username!=null &&
SecurityContextHolder.getContext().getAuthentication()==null) {
                final UserDetails userDetails =
this.userDetailService.loadUserByUsername(username);
                if(this.jwtUtil.validateToken(jwtToken, userDetails)) {
                        //token is valid
                        UsernamePasswordAuthenticationToken
usernamePasswordAuthenticationToken = new
UsernamePasswordAuthenticationToken(userDetails,null,userDetails.getAuthorities());
                        usernamePasswordAuthenticationToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));

    SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticat
ionToken);
                }
        }else {
                System.out.println("Token is not valid! Please generate a new token!");
        }


        filterChain.doFilter(request, response);


    }


}
```

```
package com.crs.config;

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 String SECRET_KEY = "secret";

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
```

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

    private boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }

    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return createToken(claims, userDetails.getUsername());
    }

    private String createToken(Map<String, Object> claims, String subject) {

        return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new
Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60 * 10))
                .signWith(SignatureAlgorithm.HS256, SECRET_KEY).compact();
    }

    public boolean validateToken(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));
    }

}
```

```
package com.crs.config;
```

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

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.config.annotation.authentication.builders.AuthenticationManag
erBuilder;

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.WebSecurityConfigurerA
dapter;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import com.crs.service.UserDetailService;


@SuppressWarnings("deprecation")

@EnableWebSecurity

@Configuration

public class SecurityConfig extends WebSecurityConfigurerAdapter{

	@Autowired

	private UserDetailService userDetailService;


	@Autowired

	private AuthEntryPoint authEntryPoint;


	@Autowired

	private JwtAuthFilter jwtAuthFilter;
```

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

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

	@Override
	protected void configure(AuthenticationManagerBuilder auth) throws Exception {

	auth.userDetailsService(this.userDetailService).passwordEncoder(passwordEncoder()
);
	}

	@Override
	protected void configure(HttpSecurity http) throws Exception {
		http
			.csrf()
			.disable()
			.cors()
			.disable()
			.authorizeRequests()
			.antMatchers("/generate-token").permitAll()
			.antMatchers(HttpMethod.OPTIONS).permitAll()
			.antMatchers("/user/**").hasAuthority("ADMIN")
			.antMatchers("/customer/**").hasAuthority("CUSTOMER")
```

```
                        .antMatchers("/manager/**").hasAuthority("MANAGER")
                        .antMatchers("/engineer/**").hasAuthority("ENGINEER")
                        .anyRequest().authenticated()
                        .and().exceptionHandling().authenticationEntryPoint(authEntryPoint)

        .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELE
SS);


                http.addFilterBefore(jwtAuthFilter,
UsernamePasswordAuthenticationFilter.class);
        }
}
```

```
package com.crs.controller;

import java.security.Principal;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
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.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
```

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

import com.crs.config.JwtUtil;
import com.crs.entities.JwtRequest;
import com.crs.entities.JwtResponse;
import com.crs.entities.User;
import com.crs.repo.UserRepo;
import com.crs.service.UserDetailService;

@RestController
@CrossOrigin(origins = "*")
public class AuthenticationController {

	@Autowired
	private AuthenticationManager authenticationManager;

	@Autowired
	private UserDetailService userDetailService;

	@Autowired
	private JwtUtil jwtUtil;

	@Autowired
	private UserRepo repo;

	@Autowired
	private BCryptPasswordEncoder passwordEncoder;

	//generate token
```

```
	@PostMapping("/generate-token")
	public ResponseEntity<?> generateToken(@RequestBody JwtRequest jwtRequest) throws Exception{
		try {
			authenticate(jwtRequest.getUsername(), jwtRequest.getPassword());
		}catch(UsernameNotFoundException e) {
			e.printStackTrace();
			throw new Exception("User does not exist!");
		}
		//validated
		UserDetails userDetails = this.userDetailService.loadUserByUsername(jwtRequest.getUsername());
		String token = this.jwtUtil.generateToken(userDetails);
		return ResponseEntity.ok(new JwtResponse(token));
	}

	private void authenticate(String username, String password) throws Exception {
		try {
			this.authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(username, password));
		} catch (BadCredentialsException e) {
			throw new Exception("Invalid Credentials! "+e.getMessage());
		}catch(DisabledException e) {
			throw new Exception("User Disabled! "+e.getMessage());
		}
	}

	//return the details of current user

	@GetMapping("/current-user")
	public User getCurrentUser(Principal principal) {
```

```
		return ((User)this.userDetailService.loadUserByUsername(principal.getName()));
	}

	@PutMapping("/change-password")
	public ResponseEntity<?> changePassword(@RequestBody User user){
		User u = this.repo.findByUsername(user.getUsername());
		if(u!=null) {
			u.setPassword(this.passwordEncoder.encode(user.getPassword()));
			this.repo.save(u);
			return ResponseEntity.status(HttpStatus.CREATED).build();
		}else {
			return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
		}

	}


}
```

```
package com.crs.controller;

import java.text.DateFormat;
import java.util.Calendar;
import java.util.List;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.crs.entities.Complaint;
import com.crs.entities.Feedback;
import com.crs.service.ComplaintService;

@RestController
@CrossOrigin(origins = "*")
@RequestMapping("/customer")
public class CustomerController {

	@Autowired
	private ComplaintService complaintService;

	@PostMapping("/create-complaint")
	public ResponseEntity<Complaint> createComplaint(@Valid @RequestBody
Complaint complaint) throws Exception{
		DateFormat df = DateFormat.getDateInstance();
		Calendar cl = Calendar.getInstance();
		String complaintDate = df.format(cl.getTime());
		complaint.setDate(complaintDate);
		complaint.setStatus("RAISED");
		complaint.setActive(true);
		complaint.setAssigned(false);
		complaint.setRemark("Ticket Raised.");
```

```
		Complaint newComplaint = this.complaintService.createComplaint(complaint);
//		URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(newComplaint.getId()).toUri();
//		return ResponseEntity.created(location).build();
		return ResponseEntity.ok(newComplaint);
	}

	@GetMapping("/get-complaint/{username}")
	public ResponseEntity<?> getComplaintByUsername(@PathVariable("username") String username){
		List<Complaint> complaints = this.complaintService.findComplaintByUsername(username);
		if(complaints.isEmpty()) {
			return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
		}else {
			return ResponseEntity.ok(complaints);
		}
	}

	@GetMapping("/complaint-feedback/{id}")
	public ResponseEntity<?> getComplaintById(@PathVariable("id") int id){
		Complaint complaint = this.complaintService.getComplaint(id);
		return ResponseEntity.ok(complaint);
	}

	@PostMapping("/save-feedback")
	public ResponseEntity<?> saveFeedback(@RequestBody Feedback feedback) throws Exception{
		Feedback savedFeedback = this.complaintService.saveFeedback(feedback);
		return ResponseEntity.ok(savedFeedback);
```

```
        }


}
```

```
package com.crs.controller;

import java.net.URI;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import javax.validation.Valid;
import javax.annotation.PostConstruct;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import com.crs.entities.Role;
import com.crs.entities.User;
import com.crs.entities.UserRole;
```

```
import com.crs.service.UserService;

@RestController
@CrossOrigin(origins = "*")
@RequestMapping("/user")
public class UserController {
	@Autowired
	private UserService userService;

	//create user
	@PostMapping("/create-user")
	public ResponseEntity<User> createNewUser(@Valid @RequestBody User user){
		Set<UserRole> userRole = new HashSet<>();
		Role role = new Role();
		if(user.getRoleName().contentEquals("CUSTOMER")) {
			role.setRoleId(102);
			role.setRoleName(user.getRoleName());
		}else if(user.getRoleName().contentEquals("MANAGER")) {
			role.setRoleId(104);
			role.setRoleName(user.getRoleName());
		}else if(user.getRoleName().contentEquals("ENGINEER")) {
			role.setRoleId(106);
			role.setRoleName(user.getRoleName());
		}

		UserRole uR = new UserRole();
		uR.setUser(user);
		uR.setRole(role);
		userRole.add(uR);
		if(this.userService.getUserName(user.getUsername())!=null) {
```

```
                System.out.println("Username already exist!");
                return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
        }else {
                User createUser = this.userService.createUser(user, userRole);
                URI location =
ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(createU
ser.getUserId()).toUri();
                return ResponseEntity.created(location).build();
        }
    }

    //create admin
    @PostConstruct
    public void createAdmin() {
        User admin = new User();
        admin.setUsername("crs-admin@abc.com");
        admin.setPassword("admin@crs");
        admin.setFirstName("Twarit");
        admin.setLastName("Soni");
        admin.setEmail("twarit.soni@gmail.com");
        admin.setPinCode(110001);
        admin.setPhone("+916265458854");
        admin.setRoleName("ADMIN");
        Role role = new Role();
        role.setRoleId(101);
        role.setRoleName(admin.getRoleName());
        Set<UserRole> userRole = new HashSet<>();
        UserRole uR = new UserRole();
        uR.setUser(admin);
        uR.setRole(role);
```

```
        userRole.add(uR);
        User userAdmin = this.userService.createUser(admin, userRole);
        System.out.println("Admin Username: "+userAdmin.getUsername());
    }

    //get user by username
    @GetMapping("/get-user/{username}")
    public ResponseEntity<User> getUserByUsername(@PathVariable("username")
String username){
        User user = this.userService.getUserName(username);
        if(user!=null) {
            return ResponseEntity.ok(user);
        }else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }
    }

    //delete user by userid
    @DeleteMapping("/delete-user/{userId}")
    public ResponseEntity<?> deleteUser(@PathVariable("userId") Integer userId){
        this.userService.deleteUserById(userId);
        return ResponseEntity.status(HttpStatus.OK).build();
    }

    //update user by username
    @PutMapping("/update-user/{username}")
    public ResponseEntity<User> updateUser(@PathVariable("username") String
username,@RequestBody User user){
        this.userService.updateUserByUsername(username, user);
        return ResponseEntity.status(HttpStatus.CREATED).build();
    }
```

```
	//get user by role name
	@GetMapping("/get-all/{roleName}")
	public ResponseEntity<?> getAllUserByRole(@PathVariable("roleName") String roleName){
		List<User> users = this.userService.getUserByRole(roleName);
		if(users.isEmpty()) {
			return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
		}else {
			return ResponseEntity.ok(users);
		}
	}

}
```

```
package com.crs.controller;

import java.util.HashMap;
import java.util.Map;

import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

@ControllerAdvice
public class ValidationHandler extends ResponseEntityExceptionHandler{
```

```
	@Override
	protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
			HttpHeaders headers, HttpStatus status, WebRequest request) {

		Map<String, String> errors = new HashMap<>();
		ex.getBindingResult().getAllErrors().forEach((error) ->{

			String fieldName = ((FieldError) error).getField();
			String message = error.getDefaultMessage();
			errors.put(fieldName, message);
		});
		return new ResponseEntity<Object>(errors, HttpStatus.BAD_REQUEST);
	}


}
```

```
package com.crs.entities;

import org.springframework.security.core.GrantedAuthority;

public class Authority implements GrantedAuthority{

	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	private String authority;

	public Authority(String authority) {
		this.authority = authority;
	}

	@Override
	public String getAuthority() {
		return this.authority;
	}
```

}

```
package com.crs.entities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

@Entity
public class Complaint {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private int id;

	@NotNull(message = "username field is required")
	private String username;

	@NotNull(message = "first name field is required")
	private String firstName;

	@NotNull(message = "last name field is required")
	private String lastName;

	@NotNull(message = "address field is required")
	private String address;

	@NotNull(message = "enter 6 digit pincode")
```

```
@Min(100000)
private int pinCode;

@NotNull(message = "state field is required")
private String state;

@NotNull(message = "contact field is required")
private String contact;

@NotNull(message = "complaint field is required")
private String complaint;

private String status;

private String assignedEngineer;

private String remark;

private String date;

private boolean isActive;

private boolean isAssigned;

public Complaint() {

}

public Complaint(int id, String username, String firstName, String lastName, String
address, int pinCode,
```

```
                String state, String contact, String complaint, String status, String
assignedEngineer, String remark, String date, boolean isActive, boolean isAssigned) {
        super();
        this.id = id;
        this.username = username;
        this.firstName = firstName;
        this.lastName = lastName;
        this.address = address;
        this.pinCode = pinCode;
        this.state = state;
        this.contact = contact;
        this.complaint = complaint;
        this.status = status;
        this.assignedEngineer = assignedEngineer;
        this.remark = remark;
        this.date = date;
        this.isActive = isActive;
        this.isAssigned = isAssigned;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }
```

```
public void setUsername(String username) {
	this.username = username;
}

public String getFirstName() {
	return firstName;
}

public void setFirstName(String firstName) {
	this.firstName = firstName;
}

public String getLastName() {
	return lastName;
}

public void setLastName(String lastName) {
	this.lastName = lastName;
}

public String getAddress() {
	return address;
}

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

public int getPinCode() {
```

```
        return pinCode;
}

public void setPinCode(int pinCode) {
        this.pinCode = pinCode;
}

public String getState() {
        return state;
}

public void setState(String state) {
        this.state = state;
}

public String getContact() {
        return contact;
}

public void setContact(String contact) {
        this.contact = contact;
}

public String getComplaint() {
        return complaint;
}

public void setComplaint(String complaint) {
        this.complaint = complaint;
}
```

```
public String getStatus() {
        return status;
}

public void setStatus(String status) {
        this.status = status;
}

public String getAssignedEngineer() {
        return assignedEngineer;
}

public void setAssignedEngineer(String assignedEngineer) {
        this.assignedEngineer = assignedEngineer;
}

public String getRemark() {
        return remark;
}

public void setRemark(String remark) {
        this.remark = remark;
}

public String getDate() {
        return date;
}

public void setDate(String date) {
```

```
		this.date = date;
	}

	public boolean isActive() {
		return isActive;
	}

	public void setActive(boolean isActive) {
		this.isActive = isActive;
	}

	public boolean isAssigned() {
		return isAssigned;
	}

	public void setAssigned(boolean isAssigned) {
		this.isAssigned = isAssigned;
	}

}
```

```
package com.crs.entities;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
```

```
@Entity
public class UserRole {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private int userRoleId;

	@ManyToOne(fetch = FetchType.EAGER)
	private User user;

	@ManyToOne
	private Role role;

	public UserRole() {

	}

	public UserRole(int userRoleId, User user, Role role) {
		super();
		this.userRoleId = userRoleId;
		this.user = user;
		this.role = role;
	}

	public int getUserRoleId() {
		return userRoleId;
	}

	public void setUserRoleId(int userRoleId) {
		this.userRoleId = userRoleId;
	}
```

```
	public User getUser() {
		return user;
	}

	public void setUser(User user) {
		this.user = user;
	}

	public Role getRole() {
		return role;
	}

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


}
```

```
package com.crs.repo;

import java.util.List;

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

import com.crs.entities.Complaint;

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

```
	public List<Complaint> findByUsername(String username);
	public List<Complaint> findByAssignedEngineer(String assignedEngineer);
	public Complaint findByComplaintAndUsernameAndIsActive(String complaint,
String username, boolean isActive);
	public List<Complaint> findByIsAssigned(boolean isAssigned);
	public List<Complaint> findByPinCodeAndIsAssigned(int pinCode, boolean
isAssigned);
}
```

```
package com.crs.repo;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.crs.entities.User;


@Repository
public interface UserRepo extends JpaRepository<User, Integer>{
	public User findByUsername(String username);
	public List<User> findByRoleName(String roleName);
}
```

```
package com.crs.service;

import java.util.List;
import java.util.Set;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import com.crs.entities.User;
```

```
import com.crs.entities.UserRole;
import com.crs.repo.RoleRepo;
import com.crs.repo.UserRepo;

@Service
public class UserService {
	@Autowired
	private UserRepo userRepo;

	@Autowired
	private RoleRepo roleRepo;

	@Autowired
	private BCryptPasswordEncoder passwordEncoder;

	public UserService(UserRepo userRepo, RoleRepo roleRepo) {
		super();
		this.userRepo = userRepo;
		this.roleRepo = roleRepo;
	}

	// creating new user
	public User createUser(User user, Set<UserRole> userRole){
		User local = this.userRepo.findByUsername(user.getUsername());
		try {
			if(local!=null) {
				throw new Exception("Username already exists!");
			}
			else {
				//user creation
```

```
                        //saving role from userRole
                        for(UserRole ur : userRole) {
                                roleRepo.save(ur.getRole());
                        }
                        //assign userRole in user
                        user.getUserRoles().addAll(userRole);

                        //encode password

user.setPassword(this.passwordEncoder.encode(user.getPassword()));

                        local = this.userRepo.save(user);
                }
        }catch(Exception e) {
                System.out.println(e);
        }
        return local;

}
//find user by username
public User getUserName(String username) {
        User findUser = this.userRepo.findByUsername(username);
        return findUser;
}

//find user by role
public List<User> getUserByRole(String roleName){
        return this.userRepo.findByRoleName(roleName);
}
```

```
	//delete user by userid
	public void deleteUserById(Integer userId) {
		this.userRepo.deleteById(userId);
	}

	//update user by username
	public User updateUserByUsername(String username,User user) {
		User u = this.userRepo.findByUsername(username);
		u.setFirstName(user.getFirstName());
		u.setLastName(user.getLastName());
		u.setPhone(user.getPhone());
		u.setEmail(user.getEmail());
		u.setPinCode(user.getPinCode());
		User updatedUser = this.userRepo.save(u);
		return updatedUser;
	}
}
```

Project Frontend Code

```
<!doctype html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>ComplaintRedressalSystem</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
    integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>

<body>
```

```
  <app-root></app-root>

  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
    integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
    crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"
    integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
    crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"
    integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
    crossorigin="anonymous"></script>
</body>

</html>
```

```
<main>
    <div class="container-fluid">
        <div class="row">
            <div class="col">
                <div class="card mx-auto bg-light mb-2 mt-2">
                    <div class="card-body">
                        <h1 class="card-title text-center">Welcome Admin:
Pravin
                        </h1>
                        <p class="card-text text-center">Love to see you again
<b>{{username}}</b></p>
                        <div class="card-deck">
                            <div class="card bg-info">
                                <img class="card-img-top"
                                    src="https://www.freeiconspng.com/thumbs/c
ustomers-icon/customers-icon-12.png"
                                    alt="Card image" style="width:100%;
height:250px;">
                                <div class="card-body">
                                    <h4 class="card-title">Create a User</h4>
                                    <p class="card-text">Create new user like
customer, manager & engineer. </p>

                                </div>
                                <div class="card-footer">
                                    <a routerLink="/register-user"
role="button"
```

```
                                        class="btn btn-danger btn-
block"><b>Create
                                            User</b>
                                    </a>
                                </div>
                            </div>

                            <div class="card bg-success">
                                <img class="card-img-top"
                                    src="https://icons.iconarchive.com/icons/a
ha-soft/free-large-boss/512/Manager-icon.png"
                                    alt="Card image" style="width:100%;
height:250px;">
                                <div class="card-body">
                                    <h4 class="card-title">Show all
Manager's</h4>
                                    <p class="card-text">Display all available
manager's.</p>

                                </div>
                                <div class="card-footer">
                                    <a routerLink="/managers" role="button"
class="btn btn-danger btn-block"><b>Show
                                            Manager's</b></a>
                                </div>
                            </div>

                            <div class="card bg-primary">
                                <img class="card-img-top"
                                    src="https://iconarchive.com/download/i693
97/aha-soft/free-large-boss/Businessman.512.png"
                                    alt="Card image" style="width:100%;
height:250px;">
                                <div class="card-body">
                                    <h4 class="card-title">Show all
Engineer's</h4>
                                    <p class="card-text">Display all available
engineer's.</p>

                                </div>
                                <div class="card-footer">
                                    <a routerLink="/engineers" role="button"
class="btn btn-danger btn-block"><b>Show
                                            Engineer's</b></a>
                                </div>
                            </div>

                            <div class="card bg-warning">
```

```
                                <img class="card-img-top"
                                    src="https://www.pngplay.com/wp-
content/uploads/7/Customer-Logo-Transparent-File.png"
                                    alt="Card image" style="width:100%;
height:250px;">
                                <div class="card-body">
                                    <h4 class="card-title">Show all
Customer's</h4>
                                    <p class="card-text">Display all available
customer's.</p>

                                </div>
                                <div class="card-footer">
                                    <a routerLink="/customers" role="button"
class="btn btn-danger btn-block"><b>Show
                                                Customer's</b></a>
                                </div>
                            </div>

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

    </div>
</main>
```

```
<div class="container">
    <div class="row">
        <div class="col-lg-5 col-md-6 mx-auto mt-5">
            <div class="card mt-3 bg-light">
                <div class="card-body">
                    <h4 class="card-title text-center">Change Password</h4>
                    <hr>
                    <form #changepassword="ngForm" (ngSubmit)="onSubmit()">
                        <div class="form-group">
                            <label for="email">Username:</label> <input
type="text" name="username" class="form-control"
                                #username="ngModel"
[(ngModel)]="credentials.username" placeholder="Enter Username"
                                id="email" required ngModel readonly>
                            <span *ngIf="username.invalid && username.touched"
style="color:red">Please enter a valid
                                username.</span>
                        </div>

                        <div class="form-group">
```

```
                            <label for="pwd1">New Password:</label> <input
type="password" name="password"
                                [(ngModel)]="credentials.password"
#password="ngModel" minlength="8"
                                class="form-control" placeholder="Enter new
password" id="pwd1" required ngModel>
                            <span *ngIf="password.invalid && password.touched"
style="color: red">Please enter a valid
                                password.</span>
                        </div>

                        <div class="form-group">
                            <label for="pwd2">Confirm New Password:</label>
<input type="password" name="cnfPassword"
                                #cnfPassword="ngModel" minlength="8"
class="form-control" [(ngModel)]="newPassword"
                                placeholder="Confirm new password" id="pwd2"
required ngModel>
                            <span *ngIf="cnfPassword.invalid &&
cnfPassword.touched" style="color: red">Please enter a
                                valid password.</span>
                        </div>
                        <button [disabled]="changepassword.invalid"
type="submit"
                            class="btn btn-danger btn-block">Change
Password</button>
                    </form>

                </div>

            </div>
        </div>

    </div>
</div>
```

```
<div class="container">
    <div class="row">
        <div class="col">
            <div class="jumbotron jumbotron-fluid bg-secondary">
                <div class="text-center text-white">
                    <h1>Welcome {{name}} !</h1>
                </div>
            </div>
            <div class="card-deck mx-auto mb-3">
                <div class="card bg-light">
                    <img class="card-img-top"
```

```
                        src="https://www.clipartkey.com/mpngs/m/125-
1257963_transparent-hearing-clipart-complaint-management-system-logo-png.png"
                        alt="Card image" style="width:100%; height:250px;">
                    <div class="card-body">
                        <h4 class="card-title">Raise a new complaint</h4>
                        <a routerLink="/create-complaint" role="button"
class="btn btn-danger btn-block"><b>Raise
                                    Complaint</b></a>
                    </div>
                </div>
                <div class="card bg-info">
                    <img class="card-img-top"
                        src="https://www.smithpublicity.com/wp-
content/uploads/complaint-box-web-300x232.png"
                        alt="Card image" style="width:100%; height:250px;">
                    <div class="card-body">
                        <h4 class="card-title">Show all Complaint's</h4>
                        <a routerLink="/customer/get-complaints" role="button"
class="btn btn-warning btn-block"><b>Show
                                    Complaint's</b></a>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

```
<br>
<main>
    <div class="container">
        <div class="row">
            <div class="col">
                <h2 class="text-center text-white bg-dark">All Available
Customer's</h2>
                <div class="table-responsive">
                    <table class="table table-dark table-striped table-
bordered table-hover">
                        <thead class="thead-dark text-center">
                            <tr>
                                <th>S.No.</th>
                                <th>Customer Id</th>
                                <th>User Name</th>
                                <th>First Name</th>
                                <th>Last Name</th>
                                <th>Pincode</th>
                                <th>E-mail</th>
                                <th>Contact</th>
                                <th>Action</th>
```

```
                        </tr>
                    </thead>
                    <tbody class="text-center">
                        <tr *ngFor="let customer of user">
                            <td></td>
                            <td><span>{{customer.userId}}</span></td>
                            <td><span>{{customer.username}}</span></td>
                            <td><span>{{customer.firstName}}</span></td>
                            <td><span>{{customer.lastName}}</span></td>
                            <td><span>{{customer.pinCode}}</span></td>
                            <td><span>{{customer.email}}</span></td>
                            <td><span>{{customer.phone}}</span></td>

                            <td>
                                <div class="btn-group" role="group" aria-
label="Basic example">
                                    <button
(click)="deleteCustomer(customer.userId)"
                                        class="btn btn-
danger">Delete</button>
                                    <button
(click)="updateCustomer(customer.username)"
                                        class="btn btn-
info">Update</button>
                                </div>
                            </td>
                        </tr>

                    </tbody>
                </table>
            </div>
        </div>
    </div>
</main>
```

```
<div class="container">
    <div class="row">
        <div class="col-6 mx-auto mt-3 mb-4">
            <div class="card">
                <div class="card-header">
                    <h4 class="text-center">Customer Feedback</h4>
                </div>
                <div class="card-body">
                    <form (ngSubmit)="onSubmit()">

                        <div class="form-group">
                            <label for="cid">Complaint Id:</label>
```

```
                            <input type="text" class="form-control" id="cid"
name="cid" [(ngModel)]="feedback.cid"
                                readonly>
                        </div>
                        <div class="form-group">
                            <label for="username">Username:</label>
                            <input type="text" class="form-control"
id="username" name="username"
                                [(ngModel)]="feedback.username" readonly>
                        </div>
                        <div class="form-group">
                            <label for="cmp">Complaint:</label>
                            <input type="text" class="form-control" id="cmp"
name="complaint"
                                [(ngModel)]="feedback.complaint" readonly>
                        </div>
                        <div class="form-group">
                            <label for="fdbck">Feedback</label>
                            <textarea class="form-control" rows="5" id="fdbck"
[(ngModel)]="feedback.feedback"
                                name="feedback" placeholder="enter your
feedback..."></textarea>

                        </div>

                        <div class="text-center">

                            <button type="submit" class="btn btn-primary"
data-toggle="modal"
                                data-target="#myModal">Submit
Feedback</button>
                            <!-- The Modal -->
                            <div class="modal fade" id="myModal">
                                <div class="modal-dialog modal-dialog-
centered">
                                    <div class="modal-content">

                                        <!-- Modal Header -->
                                        <div class="modal-header">
                                            <h4 class="modal-title">Customer
feedback</h4>
                                            <button type="button"
class="close" data-dismiss="modal">&times;</button>
                                        </div>

                                        <!-- Modal body -->
                                        <div class="modal-body">
```

```
                                        <span *ngIf="isValid"
style="color:green">{{message}}</span>
                                        <span *ngIf="!isValid"
style="color: red">{{message}}</span>
                                    </div>

                                    <!-- Modal footer -->
                                    <div class="modal-footer">
                                        <button type="button" class="btn
btn-secondary" (click)="onClick()"
                                            data-
dismiss="modal">Close</button>
                                    </div>

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

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


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