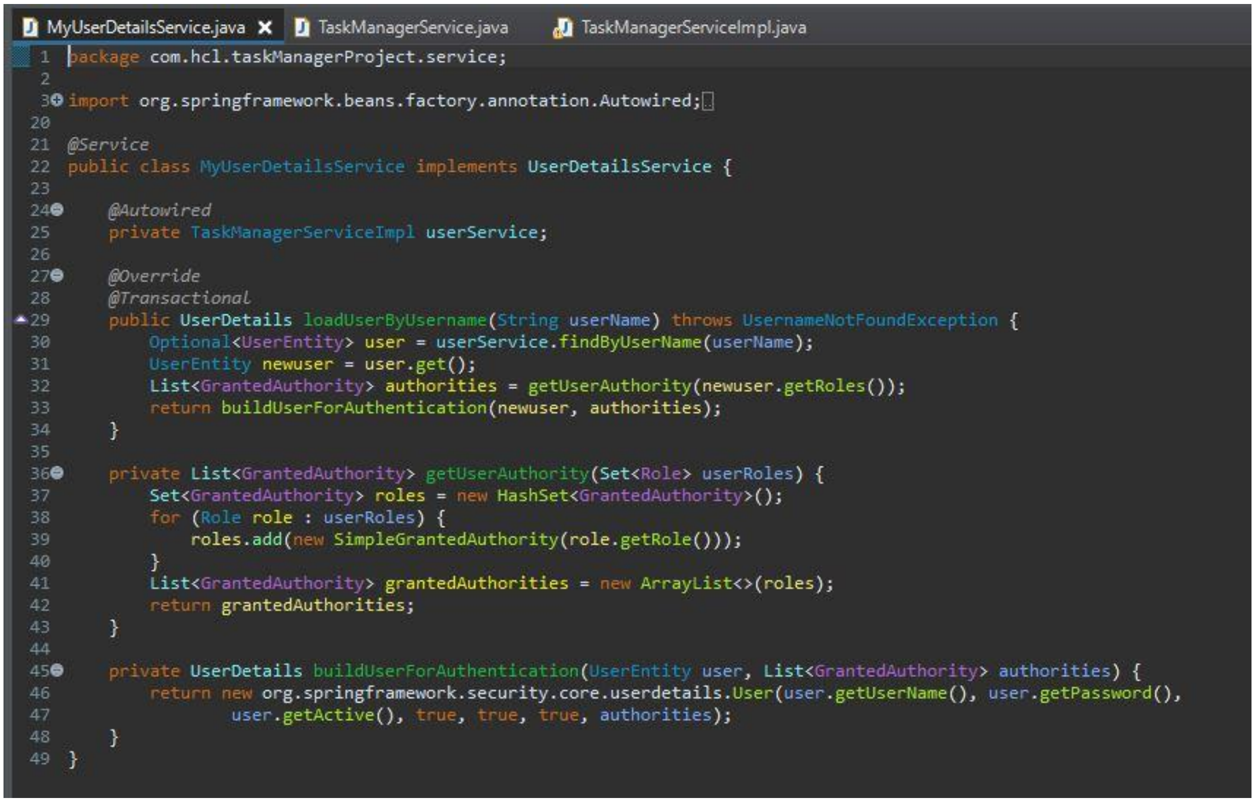
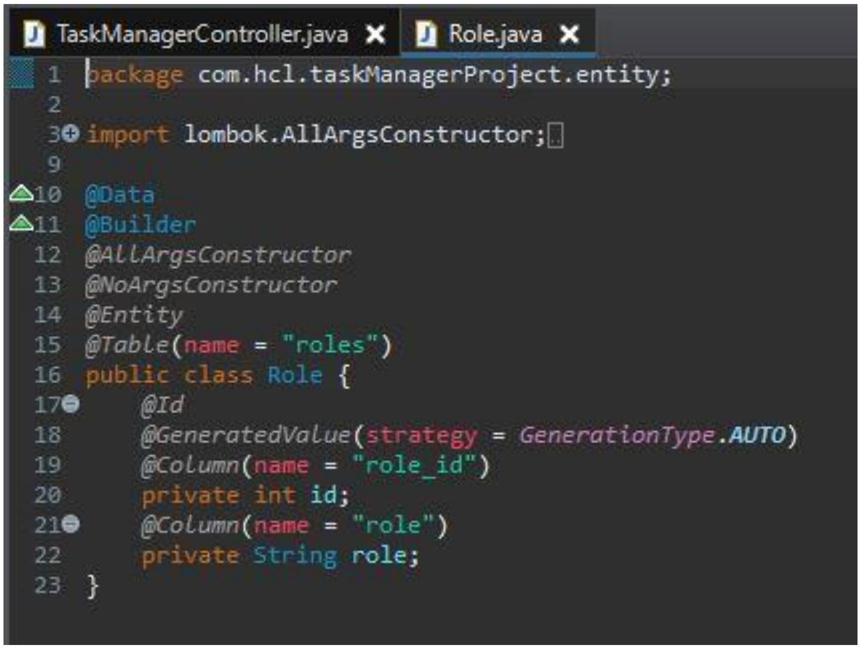
Task Manager Codes

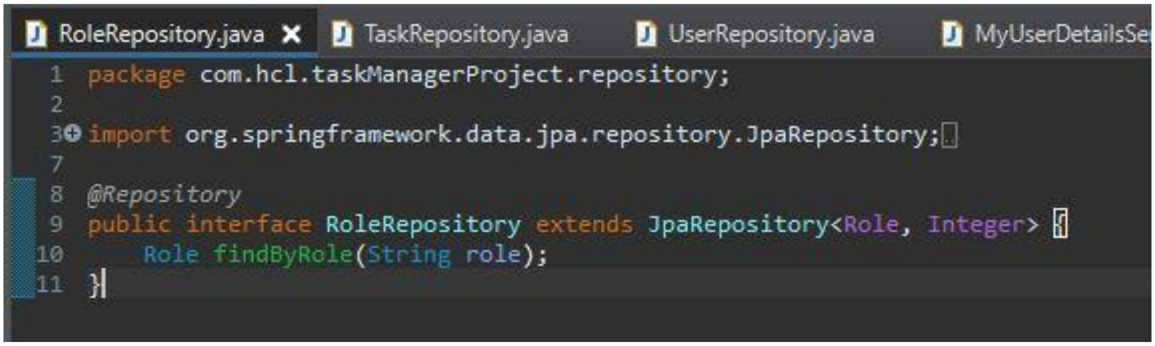
Java:



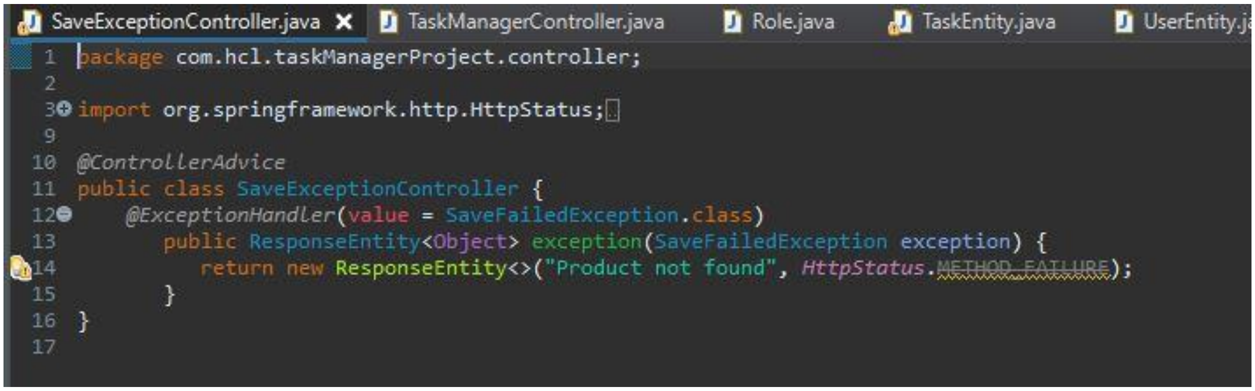
```java
package com.hcl.taskManagerProject.service;

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

@Service
public class MyUserDetailsService implements UserDetailsService {

    @Autowired
    private TaskManagerServiceImpl userService;

    @Override
    @Transactional
    public UserDetails loadUserByUsername(String userName) throws UsernameNotFoundException {
        Optional<UserEntity> user = userService.findByUserName(userName);
        UserEntity newuser = user.get();
        List<GrantedAuthority> authorities = getUserAuthority(newuser.getRoles());
        return buildUserForAuthentication(newuser, authorities);
    }

    private List<GrantedAuthority> getUserAuthority(Set<Role> userRoles) {
        Set<GrantedAuthority> roles = new HashSet<GrantedAuthority>();
        for (Role role : userRoles) {
            roles.add(new SimpleGrantedAuthority(role.getRole()));
        }
        List<GrantedAuthority> grantedAuthorities = new ArrayList<>(roles);
        return grantedAuthorities;
    }

    private UserDetails buildUserForAuthentication(UserEntity user, List<GrantedAuthority> authorities) {
        return new org.springframework.security.core.userdetails.User(user.getUserName(), user.getPassword(),
                user.getActive(), true, true, true, authorities);
    }
}
```



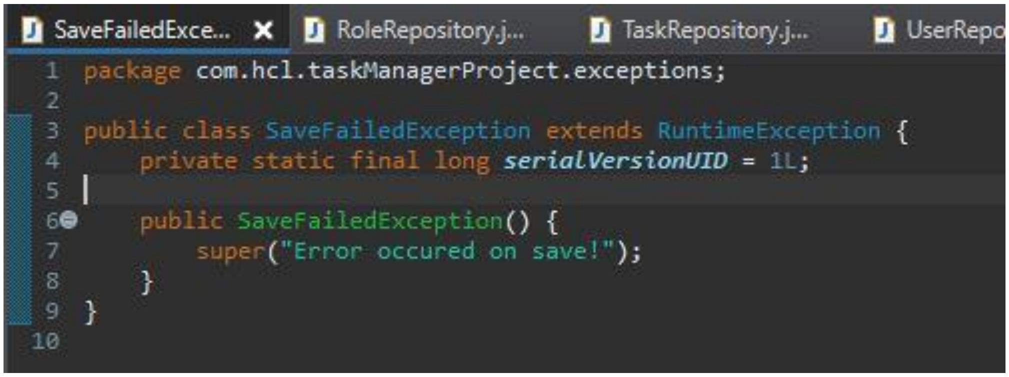
```java
package com.hcl.taskManagerProject.entity;

import lombok.AllArgsConstructor;

@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "roles")
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "role_id")
    private int id;
    @Column(name = "role")
    private String role;
}
```



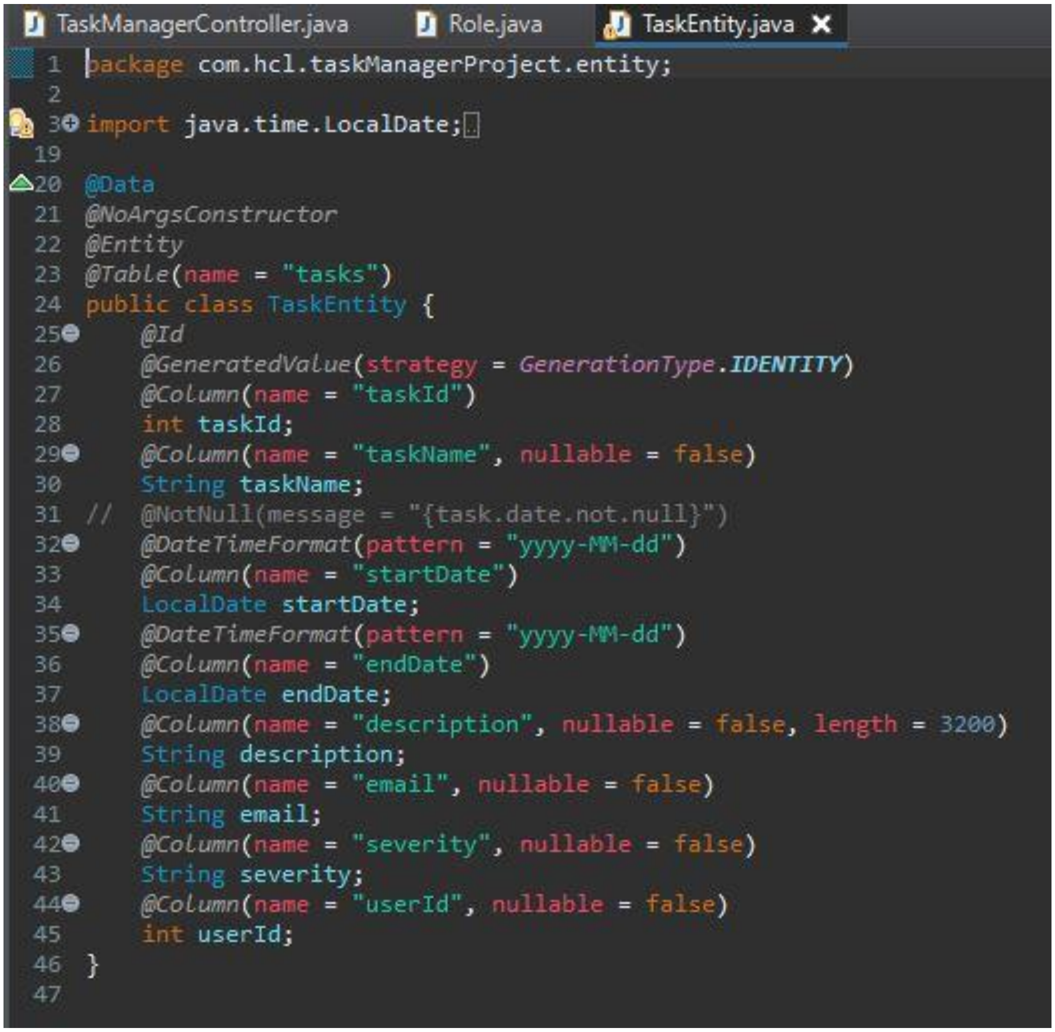
```java
package com.hcl.taskManagerProject.repository;

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

@Repository
public interface RoleRepository extends JpaRepository<Role, Integer> {
    Role findByRole(String role);
}
```



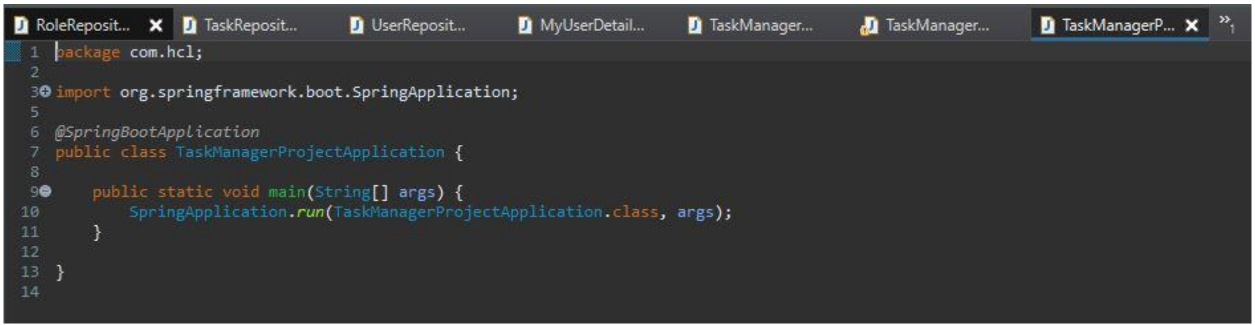
```java
package com.hcl.taskManagerProject.controller;

import org.springframework.http.HttpStatus;

@ControllerAdvice
public class SaveExceptionController {
    @ExceptionHandler(value = SaveFailedException.class)
        public ResponseEntity<Object> exception(SaveFailedException exception) {
            return new ResponseEntity<>("Product not found", HttpStatus.METHOD_FAILURE);
        }
}
```



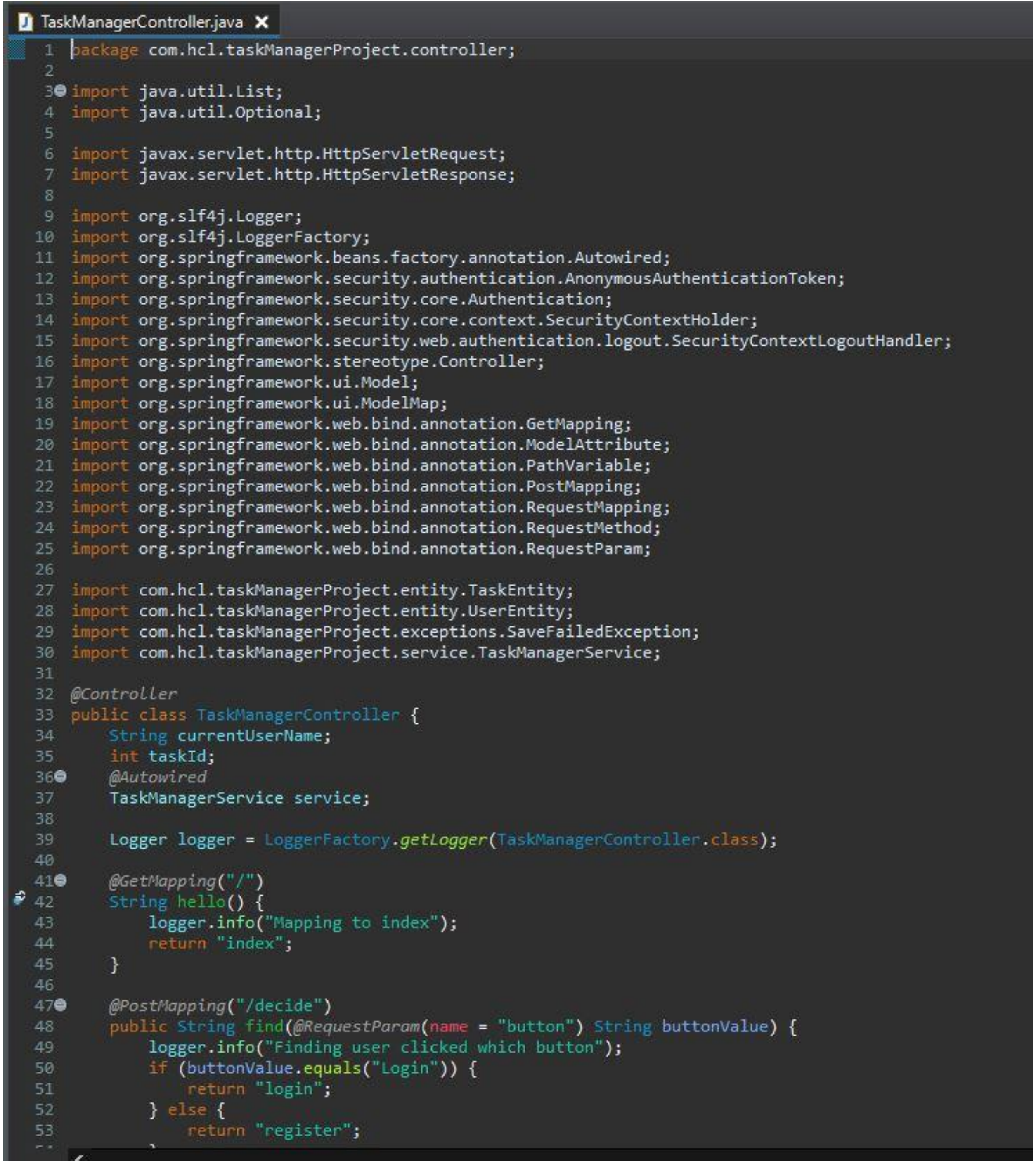
```java
package com.hcl.taskManagerProject.exceptions;

public class SaveFailedException extends RuntimeException {
    private static final long serialVersionUID = 1L;

    public SaveFailedException() {
        super("Error occured on save!");
    }
}
```



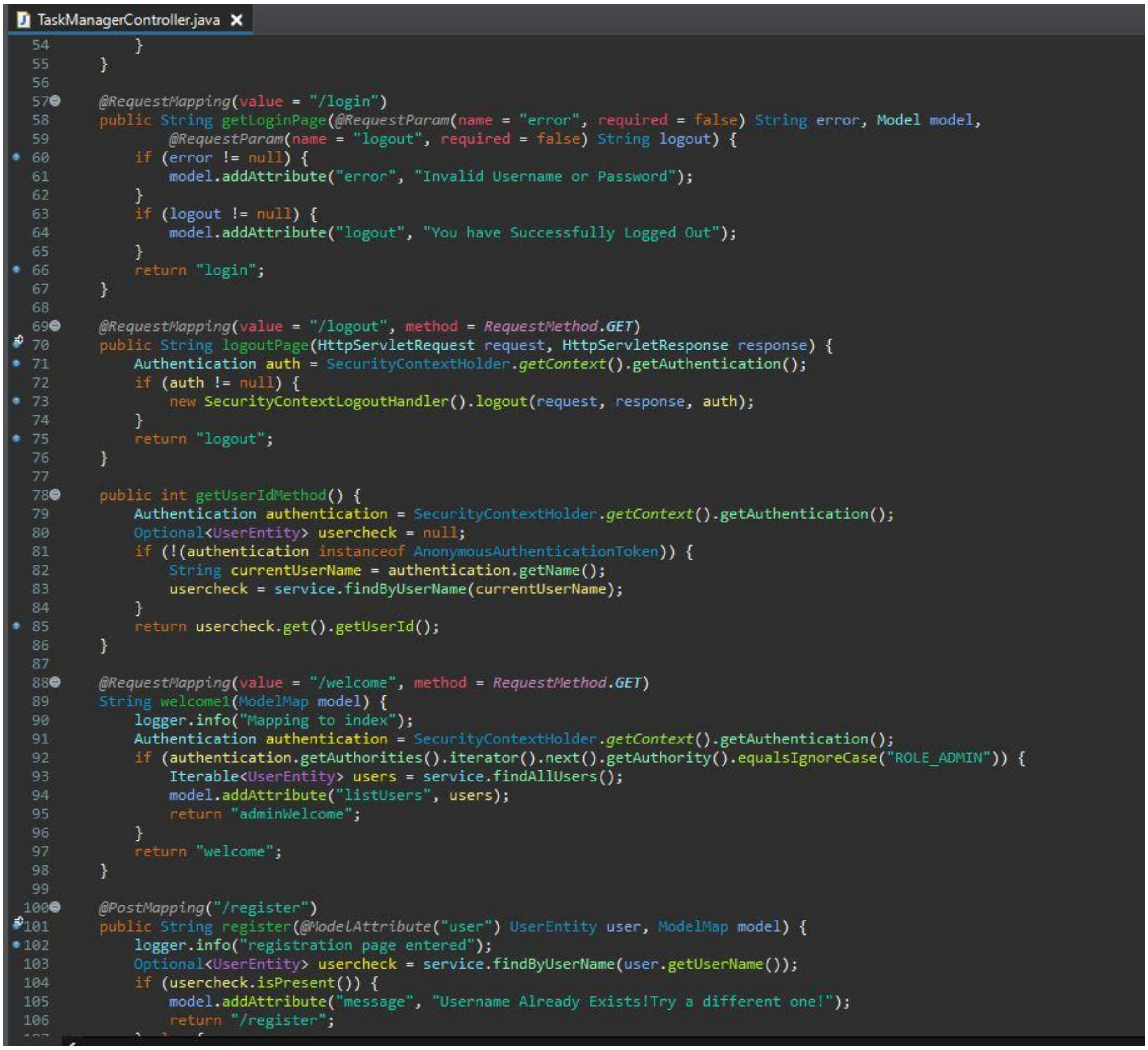
```java
package com.hcl.taskManagerProject.entity;

import java.time.LocalDate;

@Data
@NoArgsConstructor
@Entity
@Table(name = "tasks")
public class TaskEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "taskId")
    int taskId;
    @Column(name = "taskName", nullable = false)
    String taskName;
//  @NotNull(message = "{task.date.not.null}")
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    @Column(name = "startDate")
    LocalDate startDate;
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    @Column(name = "endDate")
    LocalDate endDate;
    @Column(name = "description", nullable = false, length = 3200)
    String description;
    @Column(name = "email", nullable = false)
    String email;
    @Column(name = "severity", nullable = false)
    String severity;
    @Column(name = "userId", nullable = false)
    int userId;
}
```



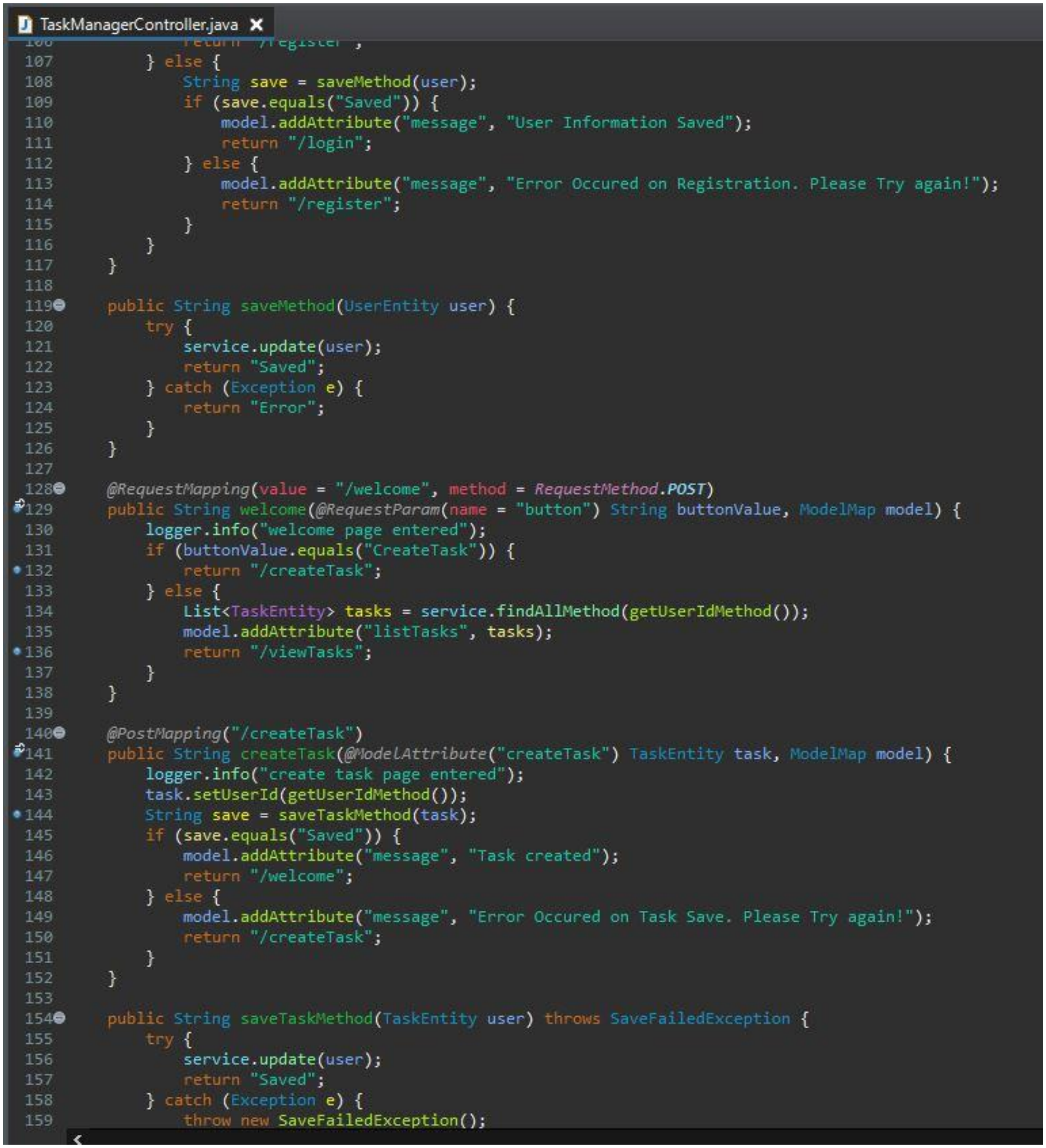
```java
package com.hcl;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class TaskManagerProjectApplication {

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

}
```



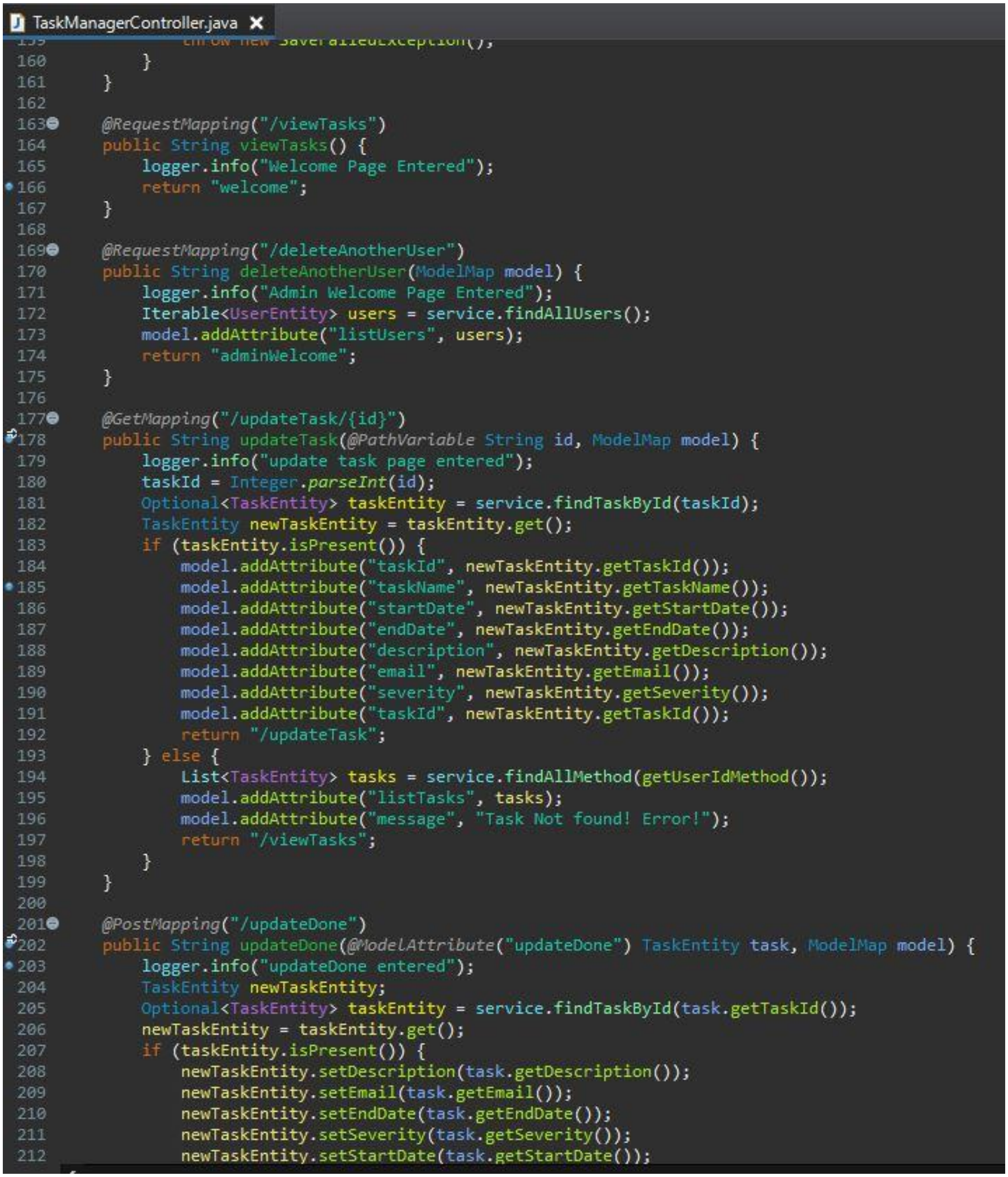
```java
package com.hcl.taskManagerProject.controller;

import java.util.List;
import java.util.Optional;

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

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.AnonymousAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import com.hcl.taskManagerProject.entity.TaskEntity;
import com.hcl.taskManagerProject.entity.UserEntity;
import com.hcl.taskManagerProject.exceptions.SaveFailedException;
import com.hcl.taskManagerProject.service.TaskManagerService;

@Controller
public class TaskManagerController {
    String currentUserName;
    int taskId;
    @Autowired
    TaskManagerService service;

    Logger logger = LoggerFactory.getLogger(TaskManagerController.class);

    @GetMapping("/")
    String hello() {
        logger.info("Mapping to index");
        return "index";
    }

    @PostMapping("/decide")
    public String find(@RequestParam(name = "button") String buttonValue) {
        logger.info("Finding user clicked which button");
        if (buttonValue.equals("Login")) {
            return "login";
        } else {
            return "register";
```



```java
        }
    }

    @RequestMapping(value = "/login")
    public String getLoginPage(@RequestParam(name = "error", required = false) String error, Model model,
            @RequestParam(name = "logout", required = false) String logout) {
        if (error != null) {
            model.addAttribute("error", "Invalid Username or Password");
        }
        if (logout != null) {
            model.addAttribute("logout", "You have Successfully Logged Out");
        }
        return "login";
    }

    @RequestMapping(value = "/logout", method = RequestMethod.GET)
    public String logoutPage(HttpServletRequest request, HttpServletResponse response) {
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        if (auth != null) {
            new SecurityContextLogoutHandler().logout(request, response, auth);
        }
        return "logout";
    }

    public int getUserIdMethod() {
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        Optional<UserEntity> usercheck = null;
        if (!(authentication instanceof AnonymousAuthenticationToken)) {
            String currentUserName = authentication.getName();
            usercheck = service.findByUserName(currentUserName);
        }
        return usercheck.get().getUserId();
    }

    @RequestMapping(value = "/welcome", method = RequestMethod.GET)
    String welcome1(ModelMap model) {
        logger.info("Mapping to index");
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        if (authentication.getAuthorities().iterator().next().getAuthority().equalsIgnoreCase("ROLE_ADMIN")) {
            Iterable<UserEntity> users = service.findAllUsers();
            model.addAttribute("listUsers", users);
            return "adminWelcome";
        }
        return "welcome";
    }

    @PostMapping("/register")
    public String register(@ModelAttribute("user") UserEntity user, ModelMap model) {
        logger.info("registration page entered");
        Optional<UserEntity> usercheck = service.findByUserName(user.getUserName());
        if (usercheck.isPresent()) {
            model.addAttribute("message", "Username Already Exists!Try a different one!");
            return "/register";
```

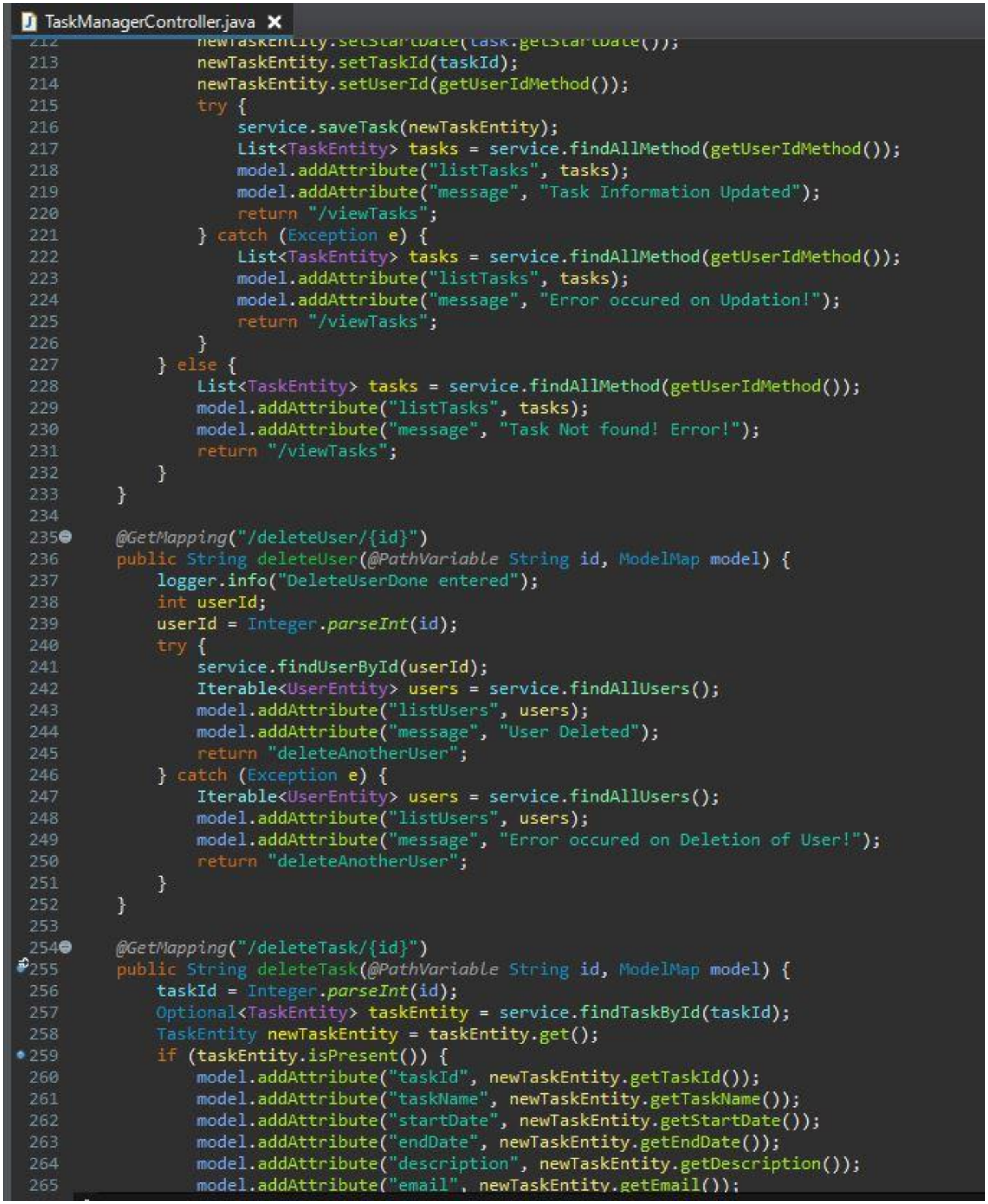


```
            return "/register";
        } else {
            String save = saveMethod(user);
            if (save.equals("Saved")) {
                model.addAttribute("message", "User Information Saved");
                return "/login";
            } else {
                model.addAttribute("message", "Error Occured on Registration. Please Try again!");
                return "/register";
            }
        }
    }

    public String saveMethod(UserEntity user) {
        try {
            service.update(user);
            return "Saved";
        } catch (Exception e) {
            return "Error";
        }
    }

    @RequestMapping(value = "/welcome", method = RequestMethod.POST)
    public String welcome(@RequestParam(name = "button") String buttonValue, ModelMap model) {
        logger.info("welcome page entered");
        if (buttonValue.equals("CreateTask")) {
            return "/createTask";
        } else {
            List<TaskEntity> tasks = service.findAllMethod(getUserIdMethod());
            model.addAttribute("listTasks", tasks);
            return "/viewTasks";
        }
    }

    @PostMapping("/createTask")
    public String createTask(@ModelAttribute("createTask") TaskEntity task, ModelMap model) {
        logger.info("create task page entered");
        task.setUserId(getUserIdMethod());
        String save = saveTaskMethod(task);
        if (save.equals("Saved")) {
            model.addAttribute("message", "Task created");
            return "/welcome";
        } else {
            model.addAttribute("message", "Error Occured on Task Save. Please Try again!");
            return "/createTask";
        }
    }

    public String saveTaskMethod(TaskEntity user) throws SaveFailedException {
        try {
            service.update(user);
            return "Saved";
        } catch (Exception e) {
            throw new SaveFailedException();
```

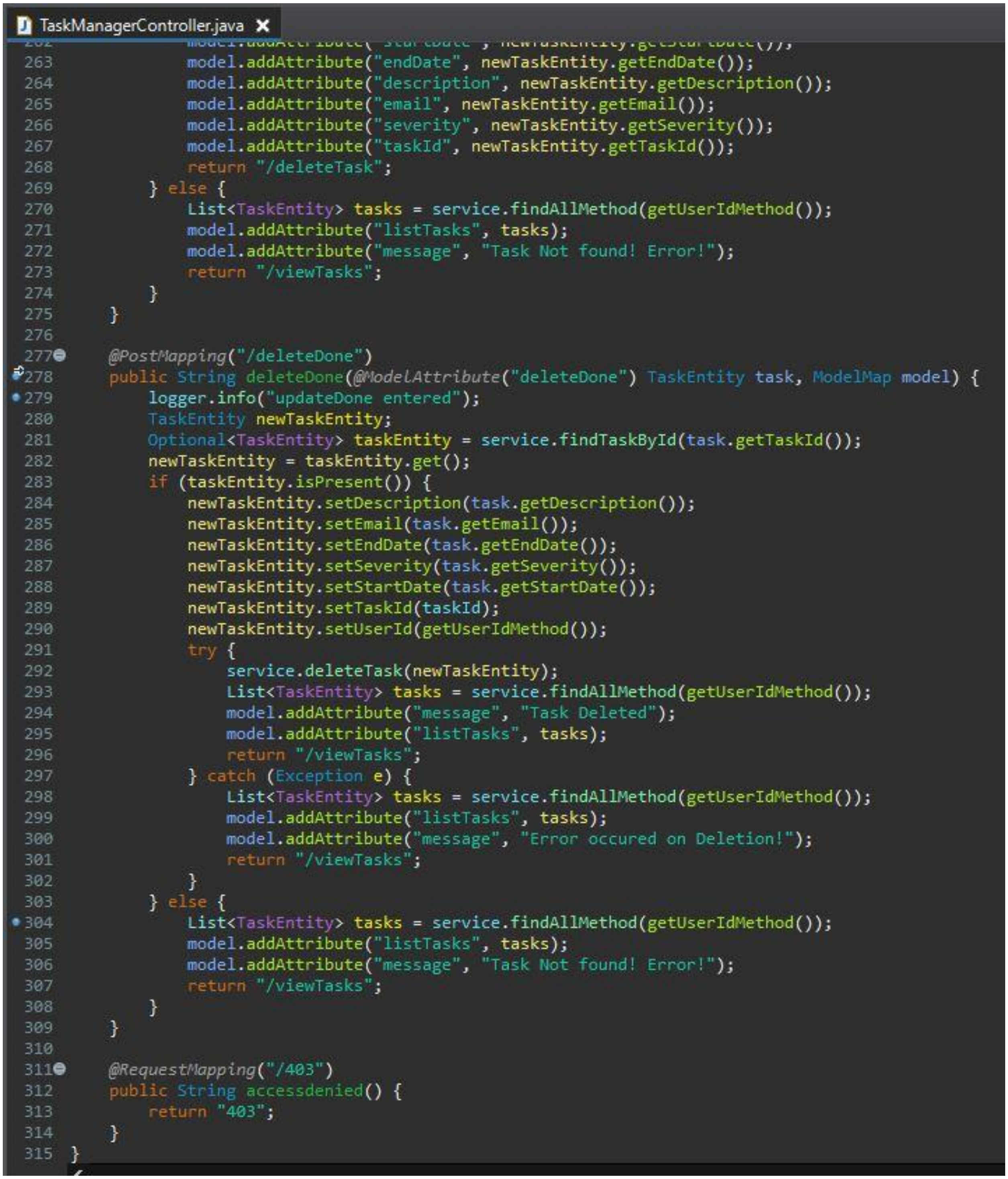


TaskManagerController.java

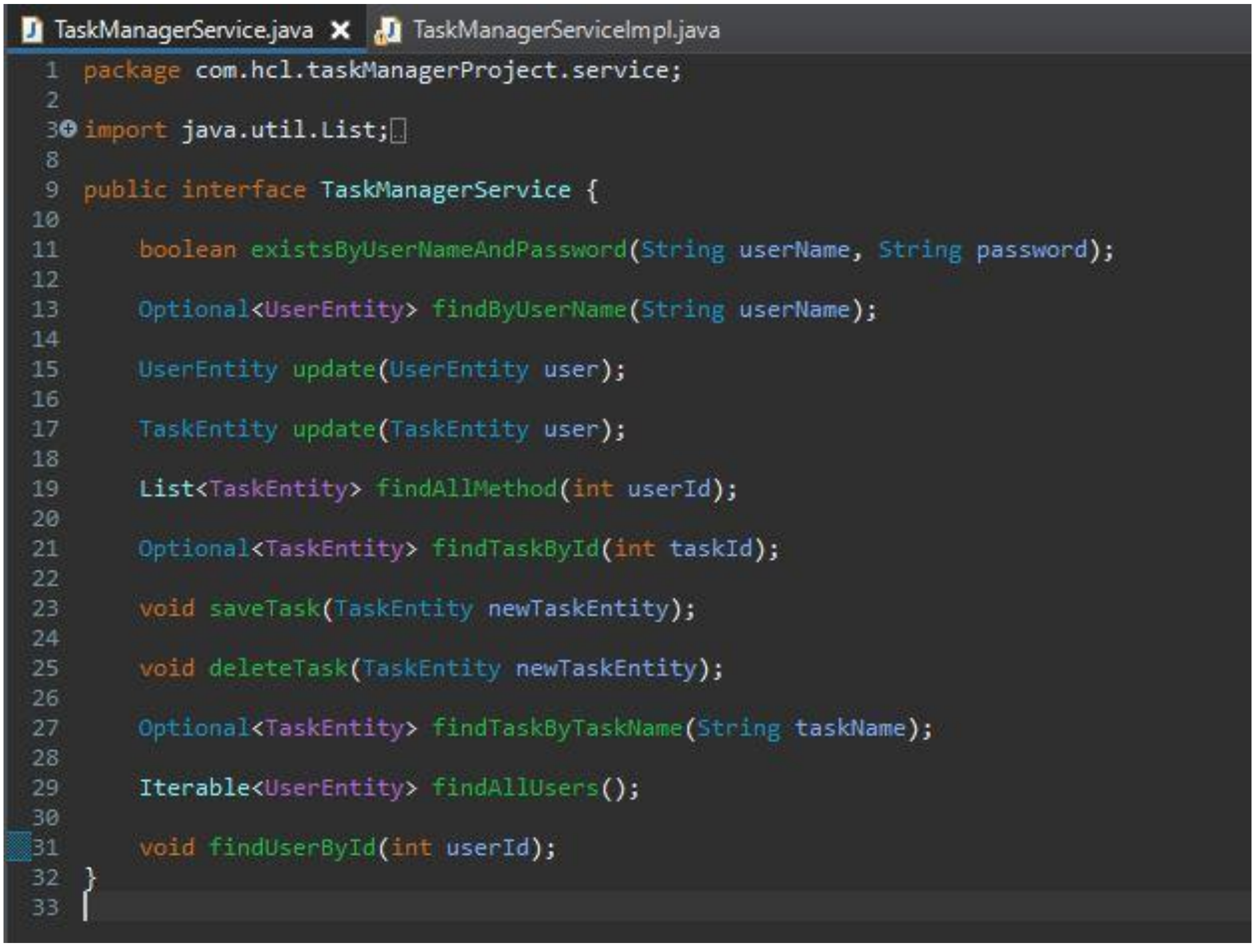
```
            throw new SaveFailedException();
        }
    }

    @RequestMapping("/viewTasks")
    public String viewTasks() {
        logger.info("Welcome Page Entered");
        return "welcome";
    }

    @RequestMapping("/deleteAnotherUser")
    public String deleteAnotherUser(ModelMap model) {
        logger.info("Admin Welcome Page Entered");
        Iterable<UserEntity> users = service.findAllUsers();
        model.addAttribute("listUsers", users);
        return "adminWelcome";
    }

    @GetMapping("/updateTask/{id}")
    public String updateTask(@PathVariable String id, ModelMap model) {
        logger.info("update task page entered");
        taskId = Integer.parseInt(id);
        Optional<TaskEntity> taskEntity = service.findTaskById(taskId);
        TaskEntity newTaskEntity = taskEntity.get();
        if (taskEntity.isPresent()) {
            model.addAttribute("taskId", newTaskEntity.getTaskId());
            model.addAttribute("taskName", newTaskEntity.getTaskName());
            model.addAttribute("startDate", newTaskEntity.getStartDate());
            model.addAttribute("endDate", newTaskEntity.getEndDate());
            model.addAttribute("description", newTaskEntity.getDescription());
            model.addAttribute("email", newTaskEntity.getEmail());
            model.addAttribute("severity", newTaskEntity.getSeverity());
            model.addAttribute("taskId", newTaskEntity.getTaskId());
            return "/updateTask";
        } else {
            List<TaskEntity> tasks = service.findAllMethod(getUserIdMethod());
            model.addAttribute("listTasks", tasks);
            model.addAttribute("message", "Task Not found! Error!");
            return "/viewTasks";
        }
    }

    @PostMapping("/updateDone")
    public String updateDone(@ModelAttribute("updateDone") TaskEntity task, ModelMap model) {
        logger.info("updateDone entered");
        TaskEntity newTaskEntity;
        Optional<TaskEntity> taskEntity = service.findTaskById(task.getTaskId());
        newTaskEntity = taskEntity.get();
        if (taskEntity.isPresent()) {
            newTaskEntity.setDescription(task.getDescription());
            newTaskEntity.setEmail(task.getEmail());
            newTaskEntity.setEndDate(task.getEndDate());
            newTaskEntity.setSeverity(task.getSeverity());
            newTaskEntity.setStartDate(task.getStartDate());
```



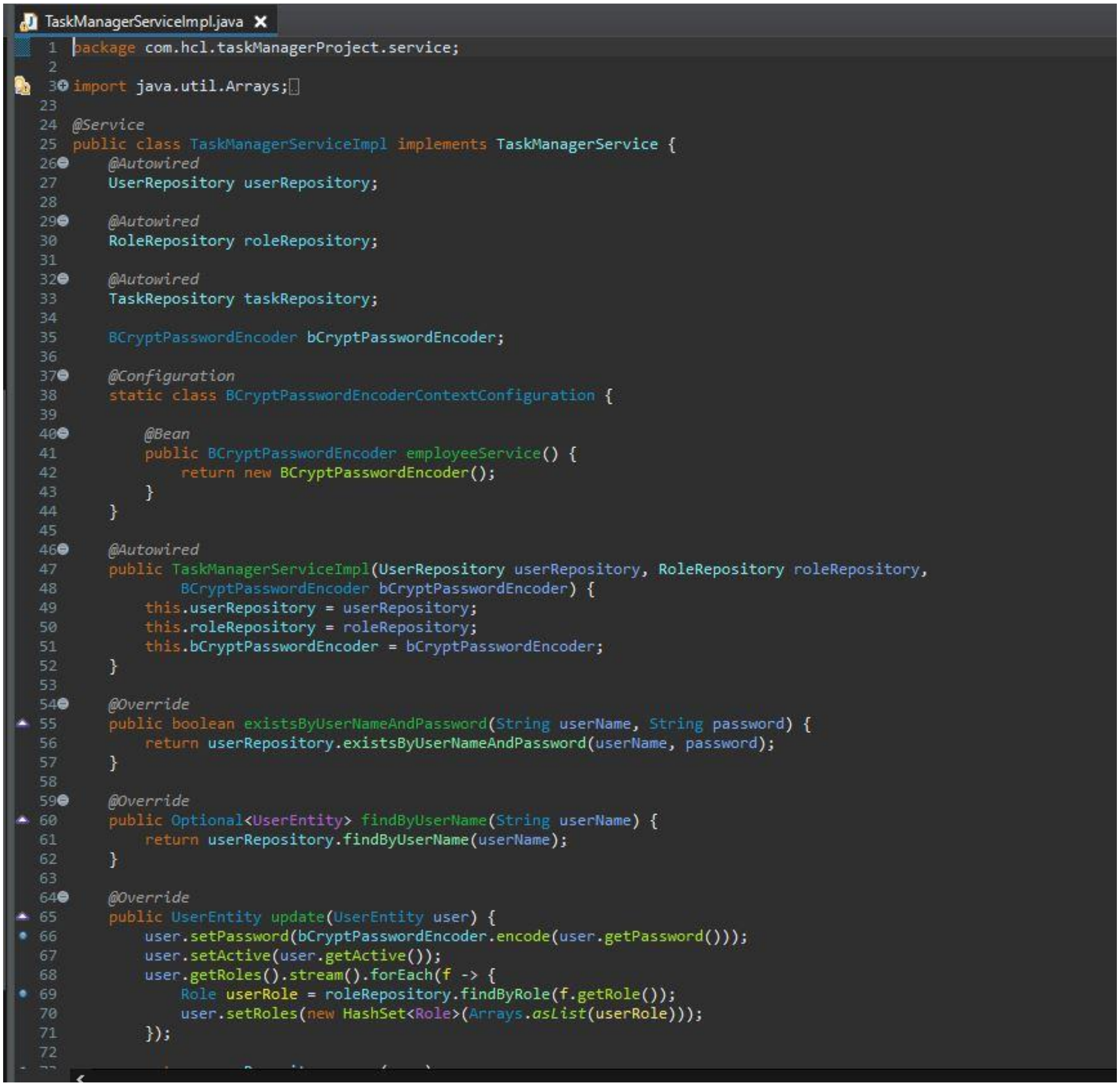
```java
            newTaskEntity.setStartDate(task.getStartDate());
            newTaskEntity.setTaskId(taskId);
            newTaskEntity.setUserId(getUserIdMethod());
            try {
                service.saveTask(newTaskEntity);
                List<TaskEntity> tasks = service.findAllMethod(getUserIdMethod());
                model.addAttribute("listTasks", tasks);
                model.addAttribute("message", "Task Information Updated");
                return "/viewTasks";
            } catch (Exception e) {
                List<TaskEntity> tasks = service.findAllMethod(getUserIdMethod());
                model.addAttribute("listTasks", tasks);
                model.addAttribute("message", "Error occured on Updation!");
                return "/viewTasks";
            }
        } else {
            List<TaskEntity> tasks = service.findAllMethod(getUserIdMethod());
            model.addAttribute("listTasks", tasks);
            model.addAttribute("message", "Task Not found! Error!");
            return "/viewTasks";
        }
    }

    @GetMapping("/deleteUser/{id}")
    public String deleteUser(@PathVariable String id, ModelMap model) {
        logger.info("DeleteUserDone entered");
        int userId;
        userId = Integer.parseInt(id);
        try {
            service.findUserById(userId);
            Iterable<UserEntity> users = service.findAllUsers();
            model.addAttribute("listUsers", users);
            model.addAttribute("message", "User Deleted");
            return "deleteAnotherUser";
        } catch (Exception e) {
            Iterable<UserEntity> users = service.findAllUsers();
            model.addAttribute("listUsers", users);
            model.addAttribute("message", "Error occured on Deletion of User!");
            return "deleteAnotherUser";
        }
    }

    @GetMapping("/deleteTask/{id}")
    public String deleteTask(@PathVariable String id, ModelMap model) {
        taskId = Integer.parseInt(id);
        Optional<TaskEntity> taskEntity = service.findTaskById(taskId);
        TaskEntity newTaskEntity = taskEntity.get();
        if (taskEntity.isPresent()) {
            model.addAttribute("taskId", newTaskEntity.getTaskId());
            model.addAttribute("taskName", newTaskEntity.getTaskName());
            model.addAttribute("startDate", newTaskEntity.getStartDate());
            model.addAttribute("endDate", newTaskEntity.getEndDate());
            model.addAttribute("description", newTaskEntity.getDescription());
            model.addAttribute("email", newTaskEntity.getEmail());
```



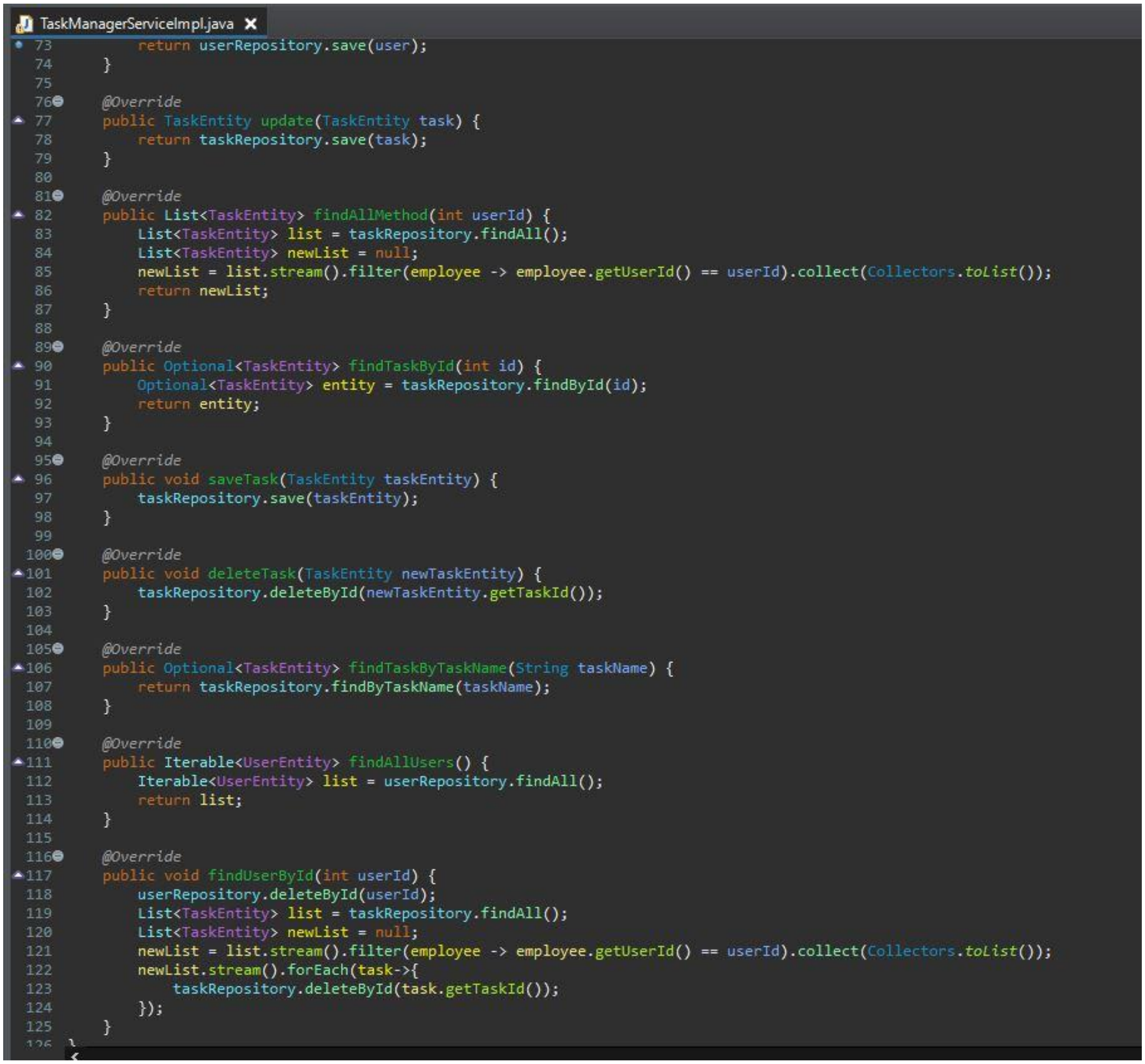
```java
            model.addAttribute("startDate", newTaskEntity.getStartDate());
            model.addAttribute("endDate", newTaskEntity.getEndDate());
            model.addAttribute("description", newTaskEntity.getDescription());
            model.addAttribute("email", newTaskEntity.getEmail());
            model.addAttribute("severity", newTaskEntity.getSeverity());
            model.addAttribute("taskId", newTaskEntity.getTaskId());
            return "/deleteTask";
        } else {
            List<TaskEntity> tasks = service.findAllMethod(getUserIdMethod());
            model.addAttribute("listTasks", tasks);
            model.addAttribute("message", "Task Not found! Error!");
            return "/viewTasks";
        }
    }

    @PostMapping("/deleteDone")
    public String deleteDone(@ModelAttribute("deleteDone") TaskEntity task, ModelMap model) {
        logger.info("updateDone entered");
        TaskEntity newTaskEntity;
        Optional<TaskEntity> taskEntity = service.findTaskById(task.getTaskId());
        newTaskEntity = taskEntity.get();
        if (taskEntity.isPresent()) {
            newTaskEntity.setDescription(task.getDescription());
            newTaskEntity.setEmail(task.getEmail());
            newTaskEntity.setEndDate(task.getEndDate());
            newTaskEntity.setSeverity(task.getSeverity());
            newTaskEntity.setStartDate(task.getStartDate());
            newTaskEntity.setTaskId(taskId);
            newTaskEntity.setUserId(getUserIdMethod());
            try {
                service.deleteTask(newTaskEntity);
                List<TaskEntity> tasks = service.findAllMethod(getUserIdMethod());
                model.addAttribute("message", "Task Deleted");
                model.addAttribute("listTasks", tasks);
                return "/viewTasks";
            } catch (Exception e) {
                List<TaskEntity> tasks = service.findAllMethod(getUserIdMethod());
                model.addAttribute("listTasks", tasks);
                model.addAttribute("message", "Error occured on Deletion!");
                return "/viewTasks";
            }
        } else {
            List<TaskEntity> tasks = service.findAllMethod(getUserIdMethod());
            model.addAttribute("listTasks", tasks);
            model.addAttribute("message", "Task Not found! Error!");
            return "/viewTasks";
        }
    }

    @RequestMapping("/403")
    public String accessdenied() {
        return "403";
    }
}
```



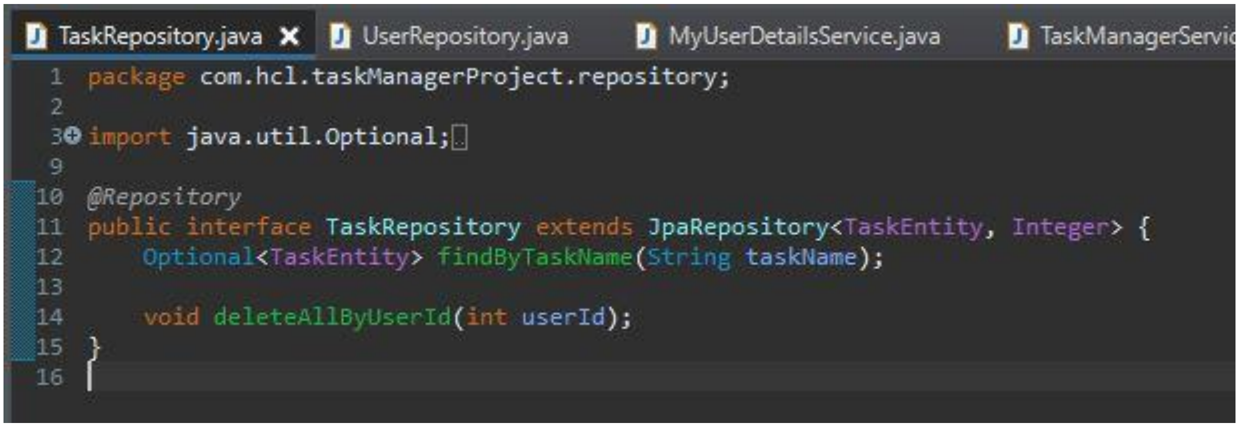
```java
package com.hcl.taskManagerProject.service;

import java.util.List;

public interface TaskManagerService {

    boolean existsByUserNameAndPassword(String userName, String password);

    Optional<UserEntity> findByUserName(String userName);

    UserEntity update(UserEntity user);

    TaskEntity update(TaskEntity user);

    List<TaskEntity> findAllMethod(int userId);

    Optional<TaskEntity> findTaskById(int taskId);

    void saveTask(TaskEntity newTaskEntity);

    void deleteTask(TaskEntity newTaskEntity);

    Optional<TaskEntity> findTaskByTaskName(String taskName);

    Iterable<UserEntity> findAllUsers();

    void findUserById(int userId);
}
```



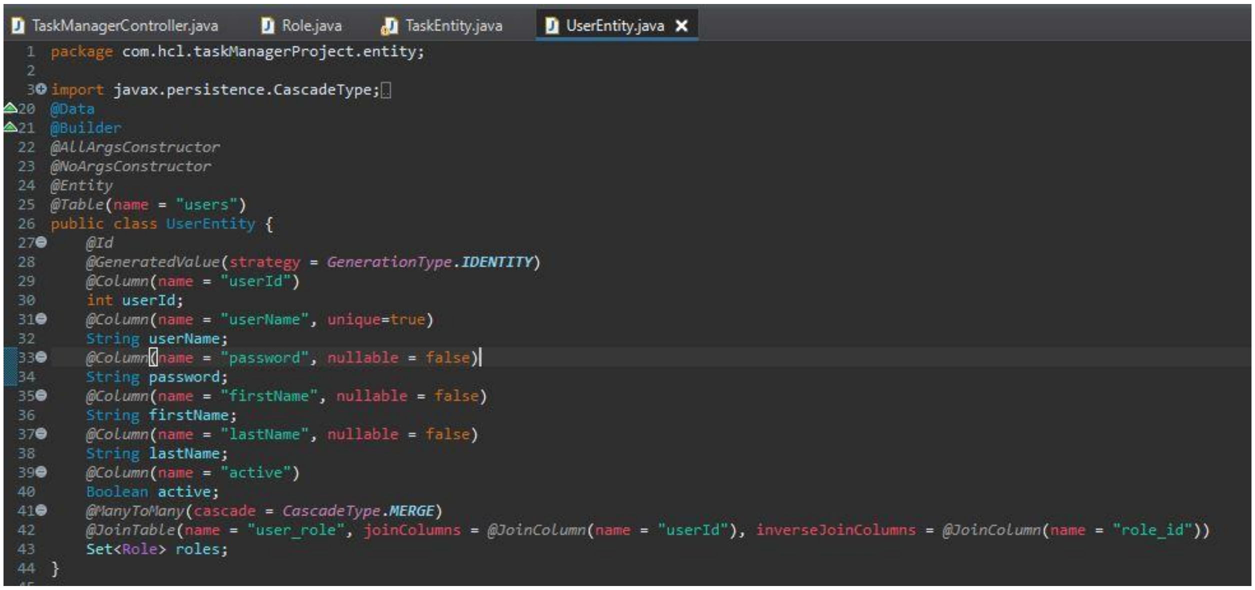
```java
package com.hcl.taskManagerProject.service;

import java.util.Arrays;

@Service
public class TaskManagerServiceImpl implements TaskManagerService {
    @Autowired
    UserRepository userRepository;

    @Autowired
    RoleRepository roleRepository;

    @Autowired
    TaskRepository taskRepository;

    BCryptPasswordEncoder bCryptPasswordEncoder;

    @Configuration
    static class BCryptPasswordEncoderContextConfiguration {

        @Bean
        public BCryptPasswordEncoder employeeService() {
            return new BCryptPasswordEncoder();
        }
    }

    @Autowired
    public TaskManagerServiceImpl(UserRepository userRepository, RoleRepository roleRepository,
            BCryptPasswordEncoder bCryptPasswordEncoder) {
        this.userRepository = userRepository;
        this.roleRepository = roleRepository;
        this.bCryptPasswordEncoder = bCryptPasswordEncoder;
    }

    @Override
    public boolean existsByUserNameAndPassword(String userName, String password) {
        return userRepository.existsByUserNameAndPassword(userName, password);
    }

    @Override
    public Optional<UserEntity> findByUserName(String userName) {
        return userRepository.findByUserName(userName);
    }

    @Override
    public UserEntity update(UserEntity user) {
        user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));
        user.setActive(user.getActive());
        user.getRoles().stream().forEach(f -> {
            Role userRole = roleRepository.findByRole(f.getRole());
            user.setRoles(new HashSet<Role>(Arrays.asList(userRole)));
        });

```



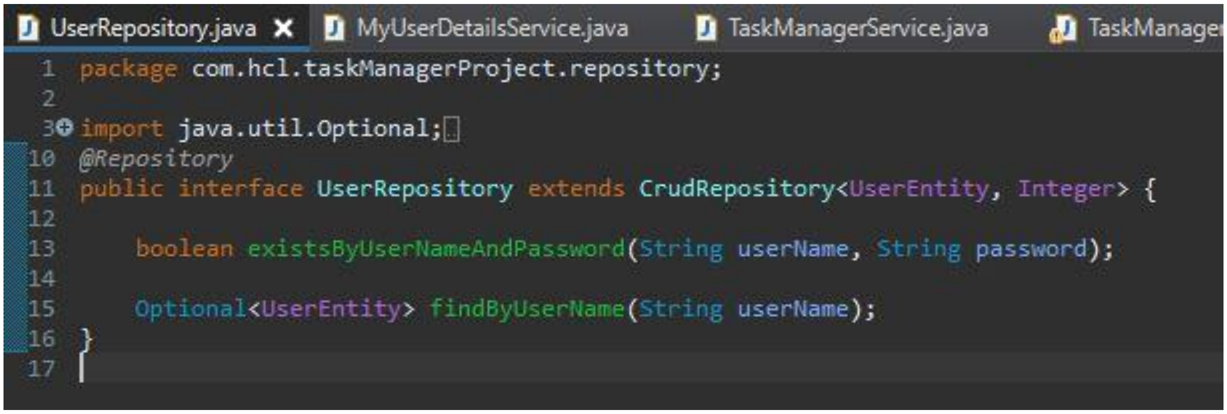
```java
        return userRepository.save(user);
    }

    @Override
    public TaskEntity update(TaskEntity task) {
        return taskRepository.save(task);
    }

    @Override
    public List<TaskEntity> findAllMethod(int userId) {
        List<TaskEntity> list = taskRepository.findAll();
        List<TaskEntity> newList = null;
        newList = list.stream().filter(employee -> employee.getUserId() == userId).collect(Collectors.toList());
        return newList;
    }

    @Override
    public Optional<TaskEntity> findTaskById(int id) {
        Optional<TaskEntity> entity = taskRepository.findById(id);
        return entity;
    }

    @Override
    public void saveTask(TaskEntity taskEntity) {
        taskRepository.save(taskEntity);
    }

    @Override
    public void deleteTask(TaskEntity newTaskEntity) {
        taskRepository.deleteById(newTaskEntity.getTaskId());
    }

    @Override
    public Optional<TaskEntity> findTaskByTaskName(String taskName) {
        return taskRepository.findByTaskName(taskName);
    }

    @Override
    public Iterable<UserEntity> findAllUsers() {
        Iterable<UserEntity> list = userRepository.findAll();
        return list;
    }

    @Override
    public void findUserById(int userId) {
        userRepository.deleteById(userId);
        List<TaskEntity> list = taskRepository.findAll();
        List<TaskEntity> newList = null;
        newList = list.stream().filter(employee -> employee.getUserId() == userId).collect(Collectors.toList());
        newList.stream().forEach(task->{
            taskRepository.deleteById(task.getTaskId());
        });
    }
}
```



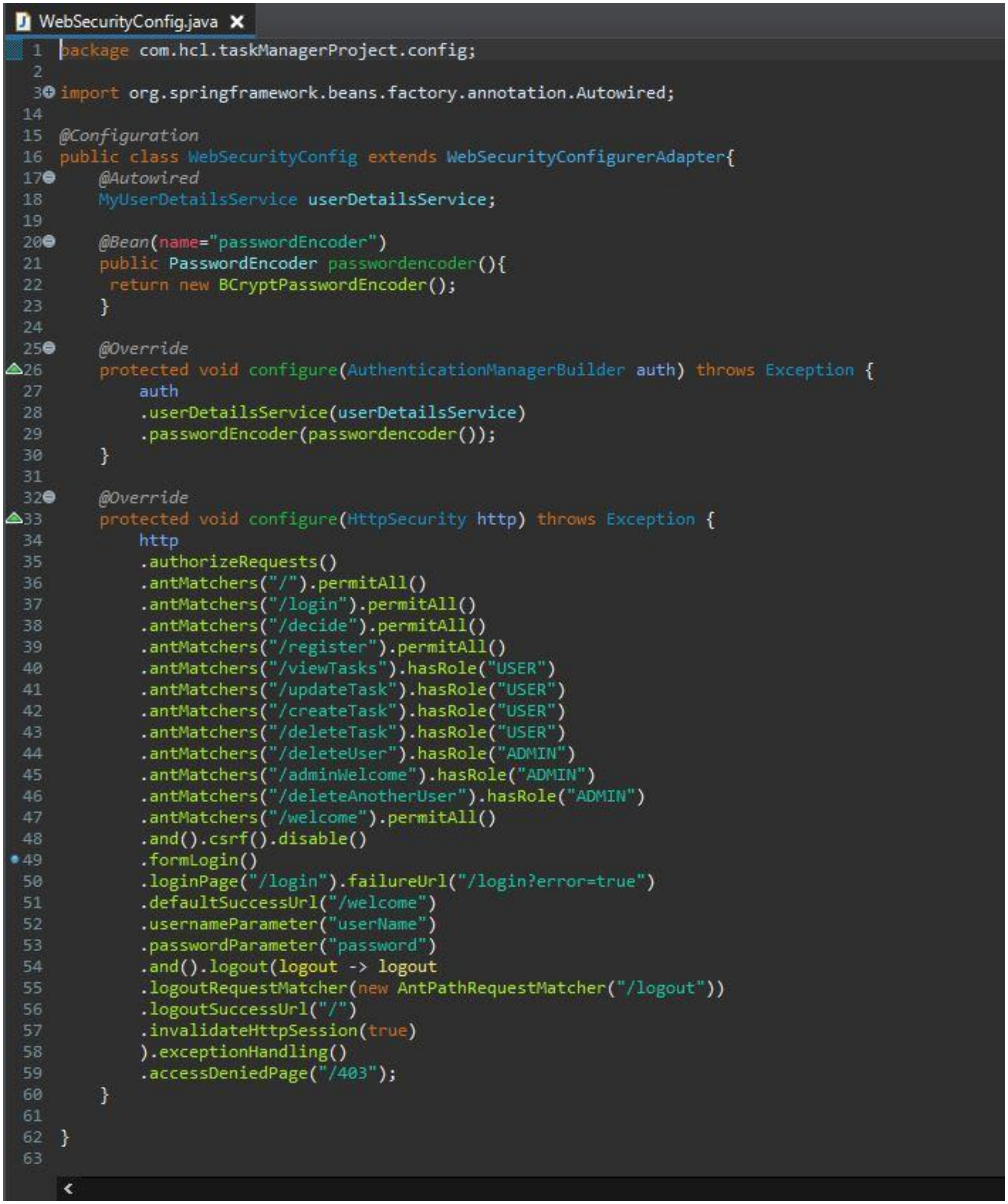
```java
package com.hcl.taskManagerProject.repository;

import java.util.Optional;

@Repository
public interface TaskRepository extends JpaRepository<TaskEntity, Integer> {
    Optional<TaskEntity> findByTaskName(String taskName);

    void deleteAllByUserId(int userId);
}
```



```java
package com.hcl.taskManagerProject.entity;

import javax.persistence.CascadeType;
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "users")
public class UserEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "userId")
    int userId;
    @Column(name = "userName", unique=true)
    String userName;
    @Column(name = "password", nullable = false)
    String password;
    @Column(name = "firstName", nullable = false)
    String firstName;
    @Column(name = "lastName", nullable = false)
    String lastName;
    @Column(name = "active")
    Boolean active;
    @ManyToMany(cascade = CascadeType.MERGE)
    @JoinTable(name = "user_role", joinColumns = @JoinColumn(name = "userId"), inverseJoinColumns = @JoinColumn(name = "role_id"))
    Set<Role> roles;
}
```



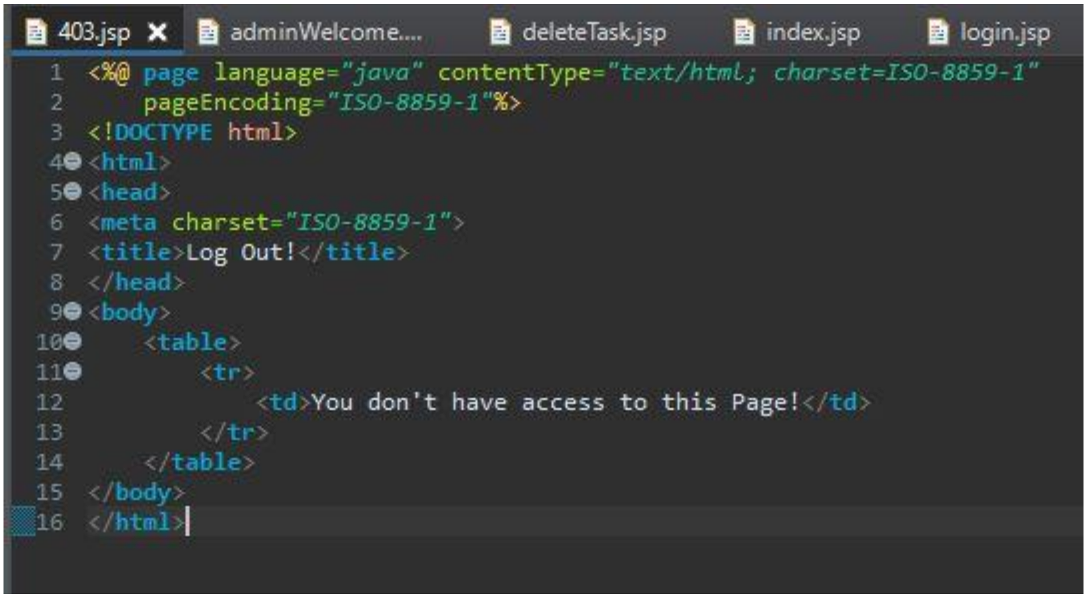
```
package com.hcl.taskManagerProject.repository;

import java.util.Optional;
@Repository
public interface UserRepository extends CrudRepository<UserEntity, Integer> {

    boolean existsByUserNameAndPassword(String userName, String password);

    Optional<UserEntity> findByUserName(String userName);
}

```



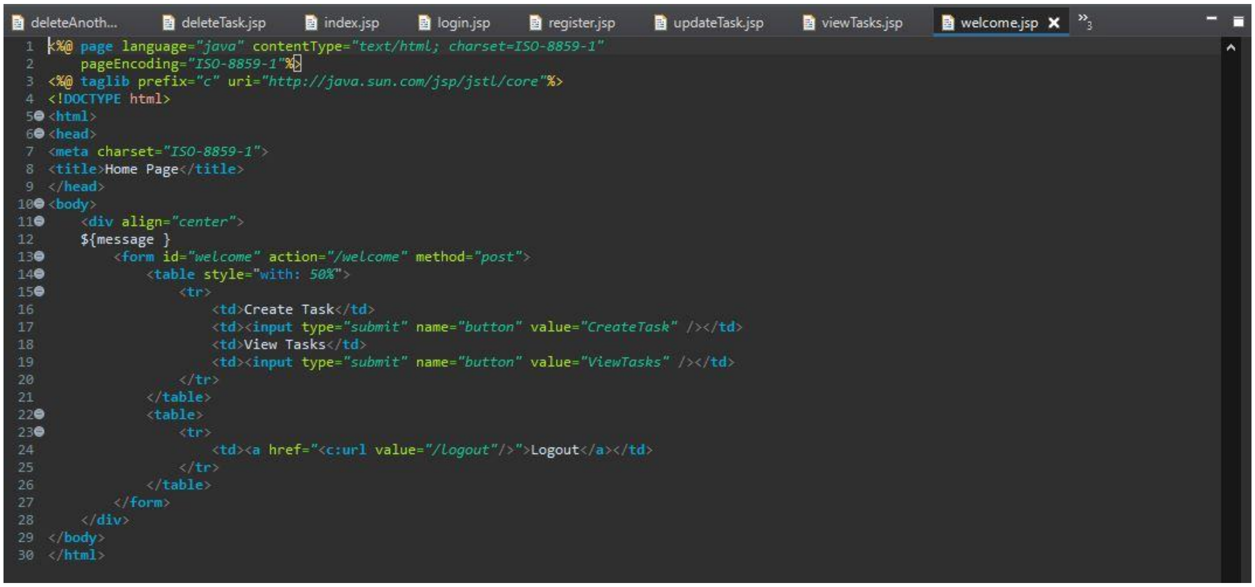
WebSecurityConfig.java

```java
package com.hcl.taskManagerProject.config;

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

@Configuration
public class WebSecurityConfig extends WebSecurityConfigurerAdapter{
    @Autowired
    MyUserDetailsService userDetailsService;

    @Bean(name="passwordEncoder")
    public PasswordEncoder passwordencoder(){
     return new BCryptPasswordEncoder();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth
        .userDetailsService(userDetailsService)
        .passwordEncoder(passwordencoder());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
        .authorizeRequests()
        .antMatchers("/").permitAll()
        .antMatchers("/login").permitAll()
        .antMatchers("/decide").permitAll()
        .antMatchers("/register").permitAll()
        .antMatchers("/viewTasks").hasRole("USER")
        .antMatchers("/updateTask").hasRole("USER")
        .antMatchers("/createTask").hasRole("USER")
        .antMatchers("/deleteTask").hasRole("USER")
        .antMatchers("/deleteUser").hasRole("ADMIN")
        .antMatchers("/adminWelcome").hasRole("ADMIN")
        .antMatchers("/deleteAnotherUser").hasRole("ADMIN")
        .antMatchers("/welcome").permitAll()
        .and().csrf().disable()
        .formLogin()
        .loginPage("/login").failureUrl("/login?error=true")
        .defaultSuccessUrl("/welcome")
        .usernameParameter("userName")
        .passwordParameter("password")
        .and().logout(logout -> logout
        .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
        .logoutSuccessUrl("/")
        .invalidateHttpSession(true)
        ).exceptionHandling()
        .accessDeniedPage("/403");
    }

}
```

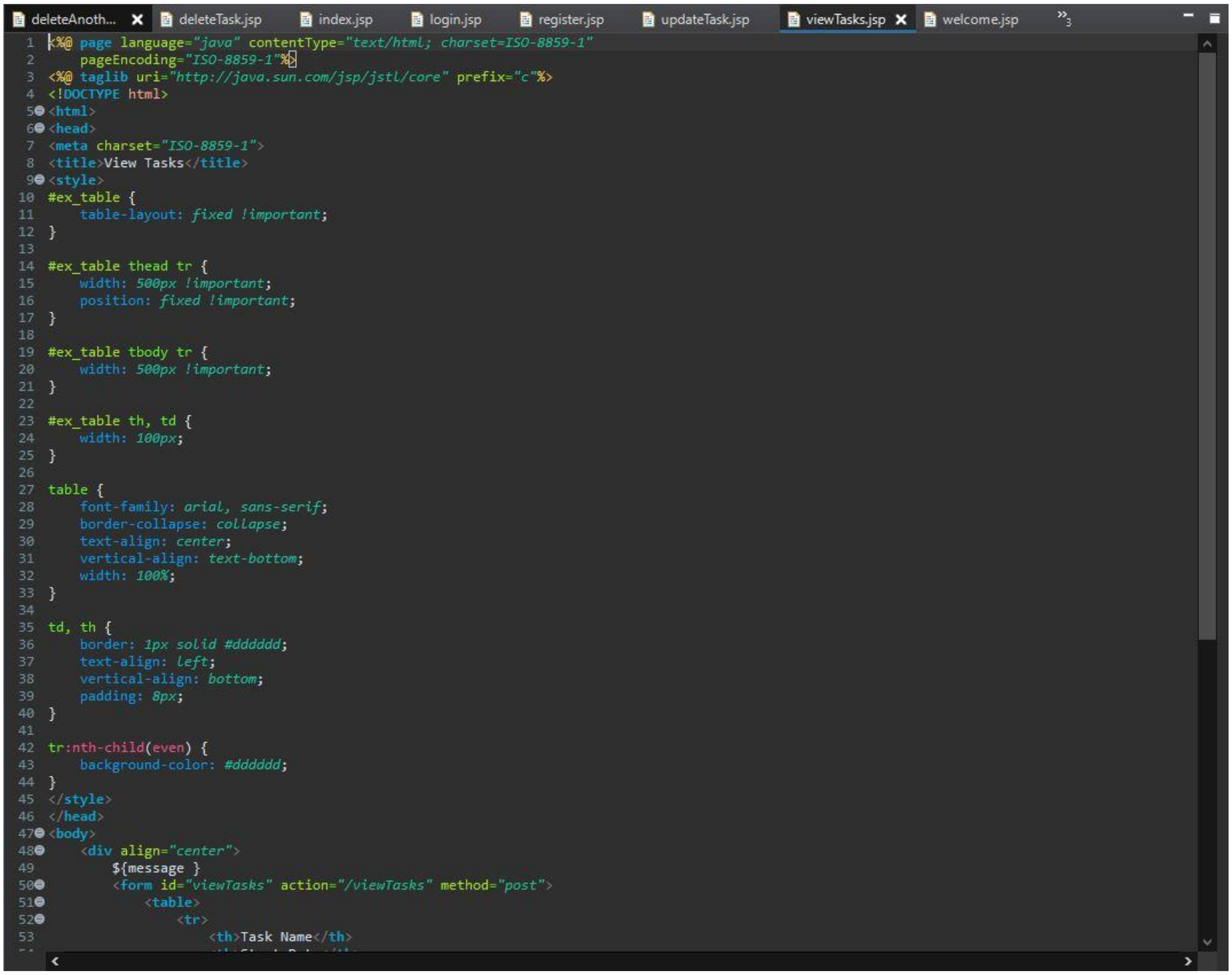
JSP:



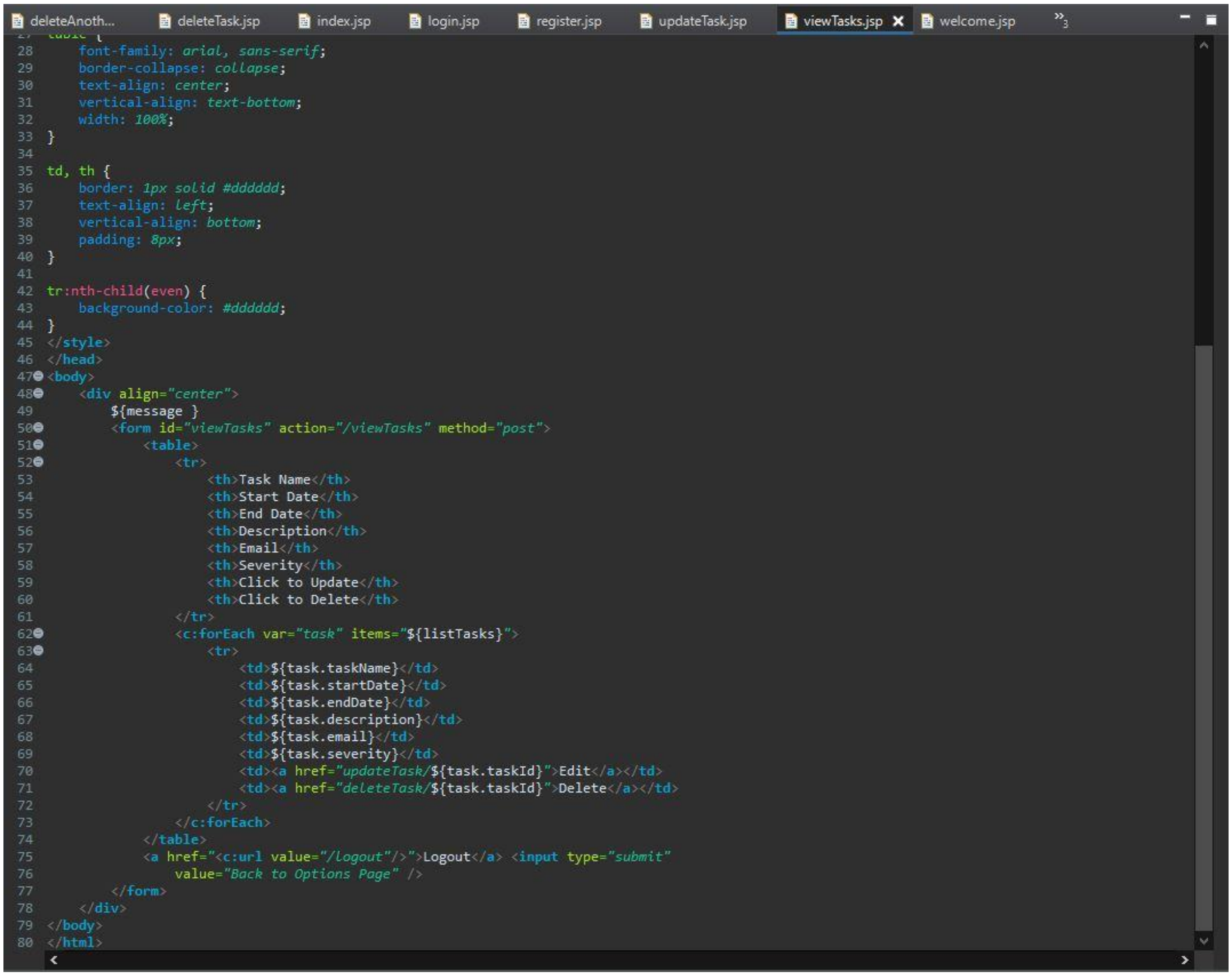
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Log Out!</title>
</head>
<body>
    <table>
        <tr>
            <td>You don't have access to this Page!</td>
        </tr>
    </table>
</body>
</html>
```



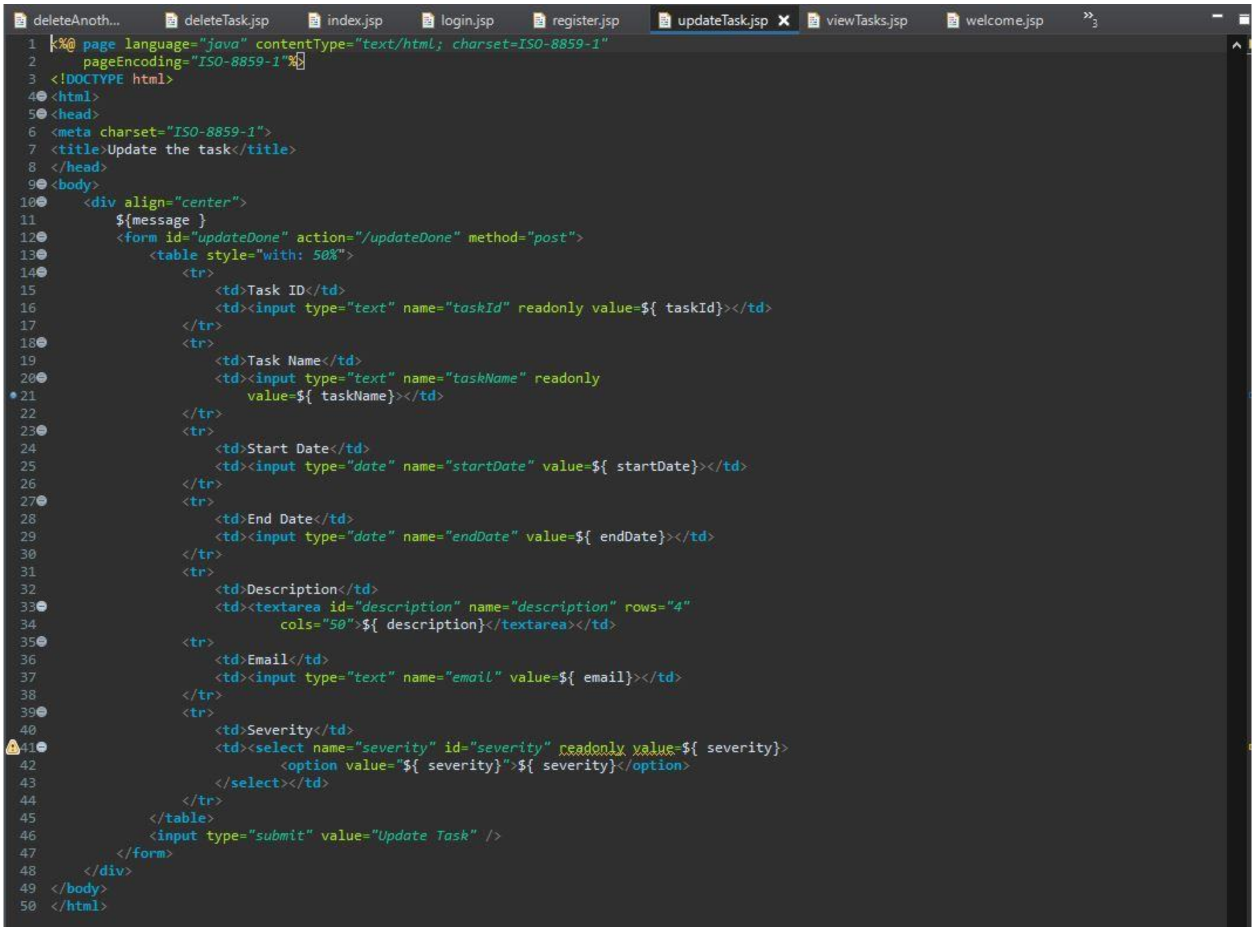
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Home Page</title>
</head>
<body>
    <div align="center">
    ${message }
        <form id="welcome" action="/welcome" method="post">
            <table style="with: 50%">
                <tr>
                    <td>Create Task</td>
                    <td><input type="submit" name="button" value="CreateTask" /></td>
                    <td>View Tasks</td>
                    <td><input type="submit" name="button" value="ViewTasks" /></td>
                </tr>
            </table>
            <table>
                <tr>
                    <td><a href="<c:url value="/logout"/>">Logout</a></td>
                </tr>
            </table>
        </form>
    </div>
</body>
</html>
```



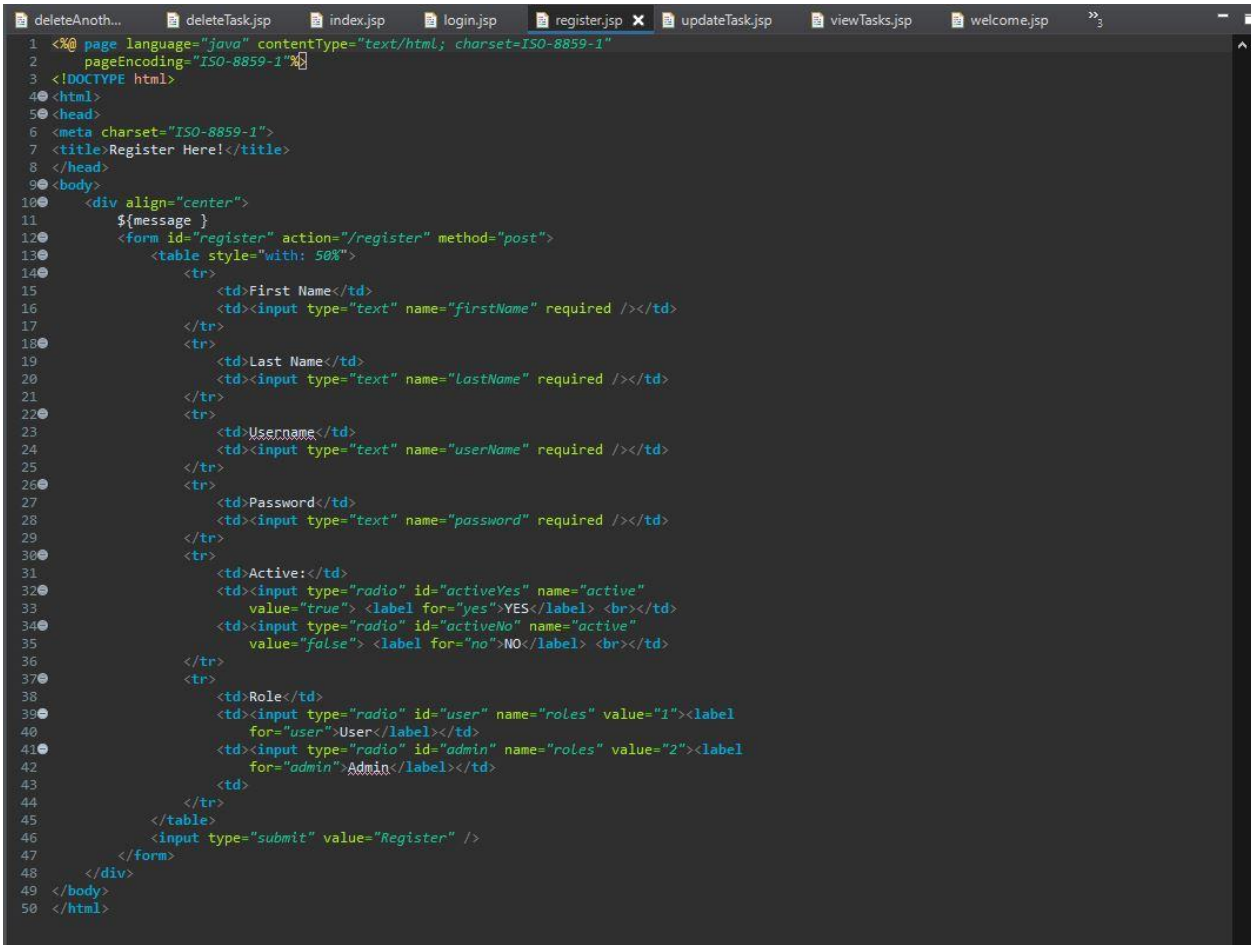
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>View Tasks</title>
<style>
#ex_table {
    table-layout: fixed !important;
}

#ex_table thead tr {
    width: 500px !important;
    position: fixed !important;
}

#ex_table tbody tr {
    width: 500px !important;
}

#ex_table th, td {
    width: 100px;
}

table {
    font-family: arial, sans-serif;
    border-collapse: collapse;
    text-align: center;
    vertical-align: text-bottom;
    width: 100%;
}

td, th {
    border: 1px solid #dddddd;
    text-align: left;
    vertical-align: bottom;
    padding: 8px;
}

tr:nth-child(even) {
    background-color: #dddddd;
}
</style>
</head>
<body>
    <div align="center">
        ${message }
        <form id="viewTasks" action="/viewTasks" method="post">
            <table>
                <tr>
                    <th>Task Name</th>
```



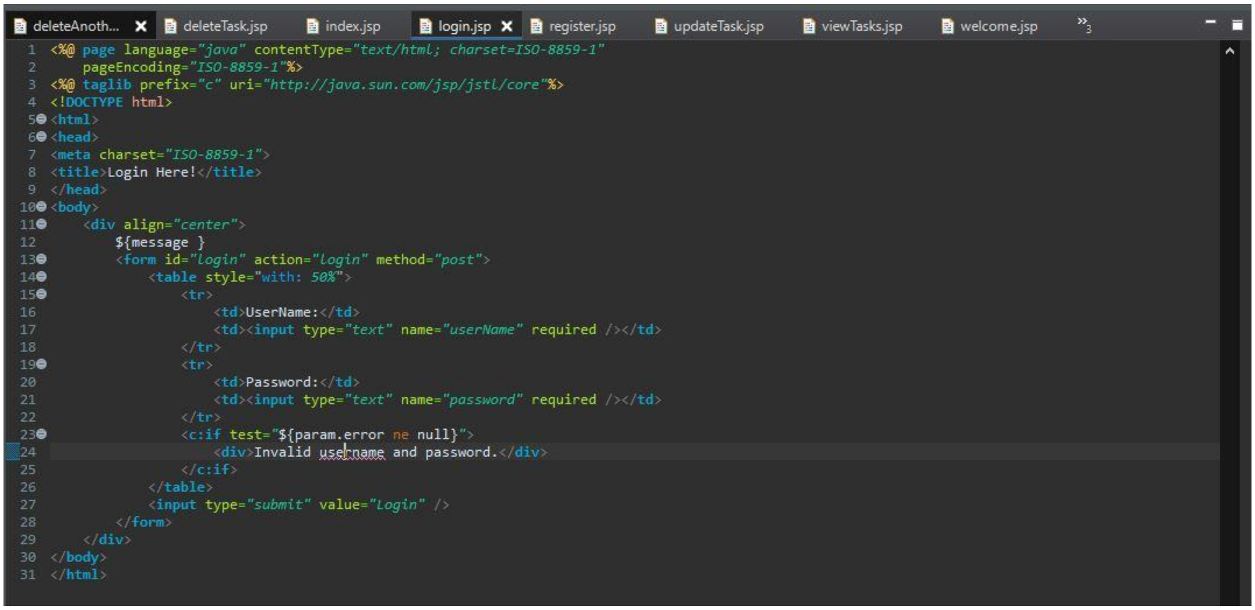
```
table {
    font-family: arial, sans-serif;
    border-collapse: collapse;
    text-align: center;
    vertical-align: text-bottom;
    width: 100%;
}

td, th {
    border: 1px solid #dddddd;
    text-align: left;
    vertical-align: bottom;
    padding: 8px;
}

tr:nth-child(even) {
    background-color: #dddddd;
}
</style>
</head>
<body>
    <div align="center">
        ${message }
        <form id="viewTasks" action="/viewTasks" method="post">
            <table>
                <tr>
                    <th>Task Name</th>
                    <th>Start Date</th>
                    <th>End Date</th>
                    <th>Description</th>
                    <th>Email</th>
                    <th>Severity</th>
                    <th>Click to Update</th>
                    <th>Click to Delete</th>
                </tr>
                <c:forEach var="task" items="${listTasks}">
                    <tr>
                        <td>${task.taskName}</td>
                        <td>${task.startDate}</td>
                        <td>${task.endDate}</td>
                        <td>${task.description}</td>
                        <td>${task.email}</td>
                        <td>${task.severity}</td>
                        <td><a href="updateTask/${task.taskId}">Edit</a></td>
                        <td><a href="deleteTask/${task.taskId}">Delete</a></td>
                    </tr>
                </c:forEach>
            </table>
            <a href="<c:url value="/logout"/>">Logout</a> <input type="submit"
                value="Back to Options Page" />
        </form>
    </div>
</body>
</html>
```



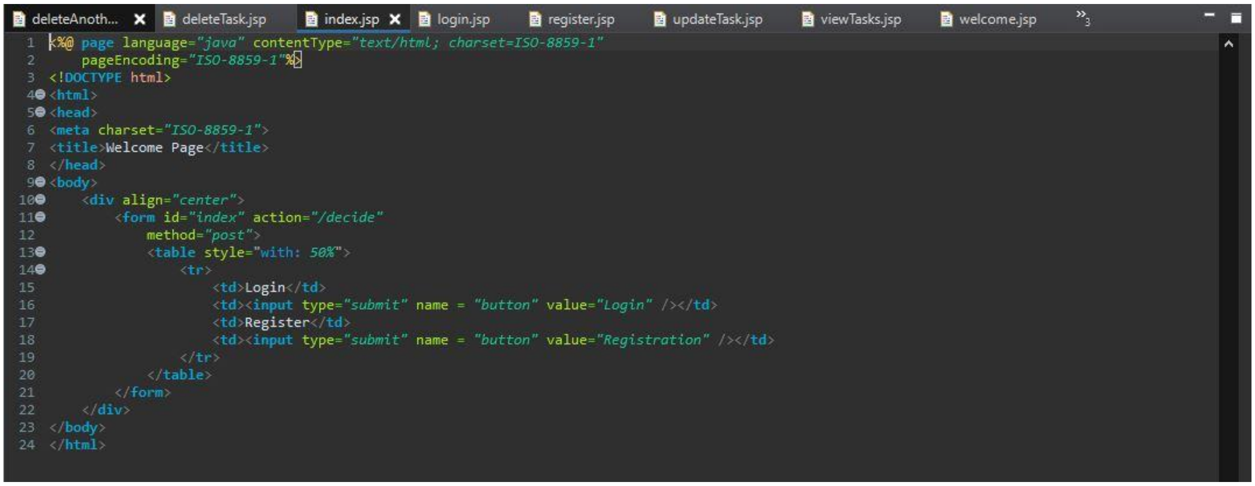
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Update the task</title>
</head>
<body>
    <div align="center">
        ${message }
        <form id="updateDone" action="/updateDone" method="post">
            <table style="with: 50%">
                <tr>
                    <td>Task ID</td>
                    <td><input type="text" name="taskId" readonly value=${ taskId}></td>
                </tr>
                <tr>
                    <td>Task Name</td>
                    <td><input type="text" name="taskName" readonly
                        value=${ taskName}></td>
                </tr>
                <tr>
                    <td>Start Date</td>
                    <td><input type="date" name="startDate" value=${ startDate}></td>
                </tr>
                <tr>
                    <td>End Date</td>
                    <td><input type="date" name="endDate" value=${ endDate}></td>
                </tr>
                <tr>
                    <td>Description</td>
                    <td><textarea id="description" name="description" rows="4"
                            cols="50">${ description}</textarea></td>
                <tr>
                    <td>Email</td>
                    <td><input type="text" name="email" value=${ email}></td>
                </tr>
                <tr>
                    <td>Severity</td>
                    <td><select name="severity" id="severity" readonly value=${ severity}>
                            <option value="${ severity}">${ severity}</option>
                    </select></td>
                </tr>
            </table>
            <input type="submit" value="Update Task" />
        </form>
    </div>
</body>
</html>
```



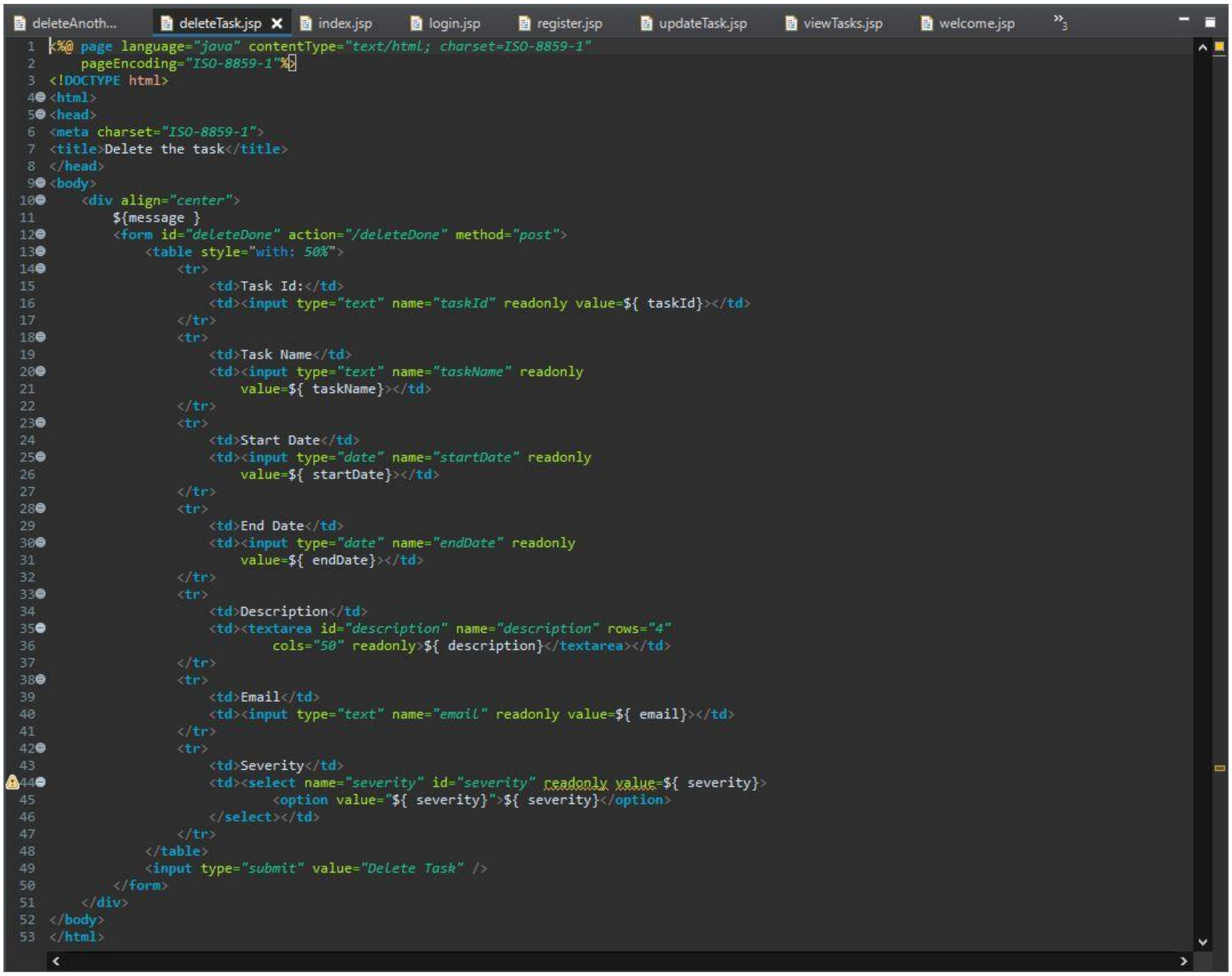
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Register Here!</title>
</head>
<body>
    <div align="center">
        ${message }
        <form id="register" action="/register" method="post">
            <table style="with: 50%">
                <tr>
                    <td>First Name</td>
                    <td><input type="text" name="firstName" required /></td>
                </tr>
                <tr>
                    <td>Last Name</td>
                    <td><input type="text" name="lastName" required /></td>
                </tr>
                <tr>
                    <td>Username</td>
                    <td><input type="text" name="userName" required /></td>
                </tr>
                <tr>
                    <td>Password</td>
                    <td><input type="text" name="password" required /></td>
                </tr>
                <tr>
                    <td>Active:</td>
                    <td><input type="radio" id="activeYes" name="active"
                        value="true"> <label for="yes">YES</label> <br></td>
                    <td><input type="radio" id="activeNo" name="active"
                        value="false"> <label for="no">NO</label> <br></td>
                </tr>
                <tr>
                    <td>Role</td>
                    <td><input type="radio" id="user" name="roles" value="1"><label
                        for="user">User</label></td>
                    <td><input type="radio" id="admin" name="roles" value="2"><label
                        for="admin">Admin</label></td>
                    <td>
                </tr>
            </table>
            <input type="submit" value="Register" />
        </form>
    </div>
</body>
</html>
```



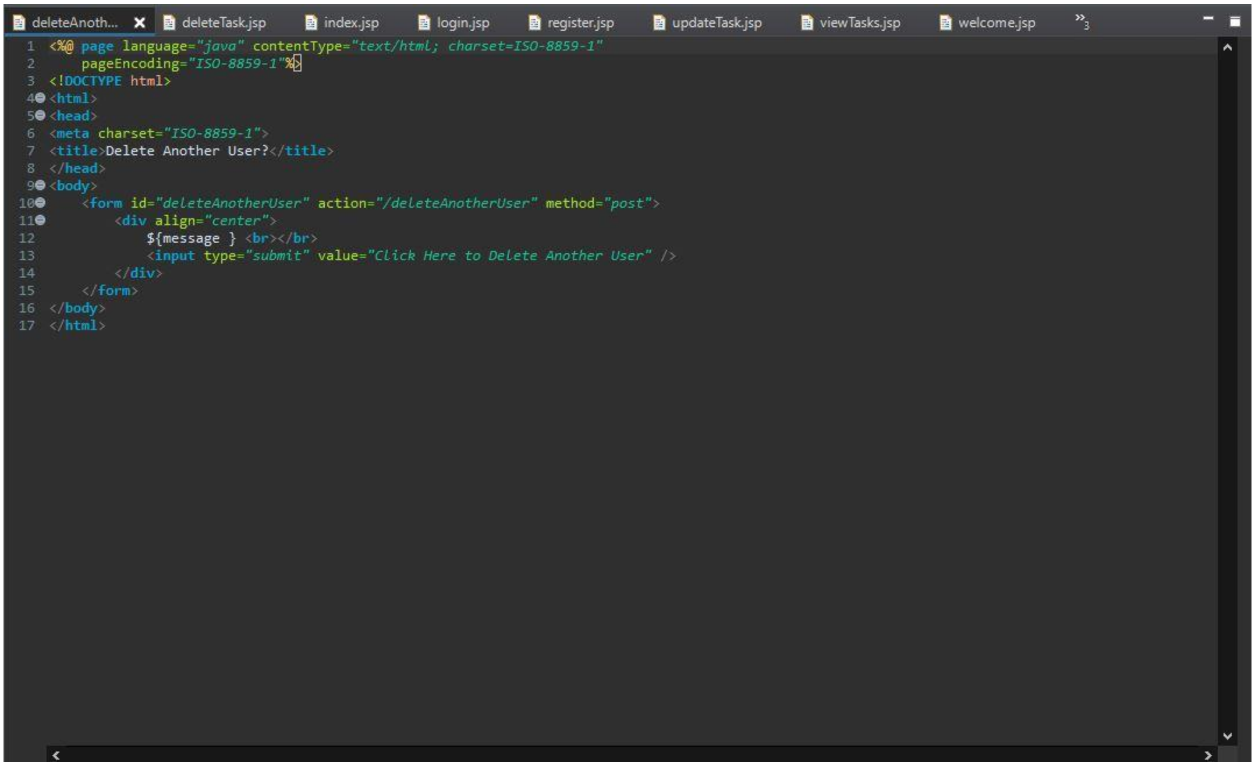
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Login Here!</title>
</head>
<body>
    <div align="center">
        ${message }
        <form id="login" action="login" method="post">
            <table style="with: 50%">
                <tr>
                    <td>UserName:</td>
                    <td><input type="text" name="userName" required /></td>
                </tr>
                <tr>
                    <td>Password:</td>
                    <td><input type="text" name="password" required /></td>
                </tr>
                <c:if test="${param.error ne null}">
                    <div>Invalid username and password.</div>
                </c:if>
            </table>
            <input type="submit" value="Login" />
        </form>
    </div>
</body>
</html>
```



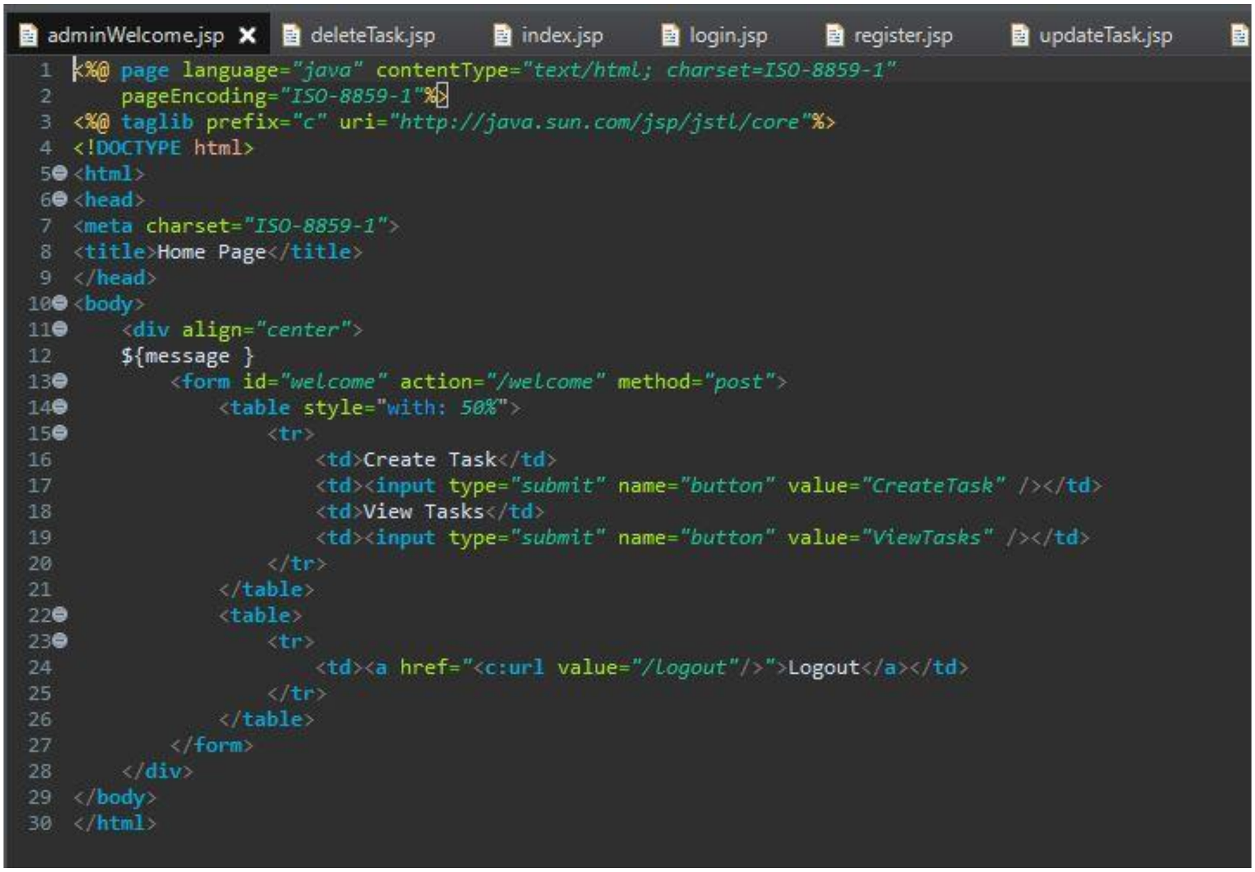
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Welcome Page</title>
</head>
<body>
    <div align="center">
        <form id="index" action="/decide"
            method="post">
            <table style="with: 50%">
                <tr>
                    <td>Login</td>
                    <td><input type="submit" name = "button" value="Login" /></td>
                    <td>Register</td>
                    <td><input type="submit" name = "button" value="Registration" /></td>
                </tr>
            </table>
        </form>
    </div>
</body>
</html>
```



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Delete the task</title>
</head>
<body>
    <div align="center">
        ${message }
        <form id="deleteDone" action="/deleteDone" method="post">
            <table style="with: 50%">
                <tr>
                    <td>Task Id:</td>
                    <td><input type="text" name="taskId" readonly value=${ taskId}></td>
                </tr>
                <tr>
                    <td>Task Name</td>
                    <td><input type="text" name="taskName" readonly
                        value=${ taskName}></td>
                </tr>
                <tr>
                    <td>Start Date</td>
                    <td><input type="date" name="startDate" readonly
                        value=${ startDate}></td>
                </tr>
                <tr>
                    <td>End Date</td>
                    <td><input type="date" name="endDate" readonly
                        value=${ endDate}></td>
                </tr>
                <tr>
                    <td>Description</td>
                    <td><textarea id="description" name="description" rows="4"
                            cols="50" readonly>${ description}</textarea></td>
                </tr>
                <tr>
                    <td>Email</td>
                    <td><input type="text" name="email" readonly value=${ email}></td>
                </tr>
                <tr>
                    <td>Severity</td>
                    <td><select name="severity" id="severity" readonly value=${ severity}>
                            <option value="${ severity}">${ severity}</option>
                    </select></td>
                </tr>
            </table>
            <input type="submit" value="Delete Task" />
        </form>
    </div>
</body>
</html>
```

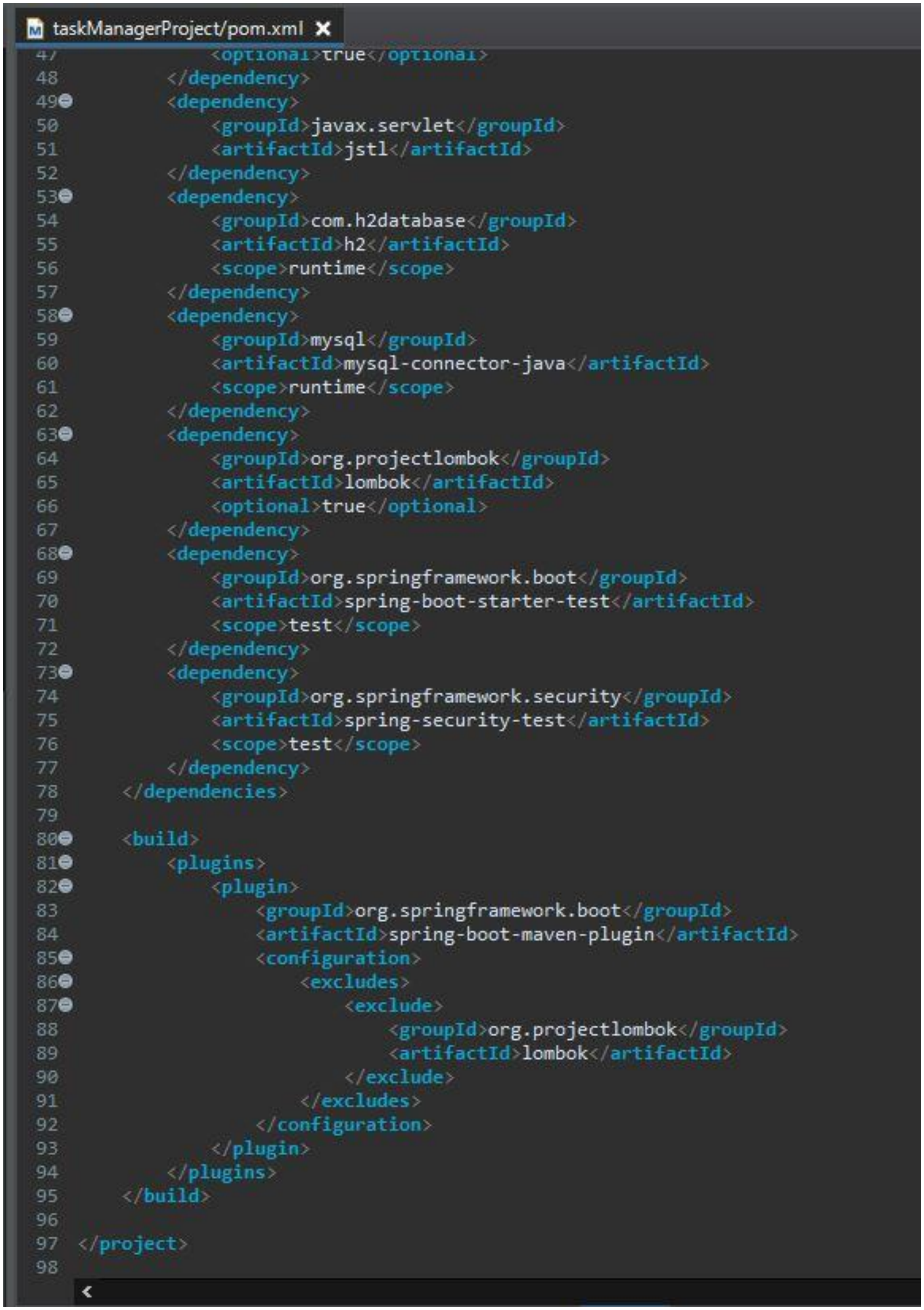


```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Delete Another User?</title>
</head>
<body>
    <form id="deleteAnotherUser" action="/deleteAnotherUser" method="post">
        <div align="center">
            ${message } <br></br>
            <input type="submit" value="Click Here to Delete Another User" />
        </div>
    </form>
</body>
</html>
```

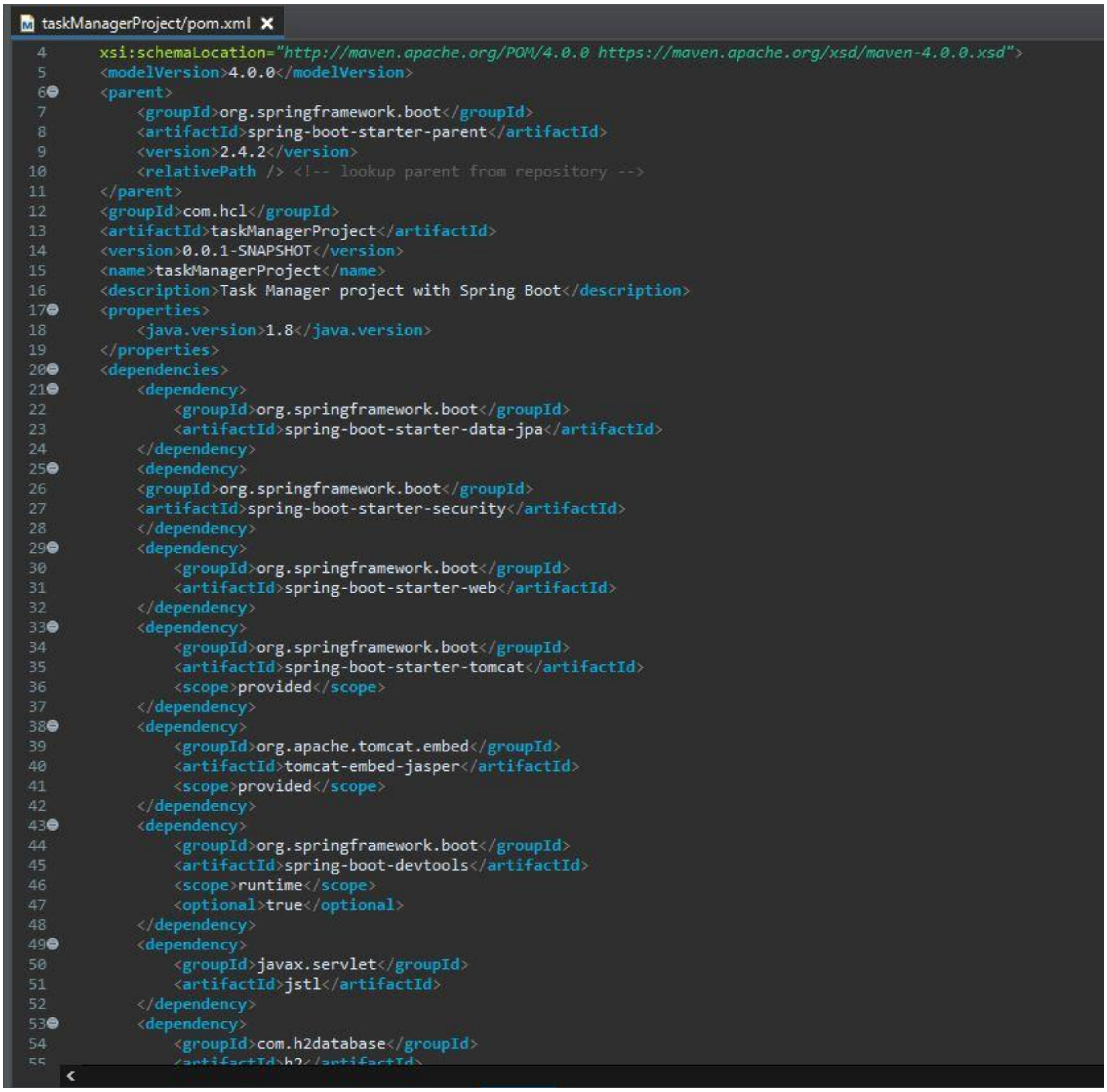


```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Home Page</title>
</head>
<body>
    <div align="center">
    ${message }
        <form id="welcome" action="/welcome" method="post">
            <table style="with: 50%">
                <tr>
                    <td>Create Task</td>
                    <td><input type="submit" name="button" value="CreateTask" /></td>
                    <td>View Tasks</td>
                    <td><input type="submit" name="button" value="ViewTasks" /></td>
                </tr>
            </table>
            <table>
                <tr>
                    <td><a href="<c:url value="/logout"/>">Logout</a></td>
                </tr>
            </table>
        </form>
    </div>
</body>
</html>
```

Pom:

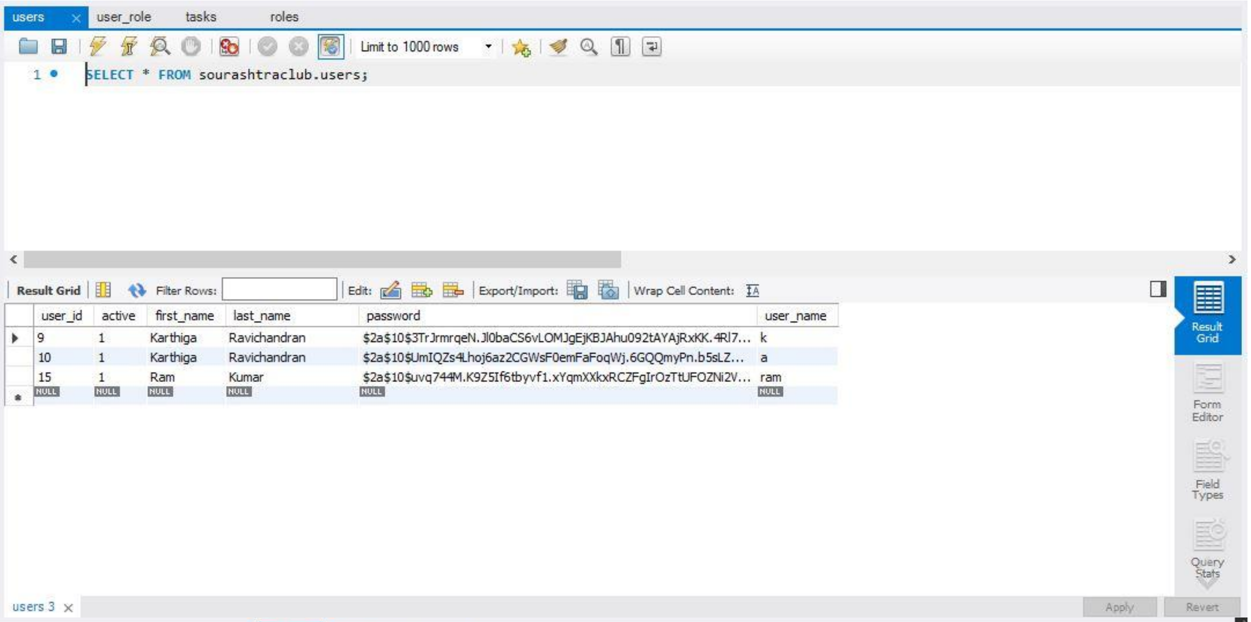


```xml
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>

```



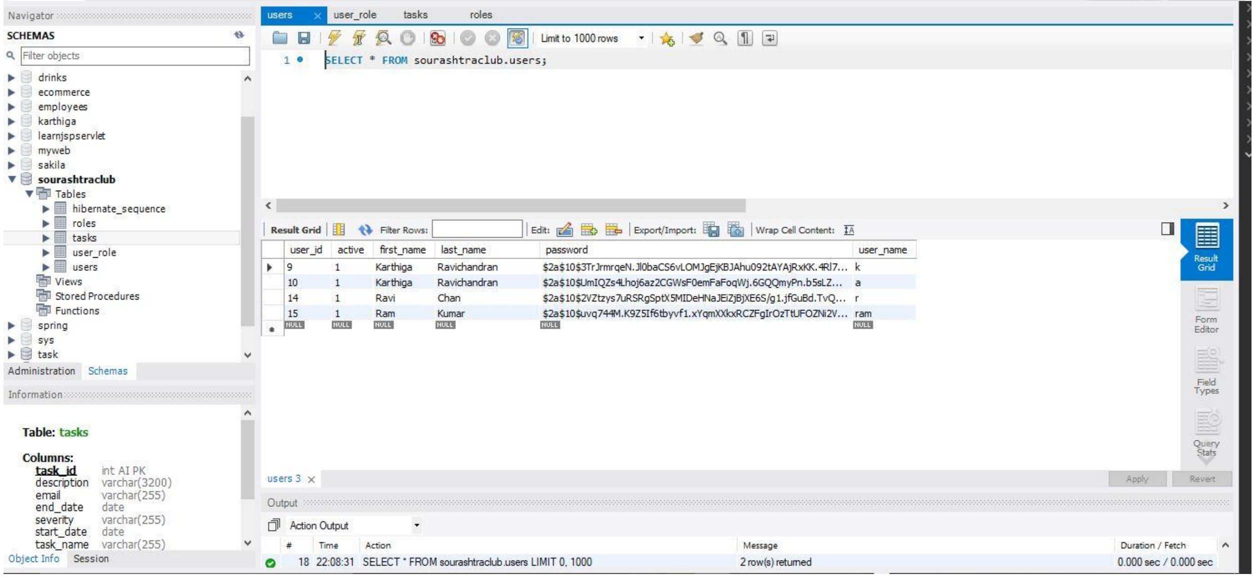
```
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.4.2</version>
        <relativePath /> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.hcl</groupId>
    <artifactId>taskManagerProject</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>taskManagerProject</name>
    <description>Task Manager project with Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.apache.tomcat.embed</groupId>
            <artifactId>tomcat-embed-jasper</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
```

WorkBench:



```
SELECT * FROM sourashtraclub.users;
```

user_id	active	first_name	last_name	password	user_name
9	1	Karthiga	Ravichandran	$2a$10$3TrJrmrqeN.Jl0baCS6vLOMJgEjKBJAhu092tAYAjRxKK.4Rl7...	k
10	1	Karthiga	Ravichandran	$2a$10$UmIQZs4Lhoj6az2CGWsF0emFaFoqWj.6GQQmyPn.b5sLZ...	a
15	1	Ram	Kumar	$2a$10$uvq744M.K9Z5If6tbyvf1.xYqmXXkxRCZFgIrOzTtUFOZNi2V...	ram
NULL	NULL	NULL	NULL	NULL	NULL



```
SELECT * FROM sourashtraclub.users;
```

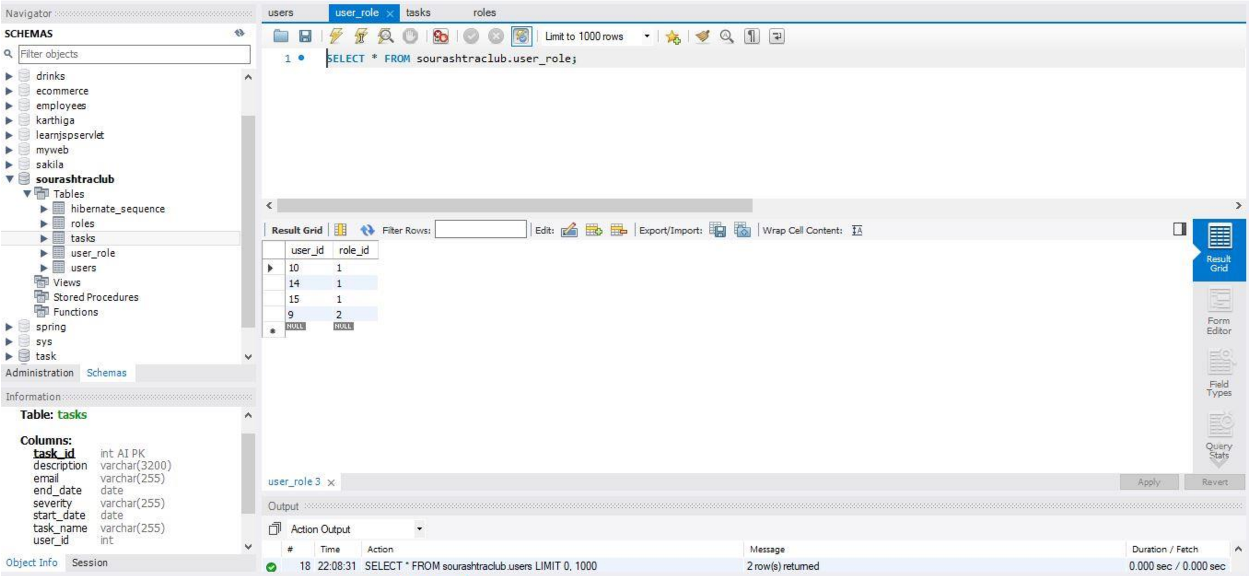
user_id	active	first_name	last_name	password	user_name
9	1	Karthiga	Ravichandran	$2a$10$3TrJrmrqeN.Jl0baCS6vLOMJgEjKBJAhu092tAYAjRxKK.4Rl7...	k
10	1	Karthiga	Ravichandran	$2a$10$UmIQZs4Lhoj6az2CGWsF0emFaFoqWj.6GQQmyPn.b5sLZ...	a
14	1	Ravi	Chan	$2a$10$2VZtzys7uRSRgSptX5MIDeHNaJEiZjBjXE6S/g1.jfGuBd.TvQ...	r
15	1	Ram	Kumar	$2a$10$uvq744M.K9Z5If6tbyvf1.xYqmXXkxRCZFgIrOzTtUFOZNi2V...	ram
NULL	NULL	NULL	NULL	NULL	NULL

Table: tasks

Columns:

task_id	int AI PK
description	varchar(3200)
email	varchar(255)
end_date	date
severity	varchar(255)
start_date	date
task_name	varchar(255)

#	Time	Action	Message	Duration / Fetch
18	22:08:31	SELECT * FROM sourashtraclub.users LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec



```
SELECT * FROM sourashtraclub.user_role;
```

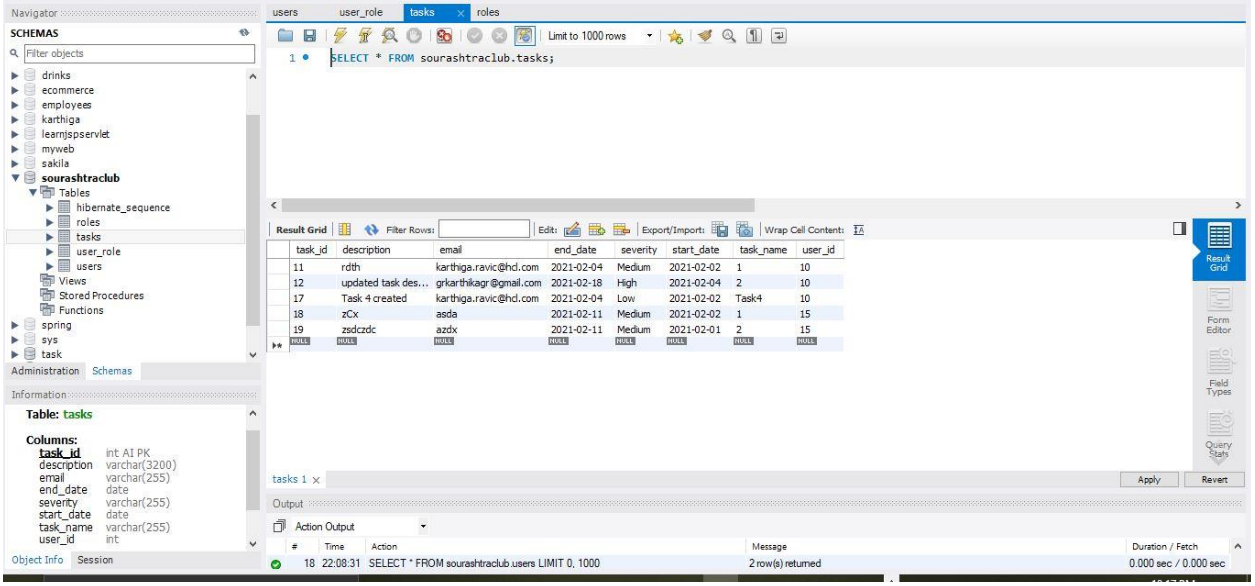
user_id	role_id
10	1
14	1
15	1
9	2
NULL	NULL

Table: tasks

Columns:

task_id	int AI PK
description	varchar(3200)
email	varchar(255)
end_date	date
severity	varchar(255)
start_date	date
task_name	varchar(255)
user_id	int

#	Time	Action	Message	Duration / Fetch
18	22:08:31	SELECT * FROM sourashtraclub.users LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec



```
SELECT * FROM sourashtraclub.tasks;
```

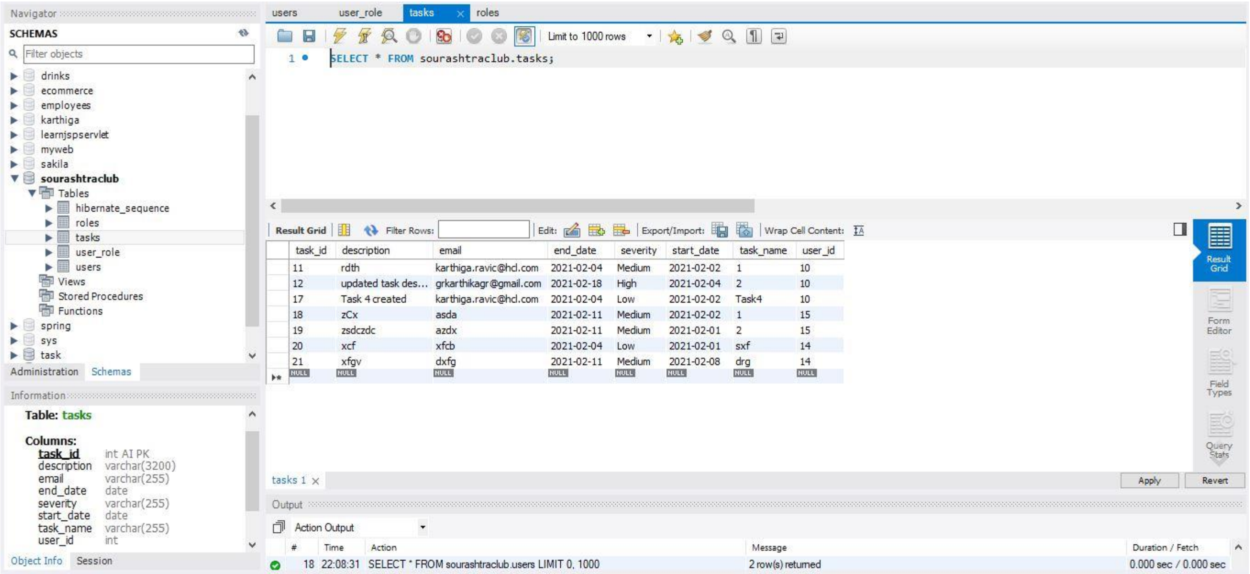
task_id	description	email	end_date	severity	start_date	task_name	user_id
11	rdth	karthiga.ravic@hcl.com	2021-02-04	Medium	2021-02-02	1	10
12	updated task des...	grkarthikagr@gmail.com	2021-02-18	High	2021-02-04	2	10
17	Task 4 created	karthiga.ravic@hcl.com	2021-02-04	Low	2021-02-02	Task4	10
18	zCx	asda	2021-02-11	Medium	2021-02-02	1	15
19	zsdczdc	azdx	2021-02-11	Medium	2021-02-01	2	15
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table: tasks

Columns:

task_id	int AI PK
description	varchar(3200)
email	varchar(255)
end_date	date
severity	varchar(255)
start_date	date
task_name	varchar(255)
user_id	int

#	Time	Action	Message	Duration / Fetch
18	22:08:31	SELECT * FROM sourashtraclub.users LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec



```
SELECT * FROM sourashtraclub.tasks;
```

task_id	description	email	end_date	severity	start_date	task_name	user_id
11	rdth	karthiga.ravic@hcl.com	2021-02-04	Medium	2021-02-02	1	10
12	updated task des...	grkarthikagr@gmail.com	2021-02-18	High	2021-02-04	2	10
17	Task 4 created	karthiga.ravic@hcl.com	2021-02-04	Low	2021-02-02	Task4	10
18	zCx	asda	2021-02-11	Medium	2021-02-02	1	15
19	zsdczdc	azdx	2021-02-11	Medium	2021-02-01	2	15
20	xcf	xfcb	2021-02-04	Low	2021-02-01	sxf	14
21	xfgv	dxfg	2021-02-11	Medium	2021-02-08	drg	14
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

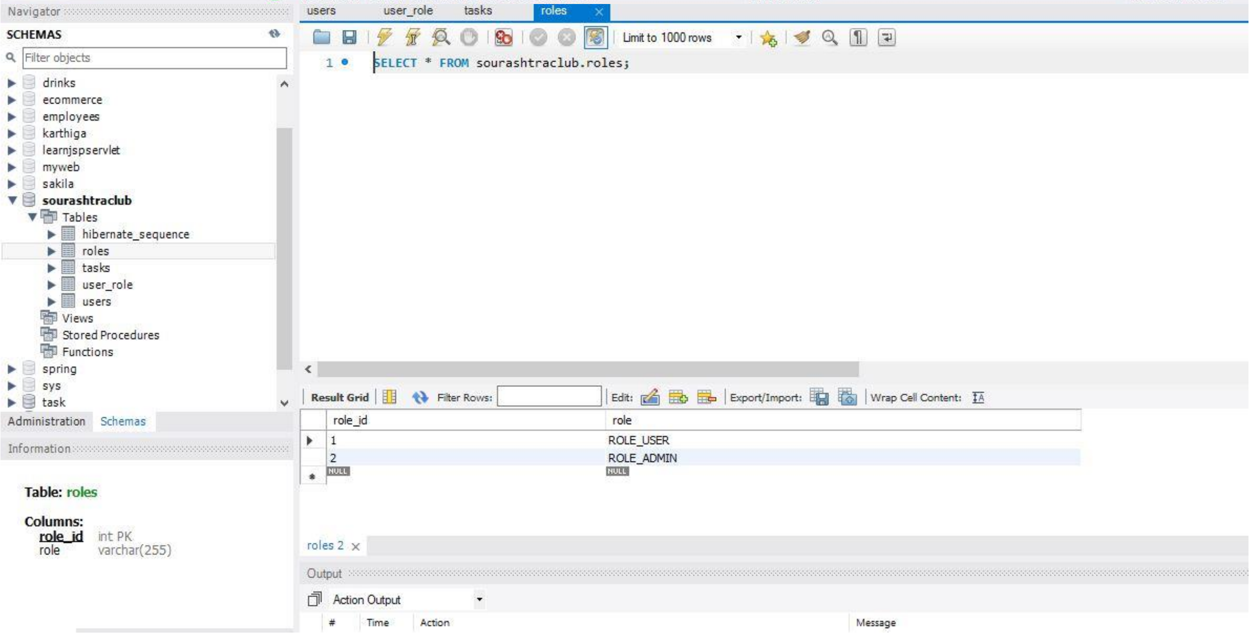
tasks 1

Table: tasks

Columns:

task_id	int AI PK
description	varchar(3200)
email	varchar(255)
end_date	date
severity	varchar(255)
start_date	date
task_name	varchar(255)
user_id	int

#	Time	Action	Message	Duration / Fetch
18	22:08:31	SELECT * FROM sourashtraclub.users LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec



```
SELECT * FROM sourashtraclub.roles;
```

role_id	role
1	ROLE_USER
2	ROLE_ADMIN
NULL	NULL

roles 2

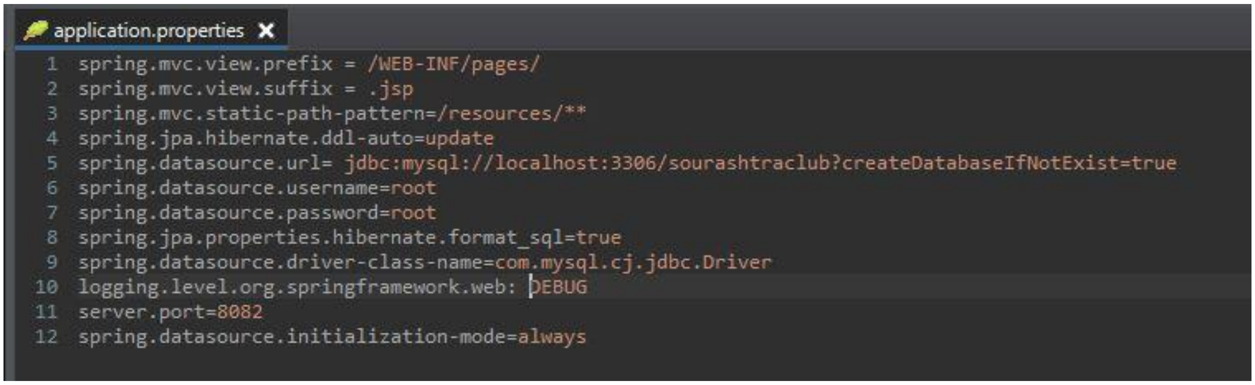
Table: roles

Columns:

role_id	int PK
role	varchar(255)

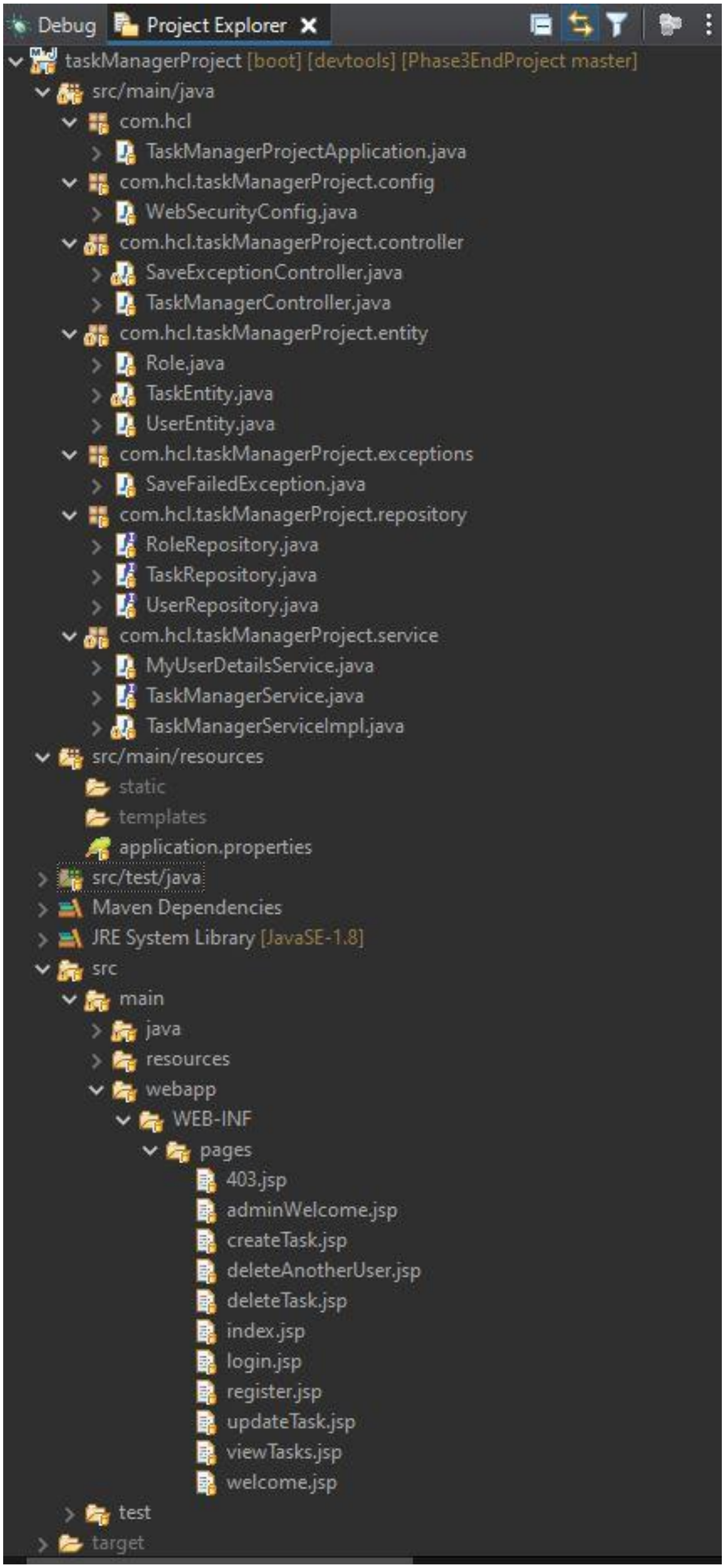
#	Time	Action	Message

Application Properties:



```
spring.mvc.view.prefix = /WEB-INF/pages/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url= jdbc:mysql://localhost:3306/sourashtraclub?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.properties.hibernate.format_sql=true
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
logging.level.org.springframework.web: DEBUG
server.port=8082
spring.datasource.initialization-mode=always
```

Folder Structure:



```
taskManagerProject [boot] [devtools] [Phase3EndProject master]
  src/main/java
    com.hcl
      TaskManagerProjectApplication.java
    com.hcl.taskManagerProject.config
      WebSecurityConfig.java
    com.hcl.taskManagerProject.controller
      SaveExceptionController.java
      TaskManagerController.java
    com.hcl.taskManagerProject.entity
      Role.java
      TaskEntity.java
      UserEntity.java
    com.hcl.taskManagerProject.exceptions
      SaveFailedException.java
    com.hcl.taskManagerProject.repository
      RoleRepository.java
      TaskRepository.java
      UserRepository.java
    com.hcl.taskManagerProject.service
      MyUserDetailsService.java
      TaskManagerService.java
      TaskManagerServiceImpl.java
  src/main/resources
    static
    templates
    application.properties
  src/test/java
  Maven Dependencies
  JRE System Library [JavaSE-1.8]
  src
    main
      java
      resources
      webapp
        WEB-INF
          pages
            403.jsp
            adminWelcome.jsp
            createTask.jsp
            deleteAnotherUser.jsp
            deleteTask.jsp
            index.jsp
            login.jsp
            register.jsp
            updateTask.jsp
            viewTasks.jsp
            welcome.jsp
    test
  target
```