Pass Task Lab 7

COS30041 Creating Secure and Scalable Software (Swinburne University of Technology)

Mustafa Mushtaq
101978603
Task 7.1P

Task 4

4.1
The C and D operations would allow the user to create or delete record based on their own will and could compromise the security of the company. A user might create different records and gain advantage of the company. Similarly, a user might indulge in a crime and can easily delete its record.

4.2
The information that can be viewed by user is restricted due to security purposes. Everything except the password can be viewed by the user. This is because the password is the key to accessing a data record and can then be easily manipulated. It is assumed that a user knows its own password before viewing its own details.

4.3
This is a good practice since it doesn't allow the managed bean to have access to the password when it doesn't even exist. This prevents from password being shown up later in the web pages.

4.4
4.4.1
The information that can updated is not like those in 4.2 and 4.3. For example, before the password could not be viewed by can still be updated by the user. Here it is assumed that the user does know its own password and wishes to update only which can be done by entering the new password.

4.4.2
Information such as the salary cannot be updated by the employee. This can be done simply not giving the option to update the salary. Only the admins will have the option to update the entire details of the user. The active field can not be changed as well, since a user might be able to gain access and change the field to active if he is not currently employed.

4.5
The actual update should occur in the business layer where the EmployeeFacade has access to the entity class that maps to the table in the database and hence any changes made are directly visible in the database. The information can be validated in the web tier and the update should occur in a business layer and data layer to push those changes.

4.6
The existing password should be sent to the client but not displayed. This is because this can be used later when the user wishes to change the password. If a user enters its old password, it can be matched with the existing password and can be used to verify.

4.7

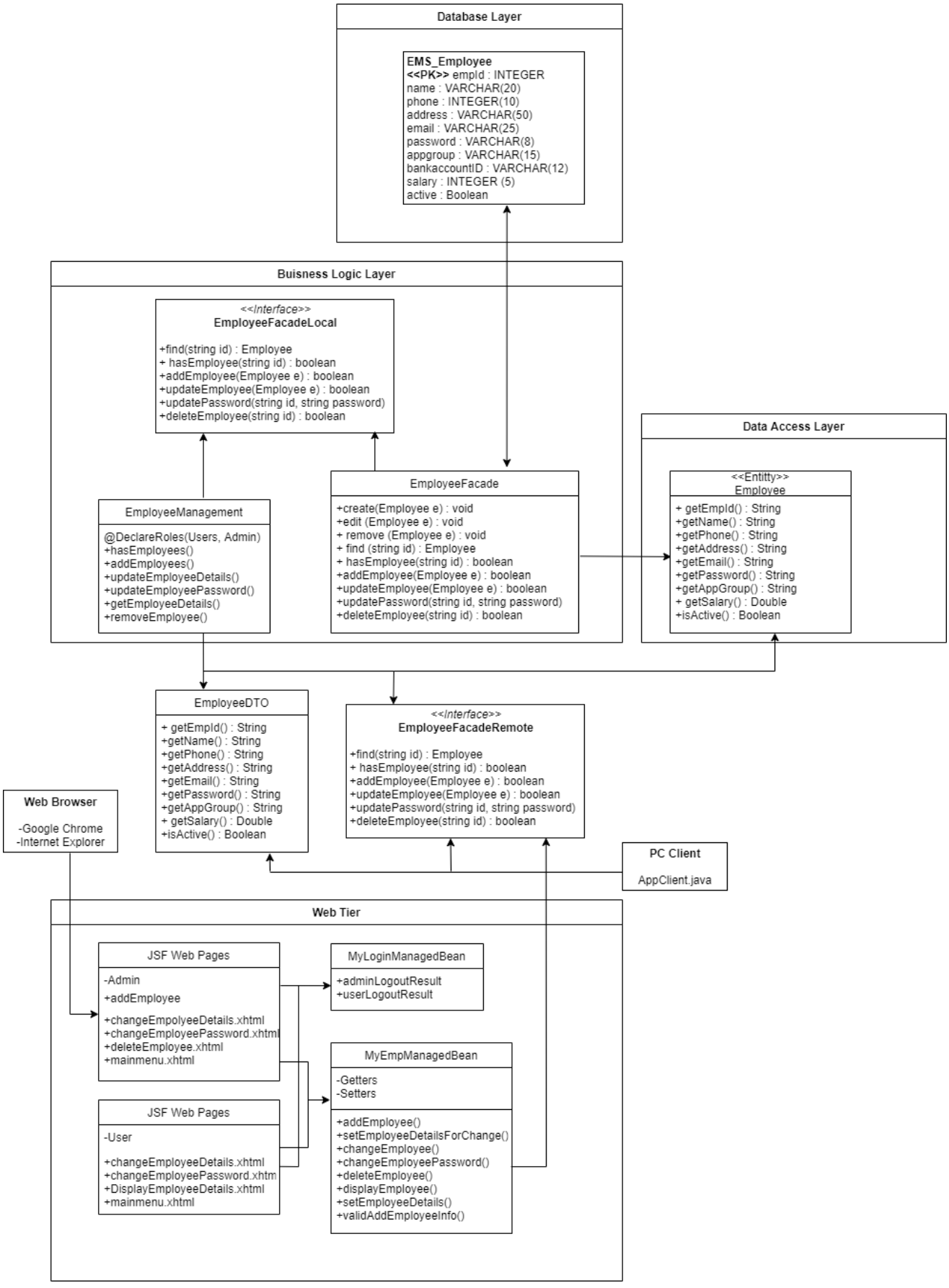
It is a good practice to keep the record for future reference in case the user wishes to get employed again. If the record is deleted, the company might never know if the user was ever a part of the company before. This also doesn't affect the records in the database since each record has a unique userid.

4.8

It can provide all features listed in the case study but might not be very efficient. For example, if there are a huge number of employees, each employee's details need to be entered into the glass fish server to get access later. If a user wishes to change the password, this is only updated in the database and not the glassfish server. This problem can be rectified to some extent by the JDBC realm where the user has access to database.

Task 5

a)



b)

Software Components

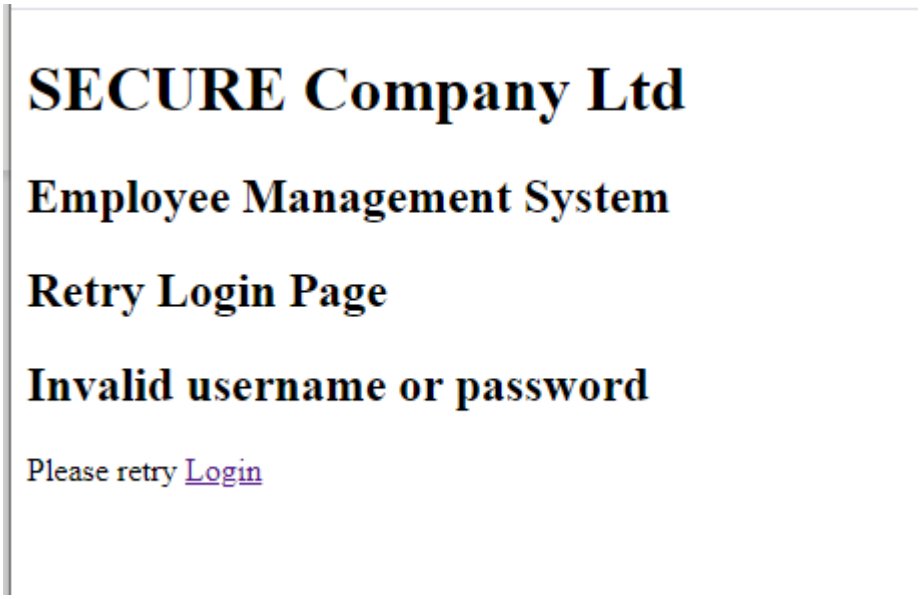
- Web Browser
  This provides the user interface for the user to access the web pages where the details can be viewed or modified
- JSF Pages
  These pages are responsible for the content that is displayed on the front end when the website is accessed. On the back end it exchanges data with the **MyEmpManagedBean** to display or modify the user details. In case of the pages that display the user details, the password has been omitted for the security of the user. It does however provide the option to update the password in which case both the user needs to enter the same password twice before it can successfully be updated. The jsf pages are divided into two groups, admin and user. A user can only access the JSF pages that relates to the user. It cannot access admin pages since they are protected by an admin id and password. Hence the options available to the users are also limited.
- MyEmpManagedBean
  This uses the EmployeeManagementRemote interface to implement methods that performs the CRUD operations as desired by the user. For example, when the user enters the new details, they are stored in the bean and thereby relative methods are called to modify the existing data with the new data.
- AppClient
  Here it doesn't have to deal with the web pages. There is no user interface, and everything is coded to deal with the modifications of the data. Test cases are setup and the results can be checked in the database.
- EmployeeDTO
  It contains the information related to an employee. It is just a data transfer object and plays no role in dealing directly with the database.
- EmployeeManagementRemote
  This is an interface that maps to the EmployeeFacade. All the methods in the façade are declared in the remote interface so that it can be accessed by the client remotely.
- EmployeeFacade
  It provides the methods that performs the CRUD operations on a record. The façade class deals directly with the database via the entity class.
- EmployeeManagement
  The role of this class is similar to that of the Façade. It has methods that provides the same functionality as that of the façade through a local interface that holds the declaration of all the methods found in façade. An added feature of this class is that it allows the roles to be declared. Only those groups can access the roles that are allowed for that specific group.

- Employee Entity
  Each instance of this object maps to a record in the database, which means that a row in the database table is an object of this entity class. Any changes made to this class can be directly seen in the database table. This can be manipulated via the entity manager.
- Database
  This contains the table that holds the information related to the user or admin.

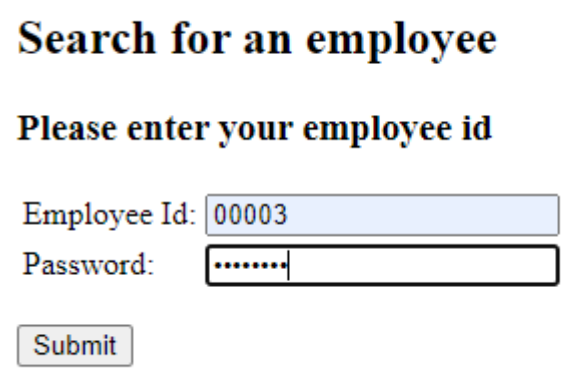
Task 6

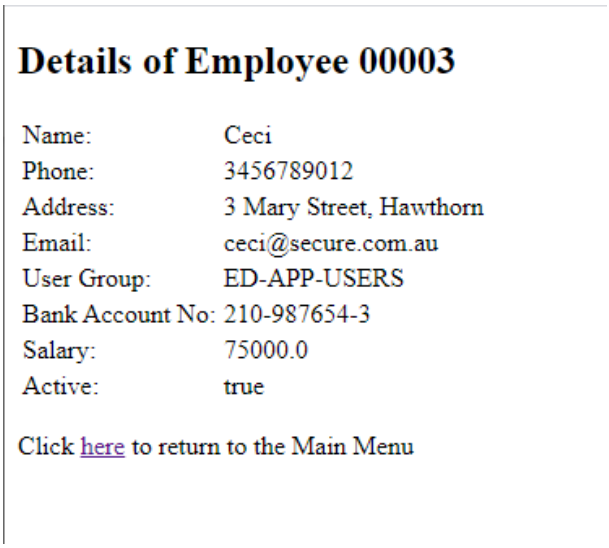
- Logging in via User Details

Wrong details will give an error.

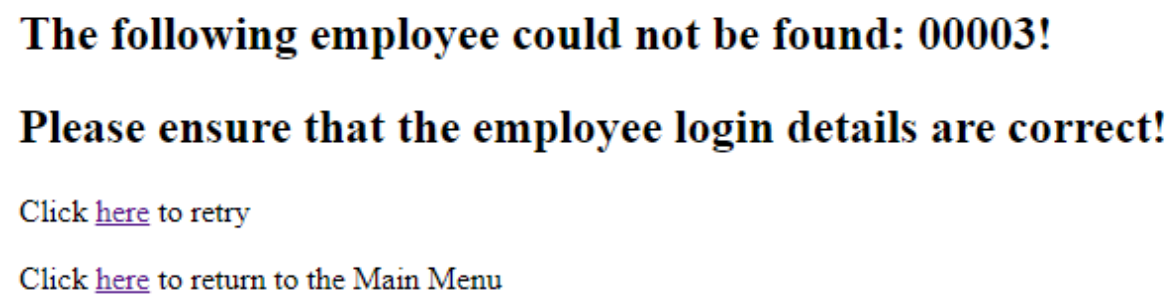


- Employees can view their own details except for the password.
  For this to work the user must enter its empId and password. If the password is not the same as the one that exists in the database, the user will not be able to access its details. Also, the user group must be “ED-APP-USERS” or the details wouldn’t be accessed.





If the details entered are not correct then it gives an error and the user must retry with proper login details.



Since the empId **00001** belongs to the admin group, it cannot be accessed via the users management system.

The following employee could not be found: 00001!

Please ensure that the employee login details are correct!

Click here to retry

Click here to return to the Main Menu

- Employees can change all details as well as password.

Change Employee Details Page

Please update the details of employee 00003

Employee Id:	00003
Name:	Brooklie
Phone:	3456789012
Address:	3 Mary Street, Hawthorn
Email:	ceci@secure.com.au
User Group:	ED-APP-USERS
Bank Account No	210-987654-3
Salary:	75000.0
Active:	☑

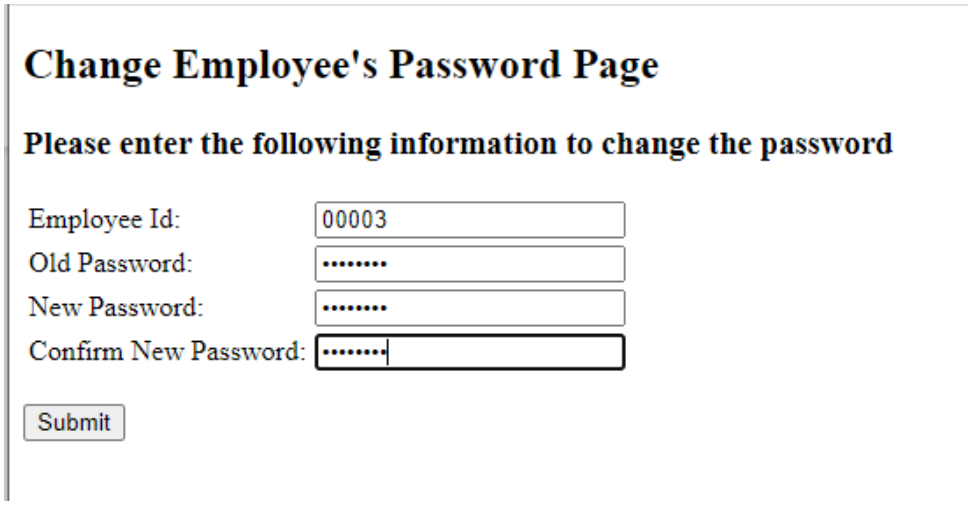
Submit

Upon submission the changes can be seen in the database.

#	EMPID	NAME
1	00001	Mustafa
2	00002	Bill
3	00003	Brooklie
4	00004	Dave
5	00009	Issac

To change the password, the user must also enter the old password as well as the new password. The old password must match the password that is set in the database before it can be updated.

If the password match is successful the password is updated.



If the details are incorrect the following error message is shown

- Code

The following methods are declared in the EmployeeManagement with only users role allowed for specific methods. Here it takes as an argument the empId and password. The empId is used to fetch the record and its password is then compared. Details are returned if the passwords match.

```
@Override
@RolesAllowed({"ED-APP-USERS"})
public boolean checkUserIdPwd(String empId, String pwd) {
    Employee e = employeeFacade.find(empId);

    if (e == null) {
        return false;
    }
    if (!(e.getAppGroup().equals("ED-APP-USERS"))) {
        return false;
    }
    return (e.getPassword().equals(pwd));
}

@Override
@RolesAllowed("ED-APP-USERS")
public EmployeeDTO getUserDetails(String empId, String pwd) {
    if (checkUserIdPwd(empId, pwd)) {
        return getUserDetailsPrivate(empId);
    }
    return null;
}

private EmployeeDTO getUserDetailsPrivate(String empId) {
    Employee e = employeeFacade.find(empId);

    if (e == null) {
        return null;
    } else {
        EmployeeDTO empDTO = new EmployeeDTO(e.getEmpid(), e.getName(),
                e.getPhone(), e.getAddress(), e.getEmail(), e.getPassword(),
                e.getAppGroup(), e.getBnkAccId(), e.getSalary(), e.isActive());
        return empDTO;
    }
}

@Override
@RolesAllowed("ED-APP-USERS")
public boolean updateUserPassword(String empId, String pwd, String nPwd) {
    if (checkUserIdPwd(empId, pwd)){
        return updateEmployeePassword(empId, nPwd);
    }
    return false;
}
```

In MyEmpManagedBean the following methods are used for the users.

```
public String setUserDetailsForChange() {
    // check empId is null
    if (isNull(empId) || conversation == null) {
        return "debug";
    }

    if (!employeeManagement.hasEmployee(empId)) {
        return "failure";
    }
    // note the startConversation of the conversation
    startConversation();

    // get employee details
    return setUserDetails();
}
```

First the password entered by the user is checked against the password in the database. Upon success the details are returned.

```
private String setUserDetails() {

    if (isNull(empId) || conversation == null) {
        return "debug";
    }

    EmployeeDTO e = employeeManagement.getUserDetails(empId, password);

    if (e == null) {
        // no such employee
        return "failure";
    } else {
        // found - set details for display
        this.empId = e.getEmpid();
        this.name = e.getName();
        this.phone = e.getPhone();
        this.address = e.getAddress();
        this.email = e.getEmail();
        this.password = e.getPassword();
        this.appGroup = e.getAppGroup();
        this.bnkAccId = e.getBnkAccId();
        this.salary = e.getSalary();
        this.active = e.isActive();
        return "success";
    }
}
```

```
public String changeUserPassword() {
    // check empId is null
    if (isNull(empId)) {
        return "debug";
    }

    // newPassword and confirmPassword are the same - checked by the validator during input to JSF form
    boolean result = employeeManagement.updateUserPassword(empId, password, newPassword);

    System.out.println("result = " + result);

    if (result) {
        return "success";
    } else {
        return "failure";
    }
}
```

To change the user password, the old password is also considered. The old password is checked against the password in the database and upon success, the new password is updated to be the current password of the user and stored in the database.

- Test Case