

7.1P


Task 1.

Lab_07a was complete and the following users were added to glassfish.

Select	User ID	Group List:
<input type="checkbox"/>	ed-none	EN-NONE
<input type="checkbox"/>	ed-guest	ED-APP-GUEST
<input type="checkbox"/>	ed-admin1	ED-APP-ADMIN
<input type="checkbox"/>	ed-admin2	ED-APP-ADMIN
<input type="checkbox"/>	ed-user1	ED-APP-USERS
<input type="checkbox"/>	ed-user2	ED-APP-USERS

Task 2.

Lab_07b was complete and the security measures were added.

 Login for user: ×

Enter Username:

Enter Password:

```
Adding an employee to the database: Edmonds
The operation is successful.
Want to remove the employee record just added? (Y/N)
Y
Removing an employee from the database: Edmonds
The remove operation is successful.
run:
BUILD SUCCESSFUL (total time: 37 seconds)
```

SECURE Company Ltd

Employee Management System

Login Page

Username

Password

SECURE Company Ltd

Employee Management System

Main Menu

1. [Add a new employee](#)
2. [Change an employee's details](#)
3. [Change an employee's password](#)
4. [Delete an employee](#)
5. [Display employee's details](#)

Click

Task 4.

4.1

Employees should not be able to create or delete either their own or other user accounts as this falls outside of the role of the regular employee. They should not have the authority to add or remove employees from the company, this would fall to an administrator presumably someone high up in the company.

4.2

When employees are review their data, they should be able to see all fields except: EMPID; APPGROUP; and ACTIVE. These fields are useful only to the database and the business logic. They pose no great relevance to an employee and are only needed by the system, so the employee does not need to see them in their reviewed data.

4.3

The password is probably excluded being sent to the employee for security reasons. If the user has left their client logged in and another user accesses the client, if the password is not included in the review data it is safe from being recorded by the third party. Not showing user's their passwords is a pretty standard practice across most systems with user accounts for this reason and I think that it is a good practice for this case as well.

4.4

4.4.1

The fields that should be able to be updated by the employee are: NAME; PHONE; ADDRESS: EMAIL; PASSWORD; and BANKACCOUNTID. These are all personal details that may change for a person and should be able to be updated by them when they need to.

4.4.2

The fields that shouldn't be updated by the employee are: EMPID; APPGROUP: SALARY; and ACTIVE. The salary field should only be set by an administrator as employees should not be able to change their salary to whatever they want in the database. The other three fields are all for the system and business logic so should also not be able to be updated by the employee.

4.5

The actual change of the employee's information occurs in the data access layer of the application. This is a good and safe practice as the data must go through at least the business logic layer to get here where there should be various checks on the data and how it was obtained before it gets to the data access layer and actually updated in the database.

4.6

For the user to update their password, their current password should probably be sent to the client but not displayed. This would allow a check to be performed where the user has to enter their current password before they can update their password without showing the original password to a third party that may have vision of the screen. The password could also be not sent at all and a separate verification method like a code in an email could be used, but that is probably not necessary here.

4.7

In my eyes the 'active' checkbox to signal current and past employees is not a great system. It works and you could still filter by it when searching the database or creating views, but a better solution would probably be to just have separate tables for both current and past employees with foreign keys leading back to this table for the rest of the employee data. Then the 'active' check box could be removed and you would have cleaner table lookups for current or past employees. This would also allow extra data to be stored for past employees such as the reason for them leaving the company, how long they were with the company and when they left.

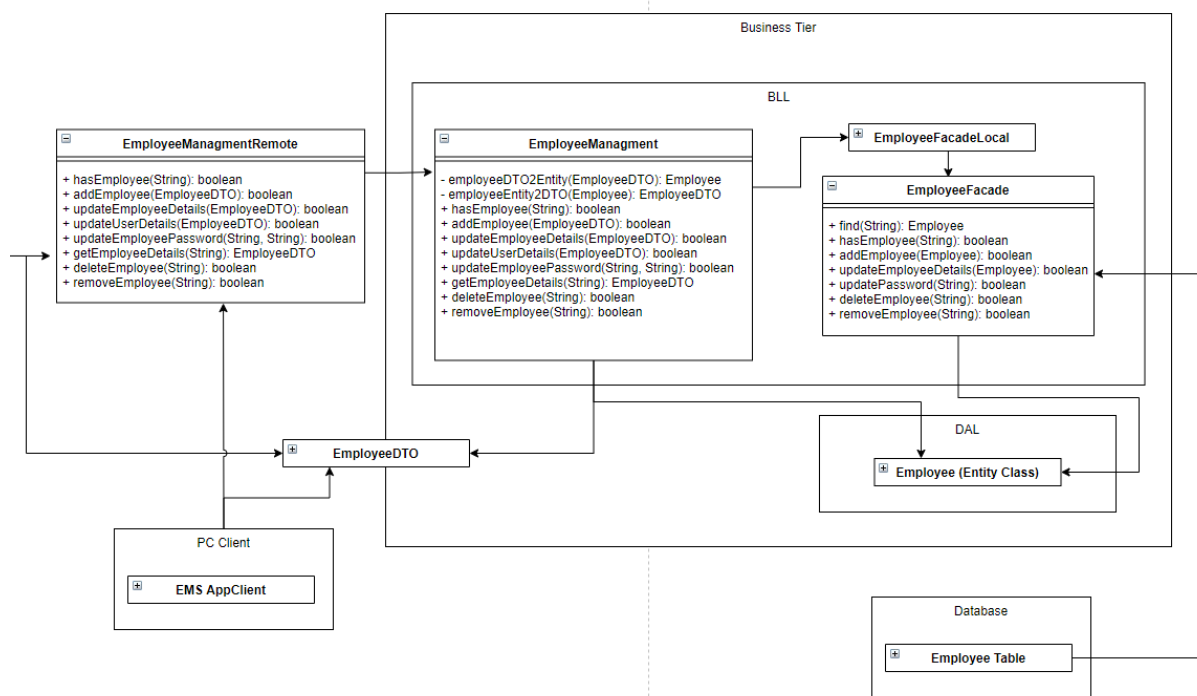
4.8

All of the features required by the case study to do with the administrator group and its privileges are already functioning in the current application. The user group does exist but does not have any restrictions or the features to review or update their details. So these will need to be added to the application.

Task 5.
(Diagram first half)



(Diagram second half)



MyEMSManagedBean (Java Bean)

myuserManagedBean is the bean that handles tasks from the web pages and uses myuserFacadeRemote to interact with the database. This includes getting data to display on the web pages from the database as well as sending data add to or update the database. This bean also handles some validation of fields on the web pages.

MyLoginManagedBean (Java Bean)

MyLoginManagedBean is a simple bean that handles terminating the current session the user is in and logging the user out of the currently logged in account.

EmployeeManagment (Stateless Session Bean)

EmployeeManagment is the bean that is responsible for performing business logic (this includes CRUD operations) with the DTO, transforming it into the DAO for the EmployeeFacade to act on the database with any changes. The main functions of this class are to allow the addition and removal of employees, updating of employee details and the retrieval of employee details. There are separate versions of some of these functions specifically for the admin and user roles to make the application more secure and only allow the correct persons access to the functions they should have.

EmployeeManagementRemote (Remote Interface for Stateless Session Bean)

EmployeeManagementRemote is the remote interface that describes what the EmployeeManagment bean is able to do. Other beans or classes can use the interface to perform tasks with the related bean.

EmployeeFacade (Stateless Session Bean)

EmployeeFacade is the stateless session bean in charge of performing business logic with the DTO (this involves CRUD operations) and changing the data into the DAO for use with the database.

EmployeeDTO (DTO)

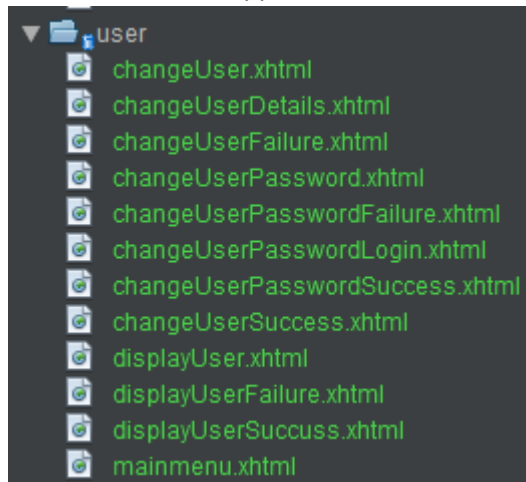
EmployeeDTO is the data transfer object which is created whenever employee data needs to be stored in an object and used or manipulated. Does not directly manipulate the database, it just holds the data that is passed around by beans.

Employee (DAO / Entity Class)

Employee is the data access object in charge of taking data and making queries to change data in the database as well as making queries to retrieve data.

Task 6.

To facilitate the requirements of allowing users to update and view their details, the following pages were added to the application:



These were mostly all based off the currently existing admin pages for similar features. A new type of page was required though for checking a user's login information not for the Glassfish server, but for the database entries. One of these pages was required for each of the operations in the users' main menu, but as they are almost identical to each other here is just one of them:

```
<h:head>
  <title>Change your Details</title>
</h:head>
<h:body>

  <h2>Change your details</h2>

  <h3>Please enter your id and password</h3>

  <h:form>
    <table>
      <tbody>
        <tr>
          <td>
            <h:outputLabel value="Employee Id:"/>
          </td>
          <td>
            <h:inputText title="EmpId" id="empid"
              required="true"
              requiredMessage="Employee id must not be blank"
              maxLength="5"
              value="#{myEmpManagedBean.empId}"
              <f:validateLength minimum="5" maximum="5"/>
            </h:inputText>
          </td>
        </tr>
        <tr>
          <td>
            <h:outputLabel value="Employee Password:"/>
          </td>
          <td>
            <h:inputSecret title="Password" id="emppassword"
              required="true"
              requiredMessage="Password must not be blank"
              value="#{myEmpManagedBean.password}"
            </h:inputSecret>
          </td>
          <td>
            <h:message for="empid" style="color:red"/>
          </td>
        </tr>
      </tbody>
    </table>

    <p></p>

    <h:commandButton id="submit" value="Submit" action="#{myEmpManagedBean.setUsersDetailsForChange()}" />

  </h:form>
</h:body>
```

This page required a new function for checking the login details and if correct then allowing the setting of the user's details for review or change:
setUserDetailsForChange() in MyEMPManagedBean.java

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

    if (!employeeManagement.hasEmployee(empId)) {
        return "failure";
    }

    EmployeeDTO empDTO = employeeManagement.getEmployeeDetails(empId);

    if (password.equals(empDTO.getPassword())) {
        // note the startConversation of the conversation
        startConversation();

        // get employee details
        return setEmployeeDetails();
    }

    else return "failure";
}
```

setEmployeeDetails() in MyEmpManagedBean.java

```
private String setEmployeeDetails() {

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

    EmployeeDTO e = employeeManagement.getEmployeeDetails(empId);

    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";
    }
}
```

The ChangeUserDetails page was very similar to the admin equivalent just with the fields that were discussed earlier to not be relevant removed. (Input texts were excluded for sake of brevity, bean data is just being set there the same as in the admin version of this page.)

```
<h:head>
  <title>Change your Details</title>
</h:head>
<h:body>

  <h2>Change User Details Page</h2>

  <h3>Please update your details
</h3>

  <p></p>

  <h:form>
    <table>
      <tbody>
        <tr>
          <td>
            <h:outputLabel value="Name:"/>
          </td>
          <td ...7 lines />
        </tr>
        <tr>
          <td>
            <h:message for="name" style="color:red"/>
          </td>
        </tr>
        <tr>
          <td>
            <h:outputLabel value="Phone:"/>
          </td>
          <td ...5 lines />
        </tr>
        <tr>
          <td>
            <h:outputLabel value="Address:"/>
          </td>
          <td>
            <h:inputText ...3 lines />
          </td>
        </tr>
        <tr>
          <td>
            <h:outputLabel value="Email:"/>
          </td>
          <td>
            <h:inputText ...3 lines />
          </td>
        </tr>
        <tr>
          <td>
            <h:outputLabel value="Bank Account No"/>
          </td>
          <td>
            <h:inputText ...3 lines />
          </td>
        </tr>
      </tbody>
    </table>

    <p></p>

    <h:commandButton id="submit" value="Submit"
      action="#{myEmpManagedBean.changeUser()}" />

  </h:form>
</h:body>
```


changeUser() in MyEmpManagedBean.java

```
public String changeUser() {
    EmployeeDTO empDTO = new EmployeeDTO(empId, name, phone,
        address, email, password, appGroup, bnkAccId, salary, active);
    boolean result = employeeManagement.updateUserDetails(empDTO);

    // note the endConversation of the conversation
    endConversation();

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

updateUserDetails() in EmployeeManagement.java

```
@Override
@RolesAllowed({"ED-APP-USERS"})
public boolean updateUserDetails(EmployeeDTO empDTO) {

    if (empDTO.getEmpid() == null) {
        return false;
    }
    // check employee exist?
    if (!hasEmployee(empDTO.getEmpid())) {
        return false;
    }

    // employee exist, update details
    // convert to entity class
    Employee employee = this.employeeDTO2EntityPartial(empDTO);
    // update details
    return employeeFacade.updateEmployeeDetails(employee);
}
```

Again, for changing the user's password, the page is very similar to the admin version.

```
<h:head>
  <title>Change Employee's Password</title>
</h:head>
<h:body>

  <h2>Change Users Password Page</h2>

  <h3>Please enter the following information to change the password</h3>

  <p></p>

  <h:form>
    <table>
      <tbody>
        <tr>
          <td>
            <h:outputLabel value="New Password:"/>
          </td>
          <td>
            <h:inputSecret id="newpwd" binding="#{newpwd}"
              title="NewPassword"
              maxlength="8"
              required="true"
              requiredMessage="New Password must not be blank"
              validator="#{myEmpManagedBean.validateNewPasswordPair}"
              value="#{myEmpManagedBean.newPassword}">
              <f:validateLength minimum="8" maximum="8"/>
              <f:attribute name="cnfpwd" value="#{cnfpwd}"/>
            </h:inputSecret>
          </td>
        </tr>
        <tr>
          <td>
            <h:message for="newpwd" style="color:red"/>
          </td>
        </tr>
        <tr>
          <td>
            <h:outputLabel value="Confirm New Password:"/>
          </td>
          <td>
            <h:inputSecret id="cnfpwd" binding="#{cnfpwd}"
              title="ConfirmPassword"
              maxlength="8"
              required="true"
              requiredMessage="Confirm New Password must not be blank"
              value="#{myEmpManagedBean.confirmPassword}">
              <f:validateLength minimum="8" maximum="8"/>
            </h:inputSecret>
          </td>
        </tr>
        <tr>
          <td>
            <h:message for="cnfpwd" style="color:red"/>
          </td>
        </tr>
      </tbody>
    </table>

    <p></p>

    <h:commandButton id="submit" value="Submit"
      action="#{myEmpManagedBean.changeUserPassword()}" />

  </h:form>
</h:body>
```

changeUserPassword() in MyEmpManagedBean.java

```
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.updateEmployeePassword(empId, newPassword);  
  
    System.out.println("result = " + result);  
  
    if (result) {  
        return "success";  
    } else {  
        return "failure";  
    }  
}
```

updateEmployeePassword() in EmployeeManagement.java

```
@Override  
@RolesAllowed({"ED-APP-ADMIN", "ED-APP-USERS"})  
public boolean updateEmployeePassword(String empId, String newPassword) {  
    return employeeFacade.updatePassword(empId, newPassword);  
}
```

Some extra navigation rules were required to link up the new pages and make sure successes and failures were handled properly.

```
<navigation-rule>
  <description>Change user</description>
  <from-view-id>/user/changeUser.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/user/changeUserDetails.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>failure</from-outcome>
    <to-view-id>/user/changeUserFailure.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
<navigation-rule>
  <description>Change user details</description>
  <from-view-id>/user/changeUserDetails.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/user/changeUserSuccess.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>failure</from-outcome>
    <to-view-id>/user/changeUserFailure.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
<navigation-rule>
  <from-view-id>/user/changeUserPasswordLogin.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/user/changeUserPassword.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>failure</from-outcome>
    <to-view-id>/user/changeUserPasswordFailure.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
<navigation-rule>
  <from-view-id>/user/changeUserPassword.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/user/changeUserPasswordSuccess.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>failure</from-outcome>
    <to-view-id>/user/changeUserPasswordFailure.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
<navigation-rule>
  <from-view-id>/user/displayUser.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>failure</from-outcome>
    <to-view-id>/user/displayUserFailure.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/user/displayUserSuccuss.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
<navigation-rule>
  <description>User's Main Menu</description>
  <from-view-id>user/mainmenu.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>logout</from-outcome>
    <to-view-id>logout.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

The security group for the user's group was also set up to make sure the correct login needed to be used to access the user management pages.

The screenshot displays the Oracle AEM Security Configuration console. The 'Security Roles' section shows a table with two roles: 'ED-APP-ADMIN' and 'ED-APP-USERS'. The 'ED-APP-USERS' role is associated with the description 'EMS Administrators'. Below the table are buttons for 'Add...', 'Edit...', and 'Remove'. The 'Security Constraints' section is expanded, showing the 'EMS-UsersOnly' constraint. It includes a 'Display Name' field with the value 'EMS-UsersOnly', a 'Web Resource Collection' table with one entry 'UsersOnly' having a URL pattern of '/faces/user/*', and checkboxes for 'Enable Authentication Constraint' (checked) and 'Enable User Data Constraint' (unchecked). The 'Role Name(s)' field is set to 'ED-APP-USERS'. The 'Transport Guarantee' is set to 'NONE'.

Role Name	Description
ED-APP-ADMIN	
ED-APP-USERS	EMS Administrators

Buttons: Add..., Edit..., Remove

Security Constraints

EMS-UsersOnly

Display Name: EMS-UsersOnly

Web Resource Collection:

Name	URL Pattern	HTTP Method
UsersOnly	/faces/user/*	

Buttons: Add..., Edit..., Remove

☒ Enable Authentication Constraint

Description:

Role Name(s): ED-APP-USERS

☐ Enable User Data Constraint

Description:

Transport Guarantee: NONE

Task 7.

Employee login:

SECURE Company Ltd

Employee Management System

Login Page

Username

Password

Success:

SECURE Company Ltd

Employee Management System

Main Menu

1. [Change your details](#)
2. [Change your password](#)
3. [Display your details](#)

Click

Failure:

SECURE Company Ltd

Employee Management System

Retry Login Page

Invalid username or password

Please retry [Login](#)

Change user details:

Login:

Change your details

Please enter your id and password

Employee Id:

Employee Password:

Update details:

Change User Details Page

Please update your details

Name:	<input type="text" value="Bob Testing"/>
Phone:	<input type="text" value="0987654321"/>
Address:	<input type="text" value="1 Place Street, Croydon"/>
Email:	<input type="text" value="bob@secure.com.au"/>
Bank Account No	<input type="text" value="098-765432-2"/>

(previous db entry)

#	EMPID	NAME	PHONE	ADDRESS	EMAIL
1	00001	Adam	1234567890	1 John Street, Hawthorn	adam@secure.com.au
PASSWORD	APPGROUP	BANKACCOUNTID	SALARY	ACTIVE	
11111111	ED-APP-ADMIN	098-765432-1	50000.00	<input checked="" type="checkbox"/>	

Success:

Your details have been updated in the system.

Click [here](#) to return to the Main Menu

(new db entry)

#	EMPID	NAME	PHONE	ADDRESS	EMAIL
1	00001	Bob Testing	0987654321	1 Place Street, Croydon	bob@secure.com.au
PASSWORD	APPGROUP	BANKACCOUNTID	SALARY	ACTIVE	
11111111	ED-APP-USERS	098-765432-2	50000.00	<input checked="" type="checkbox"/>	

Change Password:

Login:

Change your details

Please enter your id and password

Employee Id:	<input type="text" value="00001"/>
Employee Password:	<input type="password" value="....."/>

Change password:

Change Users Password Page

Please enter the following information to change the password

New Password:

Confirm New Password:

Success:

The password of the employee whose id is 00001 has been updated in the system.

Click [here](#) to return to the Main Menu

(new db entry)

#	EMPID	NAME	PHONE	ADDRESS	EMAIL
1	00001	Bob Testing	0987654321	1 Place Street, Croydon	bob@secure.com.au
PASSWORD	APPGROUP	BANKACCOUNTID	SALARY	ACTIVE	
12345678	ED-APP-USERS	098-765432-2	50000.00	<input checked="" type="checkbox"/>	

View details:

Login:

View your details

Please enter your id and password

Employee Id:

Employee Password:

Success:

Details of Employee 00001

Name: Bob Testing
Phone: 0987654321
Address: 1 Place Street, Croydon
Email: bob@secure.com.au
Bank Account No: 098-765432-2
Salary: 50000.0

Click [here](#) to return to the Main Menu

Code can be found for further review at:

https://github.com/CyrusEdgren/Secure_Scalable_Software/tree/master/7.1P