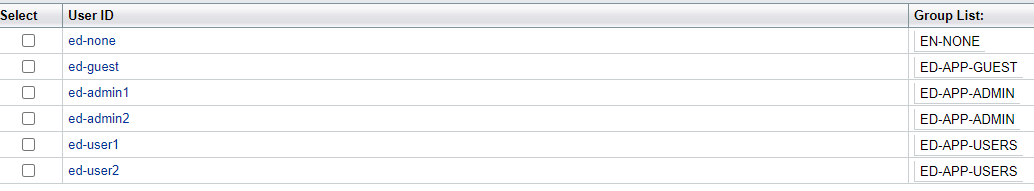
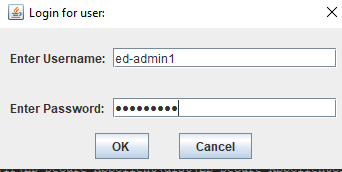
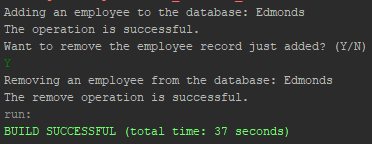
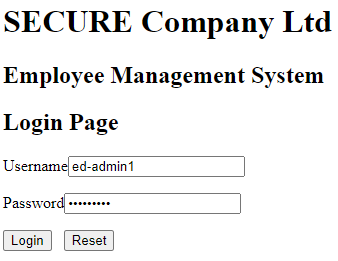
# 7.1P

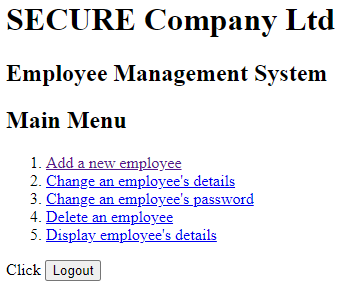
## Task 1.

Lab\_07a was complete and the following users were added to glassfish.  


## Task 2.

Lab\_07b was complete and the security measures were added.  
  






## 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.

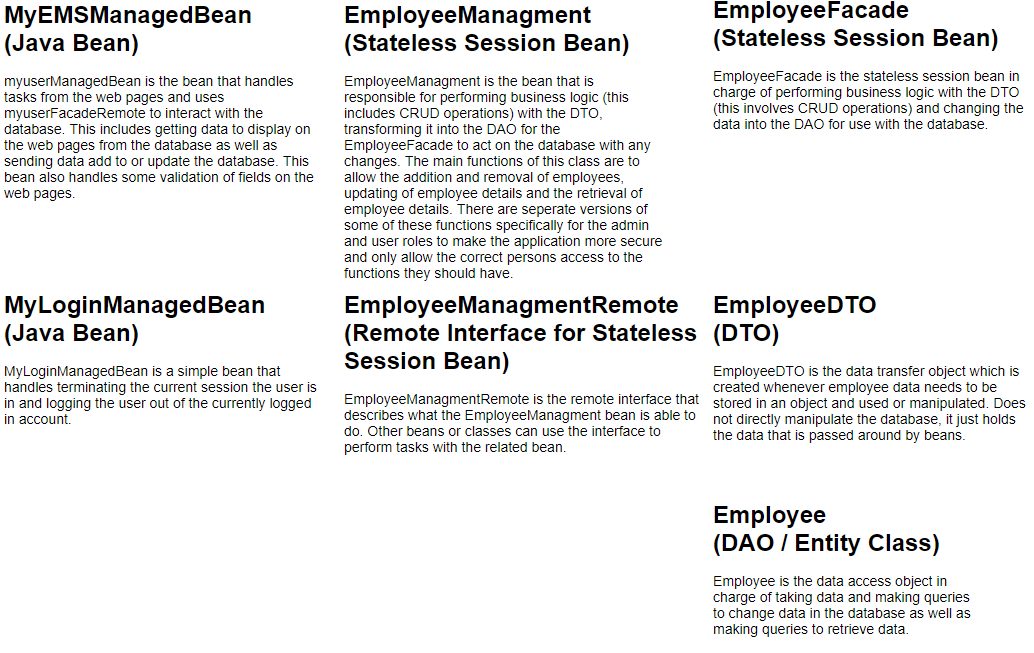
4.7  
In my eyes the ‘active’ checkbox to signal current and past employees is not a great system. It works and you could still filter by it when searching the database or creating views, but a better solution would probably be to just have separate tables for both current and past employees with foreign 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.

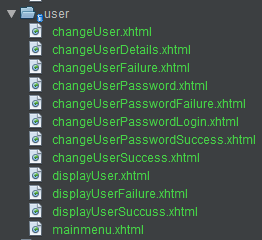
## Task 5. (Diagram first half)

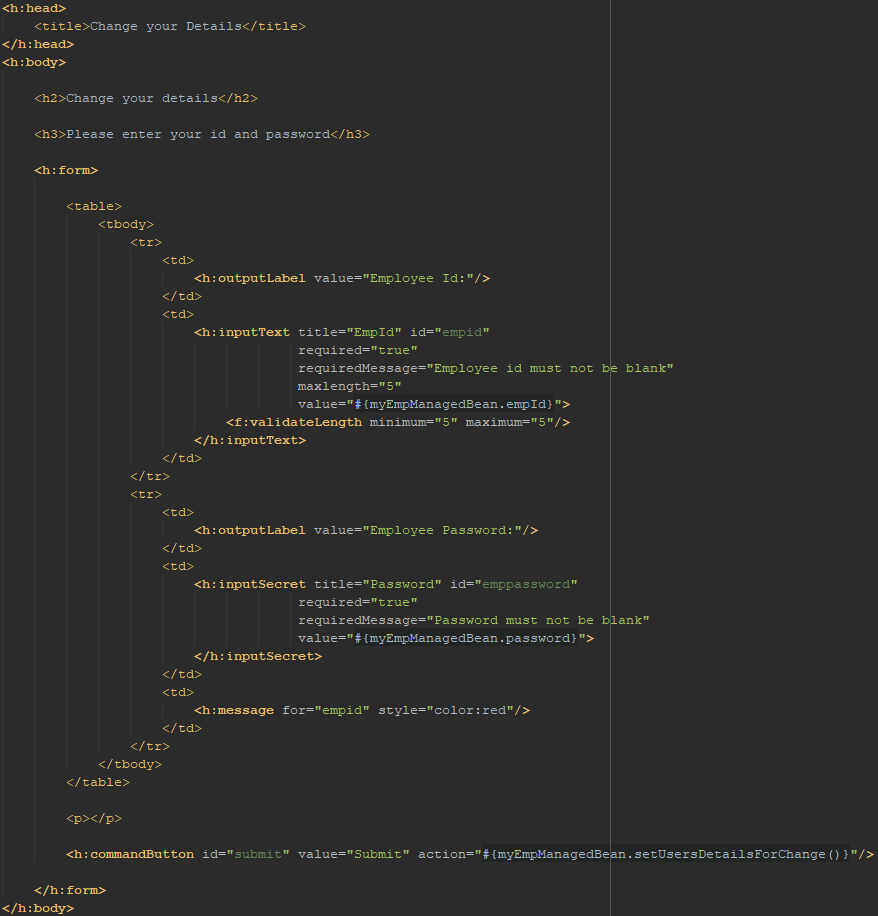
## 

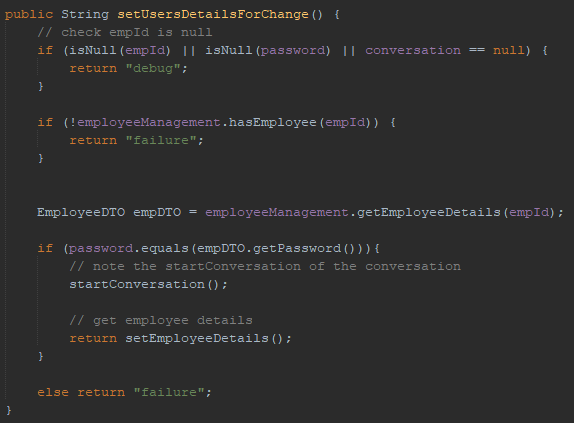
## (Diagram second half)



## Task 6.

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

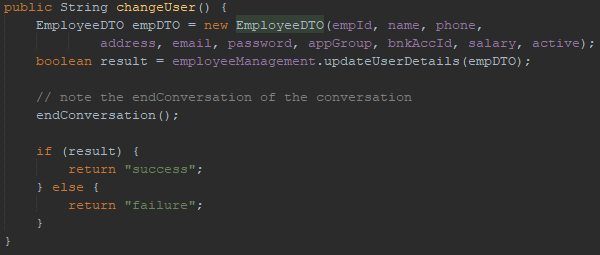
These were mostly all based off the currently existing admin pages for similar features.   
A new type of page was required though for checking a user’s login information not for the Glassfish server, but for the database entries. One of these pages was required for each of the operations in the users’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


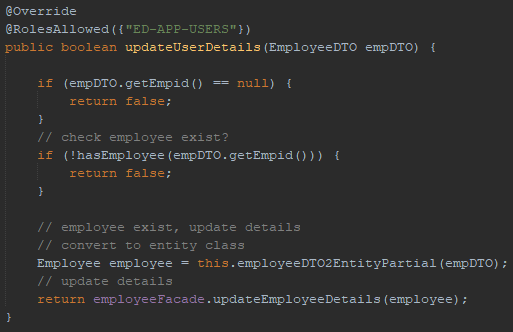
This page required a new function for checking the login details and if correct then allowing the setting of the user’s details for review or change:  
setUserDetailsForChange() in MyEMPManagedBean.java  


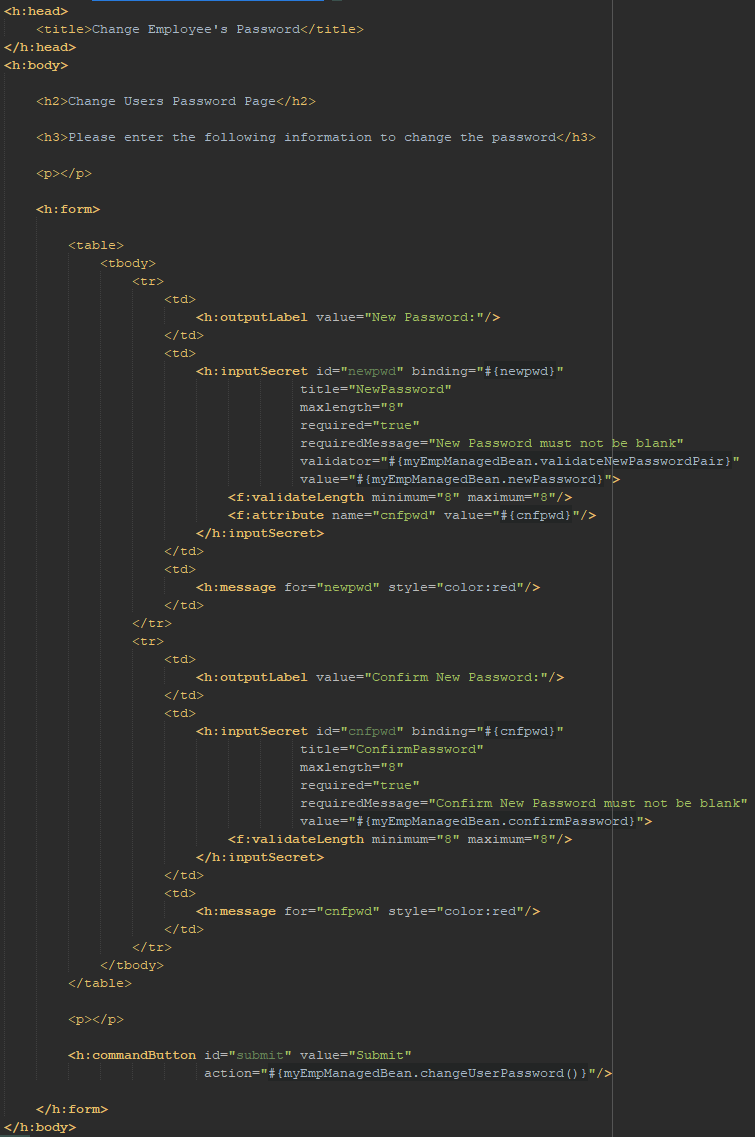
setEmployeeDetails() in MyEmpManagedBean.java

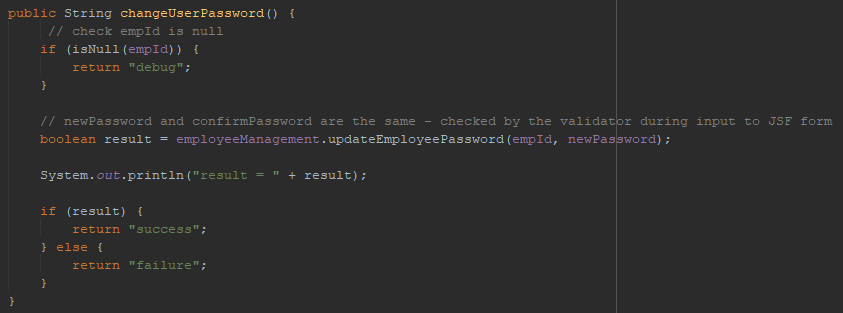


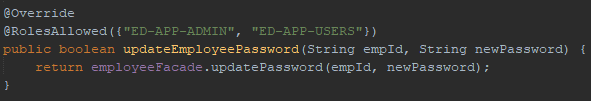
The ChangeUserDetails page was very similar to the admin equivalent just with the fields that were discussed earlier to not be relevent removed. (Input texts were excluded for sake of brevity, bean data is just being set there the same as in the admin version of this page.)  

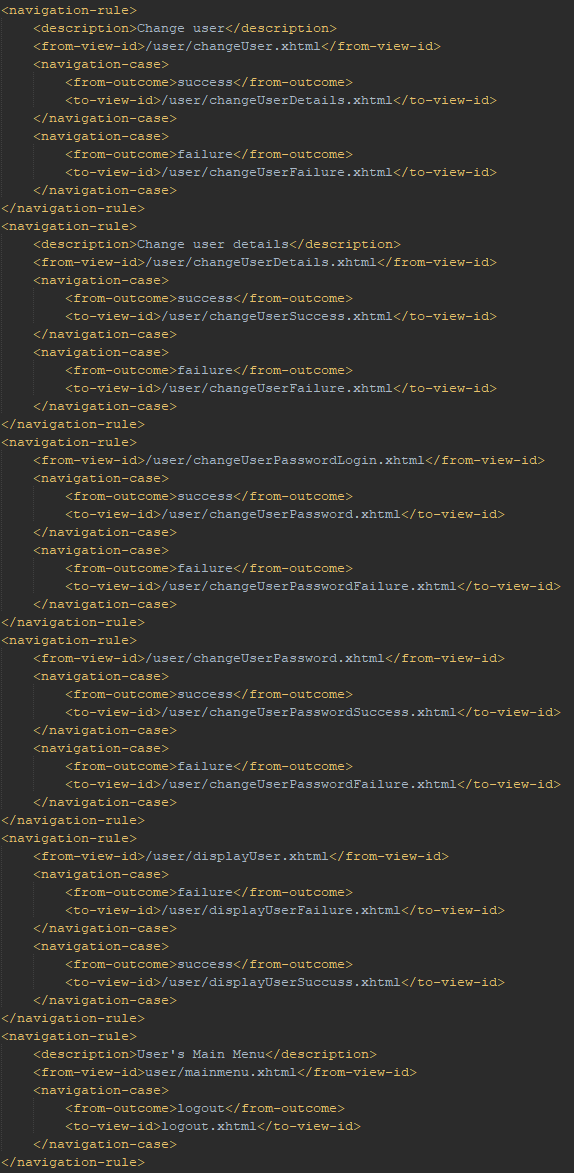

changeUser() in MyEmpManagedBean.java  


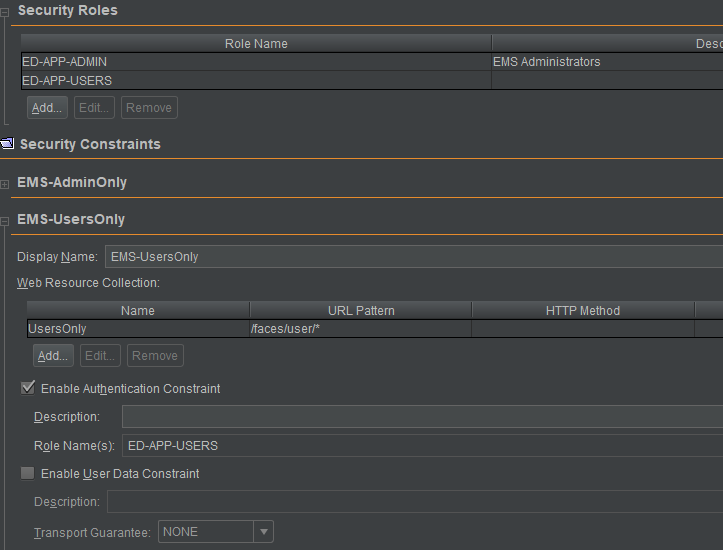
updateUserDetails() in EmployeeManagement.java  


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


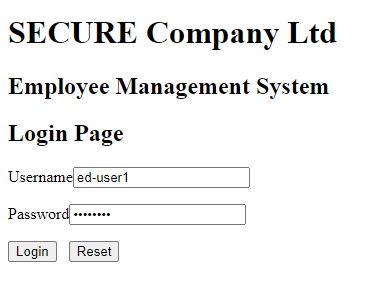
changeUserPassword() in MyEmpManagedBean.java  


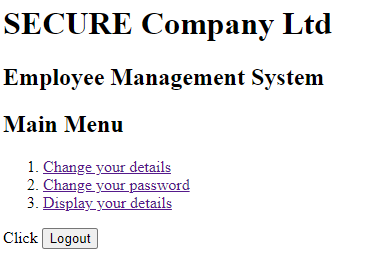
updateEmployeePassword() in EmployeeManagement.java  


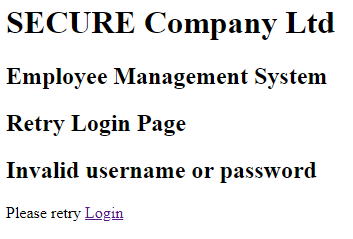
Some extra navigation rules were required to link up the new pages and make sure successes and failures were handled properly.  


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.  


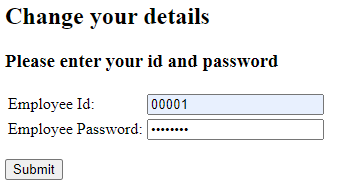
## Task 7.

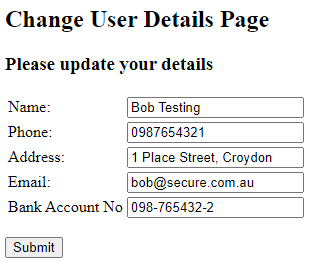
Employee login:  


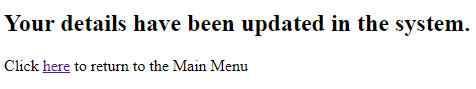
Success:  


Failure:  


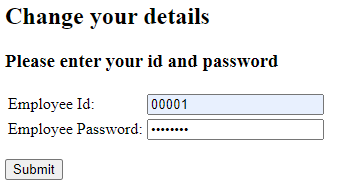
### Change user details:

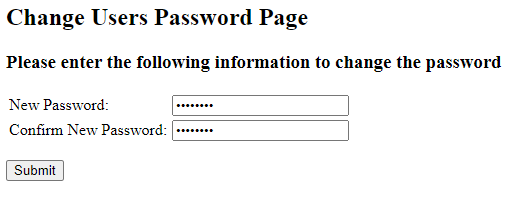
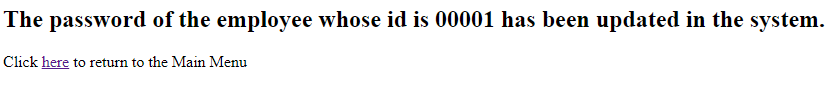
Login:  


Update details:  
  
(previous db entry)  
  


Success:  
  
(new db entry)  
  

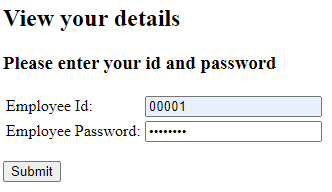

### Change Password:

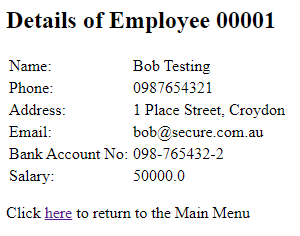
Login:  


Change password:  
  
Success:  


(new db entry)  
  


### View details:

Login:  


Success:  


Code can be found for further review at:   
<https://github.com/CyrusEdgren/Secure_Scalable_Software/tree/master/7.1P>