# Change request log

# Team

Name: Team 4

Members: Gerald Tucker (Coding) | Chikadibia Ihejimba (Documentation)

# Change Request

FEMR-158: don't require the user editing a user to fill out the "Change User Password" input

# Concept Location

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We ran the application |  |
| 2 | After logging as an administrator, we tried to edit a user and were able to replicate the bug. | This is required to test for the existence of the bug/error message. Error message needed to locate the code. |
| 3 | We went to the source, viewed the source and checked what the "submit" button actually does. Located the "admin user edit". We decided to view the code. Considering the validation bug will be a UI bug. | Because we identified a button on the screen called "Submit".  Also we concentrated on the UI code because the validation Bug is UI related. |
| 4 | Realized the UI is using the MVC pattern.  Starting with the view, Run through the code.  Switched to controllers under the admin package.  Looked at the admin controller first, but was a dead end. |  |
| 5 | We inspected the class usersController.java. And we found the "edit post" comment indicating it's for the user. |  |
| 6 | We ran through the code, and located userService.java on line 220. | UsersController uses the method userService. |
| 7 | Using the message "password field empty" we did a "find string" to determine where the logic is located.  Found in editViewModel.java, in the validate method.  Admin/users/editViewModel.java  Right clicked and "Find usage" on the validate method. (search by text option) | However, we realized this class was not relevant for our change request, because it is rather related to the data layer instead to the GUI layer. |
| 8 | We tried to use the test scripts in the projects to replicate some scenarios. It doesn't look the mock user service in the test. Doesn't seem to be commented out (JUnit test).  We decided to go ahead and debug without the JUnit test. Junit to be used later.  We commented out lines 51 and 52 of the "editViewModel" class for a quick test.  We did a make on the project, restarted the application. Tried to edit, and we got “password must have at least 6 characters…." error message | We decided to use the JUnit testing to get a clearer picture and to ensure we are in the right place. But unfortunately, the JUnit scripts doesn't seem to be working.  We were not sure that method had to be changed, therefore we decided to debug by first commenting out some lines of code. |
| 9 | We marked the class editViewModel as "located". | We confirmed this class had to be modified. |

**Time spent (in minutes):** 75

# Impact Analysis

Use the table below to describe each step you followed when performing impact analysis for this change request. Include as many details as possible, including why classes are visited or why they are discarded from the ones that have to change.

Do not take the impact analysis of your changes lightly. Remember that any small change in the code could lead to large changes in the behavior of the system. Follow the process on impact analysis covered in the class. Describe in details how you followed this process in the change request log. Provide details on how and why you finished the impact analysis process.

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We made a list of methods called by editViewModel and also using “find usage”,  list of classes that call editViewModel. | To track the classes that could be impacted by the change. |
| 2 | We inspected and reviewed the method “ValidationError”. The actual change was made on this method. | We realized this class had to be changed because the method gets called anything the UI control is validated on trying to edit.  No further research was required to ensure this change did not impact other displays based on the MVC pattern being utilized and this ViewModel being associated strictly with the Admin – User – Edit display. |

**Time spent (in minutes):** 23

# Prefactoring (optional)

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| No prefactoring required based on the simplicity of this change. | | |
|  |  |  |
|  |  |  |
|  |  |  |

**Time spent (in minutes):** 0

# Actualization

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We removed password and verifyPassword empty string check from validate method and updated password strength check accordingly in EditViewModel. The logic to checks if "newpassword" is not empty, length is less than 6, has upper case and number. | The verifyPassword empty string check was not required in validate as the UserService updateUser method already accounted for not updating a user password if it was empty. |
| 2 | We ran the program once again |  |
| 3 | We performed functional testing to ensure changes worked (see Validation below). | To make sure everything works.  All tests passed. |
| 4 | We decided to conduct a regression testing on the existing functionalities to ensure we didn't break any existing functionality | This is to ensure changes in the logic didn't impact or change the behavior of user edit functionality  All tests passed. |
| 5 | We committed and pushed our changes with git. |  |

**Time spent (in minutes):** 25

# Postfactoring (optional)

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| No postfactoring required based on the simplicity of this change. | | |
|  |  |  |
|  |  |  |
|  |  |  |

**Time spent (in minutes):** 0

# Validation

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Test case defined: Testing with an inactive user (Anon10)  Inputs: With Password too short  Expected output: Give an error message | This is the regular expected behavior.  The test passed. |
| 2 | Test case defined: Testing with an inactive user (Anon10)  Inputs: With 6 characters without an upper case  Expected output: Give an error message | This is the regular expected behavior.  Got the required error "password must have at least 6…."  The test passed. |
| 3 | Test case defined: Testing with an inactive user (Anon10)  Inputs: With 6 characters, caps and number without the upper case  Expected output: Give an error message | This is the regular expected behavior.  The test passed. |
| 4 | Test case defined: Testing with an inactive user (Anon10)  Inputs: With 6 characters, caps and number with the upper case. Also Edited "about" information  Expected output: Works. Change Saves. | This is the regular expected behavior.  The test passed. |
| 5 | Test case defined: Testing with an active user (Anon14)  Inputs: With Password too short  Expected output: Give an error message | This is the regular expected behavior.  The test passed. |
| 6 | Test case defined: Testing with an active user (Anon14)  Inputs: With 6 characters without an upper case  Expected output: Give an error message | This is the regular expected behavior.  Got the required error "password must have at least 6…."  The test passed. |
| 7 | Test case defined: Testing with an active user (Anon14)  Inputs: With 6 characters, caps and number without the upper case  Expected output: Give an error message | This is the regular expected behavior.  The test passed. |
| 8 | Test case defined: Testing with an active user (Anon14)  Inputs: With 6 characters, caps and number with the upper case. Also Edited "about" information  Expected output: Works. Change Saves. | This is the regular expected behavior.  The test passed. |
| 9 | Test case defined: Try login in with an active user (Anon14), just modified  Inputs: Username and password  Expected output: Successful login | This is the regular expected behavior.  The test passed. |

**Time spent (in minutes):** 22

# Timing

Summarize the time spent on each phase.

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 75 |
| Impact Analysis | 23 |
| Prefactoring | 0 |
| Actualization | 25 |
| Postfactoring | 0 |
| Verification | 22 |
| Total | 145 |

# Reverse engineering





# Conclusions

For this change, concept location was relatively easy because the system is small and its architecture and code are not complicated. Also the change required was just on one class. The details of Concept location, impact analysis, actualization (and change propagation) were captured using OneNote, which was very useful. Testing was performed using manual unit testing due to a lack of functional Junit tests for the existing system. Overall, this was an easy fix.

Classes and methods changed:

* femr/app/femr/ui/models/admin/users/EditViewModel.java
  + Validate()