Change request log

Team

Group 10:

Drew Smith

Nate Craver

Change Request

ID: FEMR-49 Password Constraints Needed FEMR-158

Description: If an administrator tries to edit a user, the "Change User Password" input fields are required and a notification gets sent back that the "password field is empty".

Action should only be taken if these fields are filled out - they shouldn't be required to be filled out.

Concept Location

Step #	Description	Rationale
1	Ran fEMR in IntelliJ	
2	We interacted with the system: after logging in we entered the users directory inside the admin directory	In order to get familiar with some of the features of the system, and identify the elements we had to change.
3	We searched for "changePassword“ using the regular expression feature of the IDE's search tool.	To see what classes/functions were using the changePassword as one of it’s variables
4	Most of the results came from the files CreateViewModel.java and the EditViewModel.java inside the admin directory	We know the changes must be done inside one of the admin directories as that who will be editing the patients information
5	We inspected the validate() function within the EditViewModel class	We saw exactly where the error gets returned, stating that the user must change the password
5	Once we knew where to make changes, we spectated other classes involved	To see how the functions are called to create a new user
6	We marked the class CreateViewModel as "located".	We confirmed this class had to be modified.

Time spent (in minutes): 60

Impact Analysis

Step #	Description	Rationale
1	We made a list of methods called by UsersController	The class calls EditViewModel to edit existing user
2	The validate() function inside the EditViewModel class reports any errors when trying to edit a user. So we, searched for classes that call the validate function	We want to know what classes call the validate() function to make changes to a user.
3	We thought we needed to modify the UsersController class to call the class we modified	We realized that the UsersController class calls the whole class and not just an individual function so we did not have to make any changes to the class. But we did understand that our changes would in fact impact the whether a users information will be changed or not

Time spent (in minutes): 60

Prefactoring

Step #	Description	Rationale
1	Once we implemented out changes to the EditViewModel class we noticed that the IDE suggested to group or “if-else” statements in a more preferable way	By doing so, we were able to eliminate an “if” statement
2	After the previous change, we ran the system to see if our changes took effect	The system worked as planned and no pre-factoring steps were necessary
3	We committed our changes with git	Just in case we need to revert our changes.

Time spent (in minutes): 10

Actualization

Step #	Description	Rationale
1	We completely erased the error stating that the user must enter a “changed password”	The field should not produce any errors if it is left blank
2	We then created “if” statements inside the validate() function to check whether the change password field was filled out or left blank	If filled out, continue error checking to see if the changed password is valid. If left blank, revert any changes done to the users information
3	We ran and tested the system	If the user enters in two different inputs, an error occurs

Time spent (in minutes): 60

Postfactoring

- No post factoring was necessary.

Time spent (in minutes): 0

Validation

Step #	Description	Rationale
1	Test case defined: Inputs: Password: password1 Confirm Password: password1 Expected output: //request accepted	This is the regular expected behavior as the changes to the user took effect due to entering in the change password field.
2	Test case defined: Inputs: Password: password1 Confirm Password: password2 Expected output: Error: passwords do not match	This is an excepted behavior when the two passwords that the user inputs do not match. Shows that the password fields still need to meet the password requirements if they decide to fill it out. The test passed.
3	Test case defined: Inputs: //leave the change password fields blank	This was the expected behavior. If the user decides to leave the change password fields blank, then all changes made to the user will be

	Output: //request goes through	reverted but will still let the action process. The test passed.

Time spent (in minutes): 30

Timing

Phase Name	**Time (in minutes)**
Concept location	60
Impact Analysis	60
Prefactoring	10
Actualization	60
Postfactoring	0
Verification	30
Total	220

Conclusions

The most challenging part of this change request was to actually locate the class in which needed to be modified (Concept Location). After searching through numerous Java and Java Script files, we were finally able to locate where to implement the change. At first we were concerned that we would need to edit a Java Script file for the front-end of the change request but we soon realized that the code inside the Java files that we added would take care of that on its own. It was pretty simple to edit the validate() function to allow a users information to be updated without having to change the users password, by removing one of the functions “if” statements. Last, by implementing the added if-else function inside the validate function, everything worked fine. We did not really need to do any refactoring or post factoring techniques, as they were unnecessary to the small changes we made.