# Change request log

# Team

# Name: Team 4

# Members: Gerald Tucker | Chikadibia Ihejimba

# Change Request

FEMR-208: encounter PDF not displaying amount of prescription dispensed

# Concept Location

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We ran the system |  |
| 2 | Tried to replicate the errors following the steps listed on  <https://teamfemr.atlassian.net/browse/FEMR-208>   1. Create a new patient in Triage 2. Prescribe them some medications in Medical. Make sure the amount is > 0 3. Dispense the prescriptions in Pharmacy (click Submit) 4. View the patients encounter summary by pulling their ID up in Triage and clicking "Patient History" on the bottom. 5. Select the encounter where the prescription was dispensed on the right. The dispensed prescription is displayed here with the amount. 6. 6) Click the red "Generate PDF" button. The dispensed prescription is displayed here, but without the amount. | This is required to test for the existence of the bug/error message. The generate PDF bug needed to be replicated first to be able to know exactly what we are fixing and also how to test when we are done fixing the bug. |
| 3 | We got “Execution exception” error on step 2. Retired and the error persists. Went through the code to locate the class for the prescription. | Because from the steps, we needed to first make a prescription to be able to generate the PDFs |
| 4 | Located MedicalController.Java, went through the logic to check for existence of issues with the logic.  We also went through the following classes:  MedicationRepository.Java  DataModelMapper.Java  MedicationService.Java  PrescritionItem.Java | Still trying to figure out what is causing the runtime “Execution exception “ error. |
| 5 | Stopped the execution, ran in debug mode and stepped through MedicalController.Java method.  Prescription, still not getting created. | This is monitor the variables and data follow during runtime. |
| 6 | Checked the GitHub history to see the last changes made on the file  *From the file => Right click> git > show history* | The is to determine if a recent change was made after the steps in 2 were created. |
| 7 | Checked the master repository to ensure it's in sync with the parent repository | Still trying to ensure we are using the right version of the code |
| 8 | Restarted in debug mode to step through the code once again. By doing the following:  By creating a new patient >Patient search > Treatment page> Pages > Add a patient >Medical Page | We were not sure that method had to be changed, therefore we decided to debug. Still no luck. |
| 9 | Restarted the application from the master repo.  It still failed. Also tried in debug mode. Same results. Then Switched back to the branch 208. | Still trying to figure out what is going on with the code. |
| 10 | Modifying (commenting out) the MedicationService.Java while stepping through to locate the bug. We questioned whether it was expecting an "id" instead of the name of the medication. Filled out the rest of the details without the prescription. Still no luck. |  |
| 11 | We located Form.Java and decided to step through it. On 260, 262 ---added line break to step through the code | This was due to our unfamiliarity with the code and not being sure if this was a new bug or the existing one. |
| 12 | We wondered if the bug was put in there on purpose as part of the test and we decided to ask the prof tomorrow about the bug. Meeting rescheduled to Monday 7 pm. | Team did not want to continue researching the problem until we could verify with professor. |
| 13 | Continued to research the problem. Went to the femr website and attempted to repeat the bug with the live demo which was at version 2.2.1 and was able to save the Prescription. Compared the version to the Master version (which was beyond v 2.2.3) | Comparing the working copy and demo version to replicate the bug |
| 14 | We tracked this down to app/femr/data/daos/system/MedicationRepository.java related to FEMR-86 addition of this concept last month.  Line 184 has the following check:  if (medicationName == null || medicationGenericStrengths == null || conceptMedicationForm == null)  However, we believe the database does allow for strengths and form to be null. |  |
| 15 | The Medical – Prescription form classes that eventually get to this class always pass in null for both of those.  Therefore, a prescription could never be saved.  We went to the femr site and ensured the training version they are running functioned (v2.2.1).  Master is ahead of this released version and did not include the new MedicationRepository concept. |  |
| 16 | Given the above, we changed the condition to:  If (medicationName == null)  After that, we were able to save the prescription and repeat the issue described in femr-208.  This allowed for the tracing of the issue with the PDF not outputting the prescription amounts. |  |
| 17 | We started by trying to determine which controller the HTML page went to by looking at the source of the page and following the structure (similar to the steps above).  This took us to the HistoryController. However, we were not able to find where the Generate PDF action handler was at. Given that, we then went back to the HTML page (indexEncouter.scala.html) again to attempt to locate the action. This led to point to femr.ui.controllers.routes.PDFController.  However, source code could not be found for this. Finally did a string search for PDFController in the code based on and found the proper controller. | This was based on the usage of the MVC pattern for the displays with the View using the model and controller files. |
| 18 | We marked the class PDFController as "located". | We confirmed this class had to be modified. |

**Time spent (in minutes):** 152

# Impact Analysis

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We made a list of methods called by PDFController.java and also using “find usage”, list of classes that call PDFController. | To track the classes that could be impacted by the change. |
| 2 | We inspected and reviewed the method “getAssessments”. 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. |
| 3 | We also inspected " indexEncounter.scala.html”, considering that this is a UI change. There was no need to further review the impact. | The change was minimal and it was a HTML display tag change. |
| 4 | We made a list of methods called by MedicationRepository.java and also using “find usage”, list of classes that call MedicationRepository.  We also inspected and reviewed the method “createNewMedication”. The actual change was made on this method. | To track the classes that could be impacted by the change. The change wasn’t part of the change request, but we had to apply a temporary fix in order to be able to test the CR 208, and apply the fix. |

**Time spent (in minutes):** 38

# 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 | Fixed issue with dosage amount not being displayed on the PDF. After adding, ensured that both display (indexEncounter.scala.html) and PDF formats where identical.  Finally, found a new bug in MedicationRepository that would not allow for the addition of a new prescription. Added a temporary workaround, commented with a todo, and proceeded with the original fix. |  |
| 2 | 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 |
| 3 | We performed functional testing. | To make sure everything works. |
| 4 | We committed and pushed our changes with git. |  |

**Time spent (in minutes):** 62

# Postfactoring (optional)

Using the table below, describe each step you followed to postfactor the code. Include as many details as possible, including the refactoring operations used (e.g., move method, extract class, etc.) and classes/methods/fields that were modified, added, removed, renamed, etc.

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | During testing to repeat the bug and of the fix we noticed that the History display format for the prescription information did not match that of the PDF (specifically the medication form information). Also, for replaced prescriptions, the amount was not shown. | The concern was that we were already fixing an issue of inconsistency of report and display for the users. Given that we found additional issues like this during testing, we wanted to proactively resolve them to avoid confusion for the users. |
| 2 | To fix the display issue we modified history/indexEncounter.scala.html on line 203, added @prescription.getAmount in the strikethrough tag before getOriginalMedicationName. |  |
| 3 | To fix the PDF issue we modified PDFController to retrieve medicationForm on line 441 and display it on the report on line 460 and 464. |  |
| 4 | After the previous change, we ran the system and executed functional tests (described in Validation). | We tested everything was working as before, after the refactoring. |

**Time spent (in minutes):** 14

# Validation

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Test case defined: Create a New Patient in Triage, add some Prescriptions, dispense and generate PDF  Inputs: Patient Name, prescriptions values  Expected output: Generate a PDF with same display as UI | This is the regular expected behavior.  The test passed. |
| 2 | Test case defined: Add some Prescriptions to an existing patient, dispense and generate PDF.  Inputs: prescriptions values  Expected output: Generate a PDF with same display as UI. | This is the regular expected behavior.  The test passed. |
| 3 | Test case defined: Create a replacement prescription to ensure the strikethrough portion with dosage was shown  Inputs: Patient Name, prescriptions values  Expected output: Generate a PDF with same display (strikethrough portion) as UI | This is the regular expected behavior.  The test passed. |

**Time spent (in minutes):** 15

# Timing

Summarize the time spent on each phase.

|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 152 |
| Impact Analysis | 38 |
| Prefactoring | 0 |
| Actualization | 62 |
| Postfactoring | 14 |
| Verification | 15 |
| Total | 281 |

# Reverse engineering





# Conclusions

For this change, concept location was at first difficult, not because the system size or the architecture but due to a bug encountered while trying to replicate the original bug in UI. The new bug was eventually located and temporarily worked around before the concept location of the actual change request. The details of concept location, impact analysis, actualization (and change propagation) was 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. The fix was easy, though we had to go extra mile by formatting the displays on both UI and generated PDF to completely match.

Classes and methods changed:

* app/femr/data/daos/system/MedicationRepository.java
* app/femr/ui/controllers/PDFController.java
* app/femr/ui/views/history/indexEncounter.scala.html