# Change request log

# Team

Team:6

Raghupathi Vasantavada(Raghu)

Devika Chakraborty

# Change Request

FEMR-137: flag birthdays as being accurate or a guess

-As a researcher, I want fEMR to flag whether or not the patient's birthdate is real or if fEMR tried to guess what it was so that data is accurately stored with integrity.

Notes:  
1) Patients often do not know what day they were born and sometimes don't even know their age.  
2) fEMR allows you to identify how old a patient is in 3 different ways - enter the actual birthdate, enter an age integer, or select a category (child,adult,elder,etc). Our database has 2 options of storing this data - a birthdate OR the category. If the user enters an age integer, the system will create a fake birthdate, but it won't flag the birthdate as fake for future reference.

**Concept Location**

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
|  | We started the application and logged in.  Clicked on Triage Screen and figured that age is captured on this screen. | To figure out which screen is responsible for capturing birthday. |
|  | We figured where age is captured on the UI using 3 different ways- enter the actual birthdate, enter an age integer, or select a category (child,adult,elder,etc). After looking at the UI, we guessed that birthday related data may have been stored in some patient master table | To identify how data is displayed on the UI and possibly, how and what data is captured.  The assumption about the patient related master table comes from common sense understanding that patient profile data is most likely stored in one table. |
|  | Using MySQL workbench, we reverse engineered to generate ER diagram after filtering for tables that begin with ‘Patient’ | To confirm our assumption from the above step. |
|  | After carefully observing the ER diagram by comparing it with the UI fields on the Triage screen   1. we realized that patient profile information is being stored in the ‘patients’ table. 2. Patient\_age\_classification is the master table where the age classification is stored. E.g., (0-1) baby etc. 3. We looked at the data in patients table and figured that the age is always stored in mm/dd/yyyy format and hence unsure how the data in ‘patient\_age\_classification’ is mapped to this table. Again revisited the ERD and concluded that ‘patient\_age\_classification’ has a mapping with ‘Patient\_encounters’. Looked at the data in the ‘Patient\_encounters’ table and figured out that the ‘Patient\_age\_classification’\_id is stored in this table for each patient. 4. We figured out that only If the patient\_age\_classification\_id in the ‘Patient\_encounters’ table is not null then the patient’s record in patients table is populating birthday date. 5. At this point we are clear that if the user chooses ‘date of birth’ or ‘years and months’ , the age column in the patients table is getting populated. We confirmed the same after entering 3 patient records from UI by choosing each of the 3 ways to enter the age. we concluded that we needed to focus on the other two options- date of birth or age in years and months to fix this enhancement. | We wanted to check how the birthday is stored in patient\_encounters table |
|  | At this point, we discussed the following options that would make it easier to guess the birthday  Option A- Assume Birthday as fake if the date part of birthday is same as Trip date.  Option B- Make the User forcefully enter if the date of birth is real or fake.  We concluded that option A is feasible because of the following reasons  Trip can be conducted for a date range and if patient’s real birthday falls on one of the trip dates, the system might treat birthday as Guess when it is real. | Find options to fix the change request the right way.    We zeroed in on adding a new field to determine real or fake birthday at the triage screen because this is the only screen when age(DOB) is created. And in the edit screen, the age is not an editable field. |
|  | We went to the triage screen and checked the fields where age is entered.  We concluded that we might need to add a new field (checkbox or radio button) so that user can capture this flag and persist in the ‘patients’ table for every patient.  We checked the edit patient screen to see if birthdate can be modified and realized it’s not editable which confirmed that the new field in the triage screen itself to check for fake or real birthday would suffice. | To identify the screen location |
|  | We determined that it would be nice to the end user if we implement this field just like the Gender | As the gender is implemented with a group button, in order to keep the UI look and feel consistent, it would be wise to have a radio button |
|  | Searched through the IDE for ‘Triage’ package for finding the view where we need to make code change and concluded that index.scala.html needs to be modified under femr.ui.views.triage package | To reflect the change in the UI |
|  | We also needed to add a new column in the database table Patients to store the value of the above flag | For retrievals/references/research purposes |

**Time spent (in minutes):** 45

# Impact Analysis

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Impact analysis was hard for this as it’s an enhancement. We had to check in every layer wherever patient object is used. So we decided to generate a Sequence Diagram using visual-paradigm tool. After which we made a list of methods that could be affected by the addition of the new radio buttons and to make it work | To track the classes that could be impacted by the change. |
| 2 | We have navigated through edit triage, medical, pharmacy and research screen to figure out where possibly this field could be in use. | In order to understand the impact of the change |
| 3 | We manually tested the application by creating a new patient and then retrieving the patient from the database. We needed to check if the real fake birthday radio buttons are working correctly and no other functionality changed because of this | To make sure the enhancement works and does not affect another functionality |

**Time spent (in minutes):** 45

# Actualization

Using the table below, describe each step you followed when changing the code. Include as many details as possible, including why classes/methods were modified, added, removed, renamed, etc.

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We created a new partial page InputAgeReal.scala.html under femr.ui.views.partials.triage, and added the label ‘is age real?’ with the 2 yes or no toggle buttons | To display and accept the age is Real or Not input |
| 2. | For the UI change, we added radio button with the name ‘isAgeReal’  in the index.scala.html under femr.ui.views.triage | To reflect the change in the UI |
| 2 | We updated the indexViewPostModel.java under Femr.ui.models.triage. by creating a new private field ‘isAgeReal’ with getters and setters | To update view model |
| 3 | Under femr.common.models, we updated PatientItem with new private field ‘isAgeReal’ with getters and setters | To update the model |
| 4 | As we were done with editing UI layer(MVC), went to PatientService.java as it is called by TriageController and realized that createPatient method of the  dataModelMapper is returning IPatient type of object   1. Edited IPatient.java interface in the package femr.data.models.core and added a private field ‘isAgeReal’ with getters and setters 2. Edited Patient.java in the package femr.data.models.mysql   And added the new field is AgeReal along with the class-table mapping annotations.   1. PatientService is calling createPatientItem method on the   ItemModelMapper and hence edited IItemModelMapper.java interface to add ‘isAgeReal’ in the constructor. | To update Service Layer and DAO layer. |
| 5 | We went SearchService.java. Under SearchService we updated every reference to patient object by adding the ‘IsAgeReal’ property | The PatientID search should return the field isAgeReal |
| 6 | To store the value of real or guessed birth day, we needed to create a column IsAgeReal in the patients table in the database | To store the value of real/guessed birthdays for future reference (with value=0 meaning real birthday and value =1 meaning guessed/fake birth day) |
| 7 | We tested manually | To make sure everything works. |

**Time spent (in minutes):** 140

# Validation

Using the table below, describe any validation activity (e.g., testing, code inspections, etc.) you performed for this change request. Include the description of each test case, the result (pass/fail) and its rationale.

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | Test case defined:  Inputs:  1.Create a new patient in the triage screen.  2. The ‘Birthday type’ label with 2 radio buttons ‘real’ and ‘fake’ should appear | This test validates that the newly added field ‘birthday type’ shows up  Test passed |
| 2 | Test case defined:  Inputs:  1.Create a new patient in the triage screen.  2. Choose ‘Birthday type’ as ‘fake’  3. Enter his age , say 40 years, 3months .  4. Enter other details then Submit    Expected output:   1. ‘The patient ID <ID> created successfully’ message should be displayed 2. check the database column IsAgeReal, it should have value 1 for fake birthday using the following query   “Select IsAgeReal from patients where patient\_id=<PatientID displayed on screen>’ | The test validates that the create patient logic is working as expected when patient’s birthday type as fake is selected  The test passed. |
| 3 | Test case defined:  Inputs:  1.Create a new patient in the triage screen.  2. Choose‘Birthday type’ as ‘real’  3. Enter his exact date of birth, in mm/dd/yyyy format  4. Enter other details then  Submit    Expected output:   1. ‘The patient ID <ID> created successfully’ message should be displayed 2. check the database column IsAgeReal, it should have value 1 for fake birthday using the following query 3. “Select IsAgeReal from patients where patient\_id=<PatientID displayed on screen>’ | The test validates that the create patient logic is working as expected when patient’s birthday type as Real is selected  The test passed. |
| 4 | Test case defined:  Inputs:  1.Go to Triage screen and enter patientID obtained from Test case 2 and hit submit      Expected output:   1. The triage screen should populate patient’s details 2. The Birthday type ‘Fake’ should be displayed as selected | This step confirms that edit triage screen is displaying patient’s birthday type correctly |
|  | Test case defined:  Inputs:  1.Go to Triage screen and enter patientID obtained from Test case 3 and hit submit      Expected output:   1. The triage screen should populate patient’s details 2. The Birthday type ‘Real should be displayed as selected | This step confirms that edit triage screen is displaying patient’s birthday type correctly |

**Time spent (in minutes):** 20

# 

# 

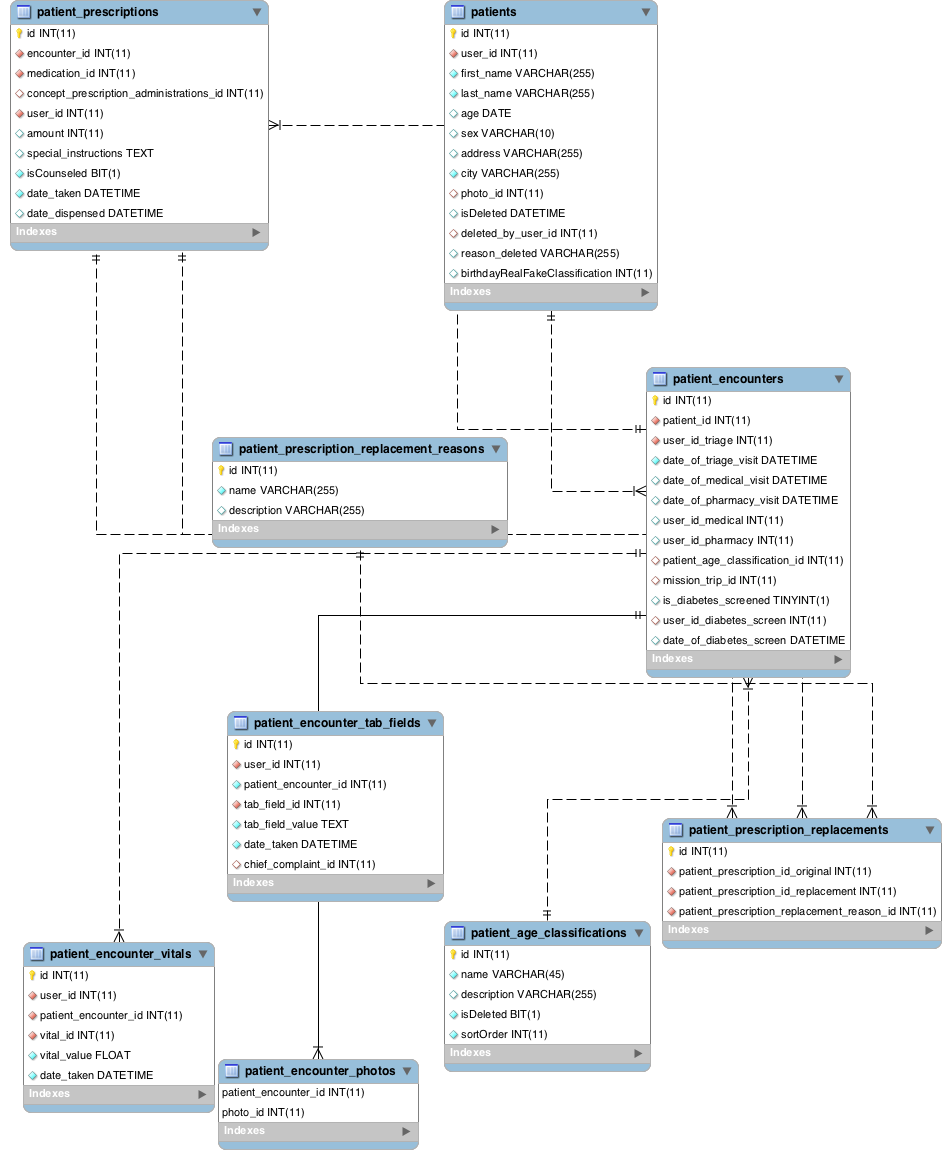
# Timing

Summarize the time spent on each phase.

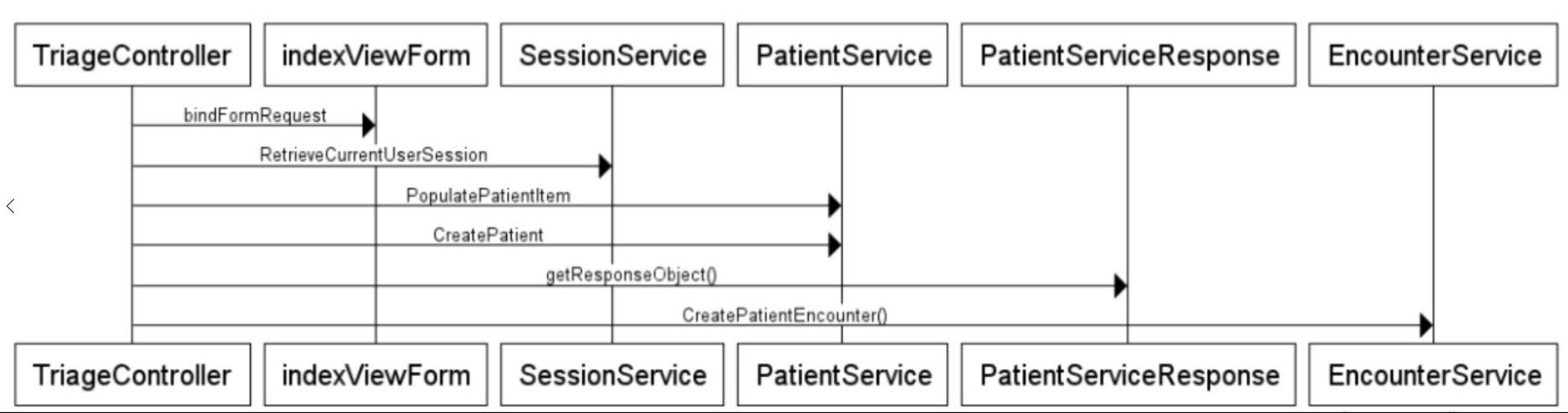
|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 45 |
| Impact Analysis | 45 |
| Prefactoring | 0 |
| Actualization | 140 |
| Postfactoring | 0 |
| Verification | 20 |
| Total | 250 |

# Reverse engineering

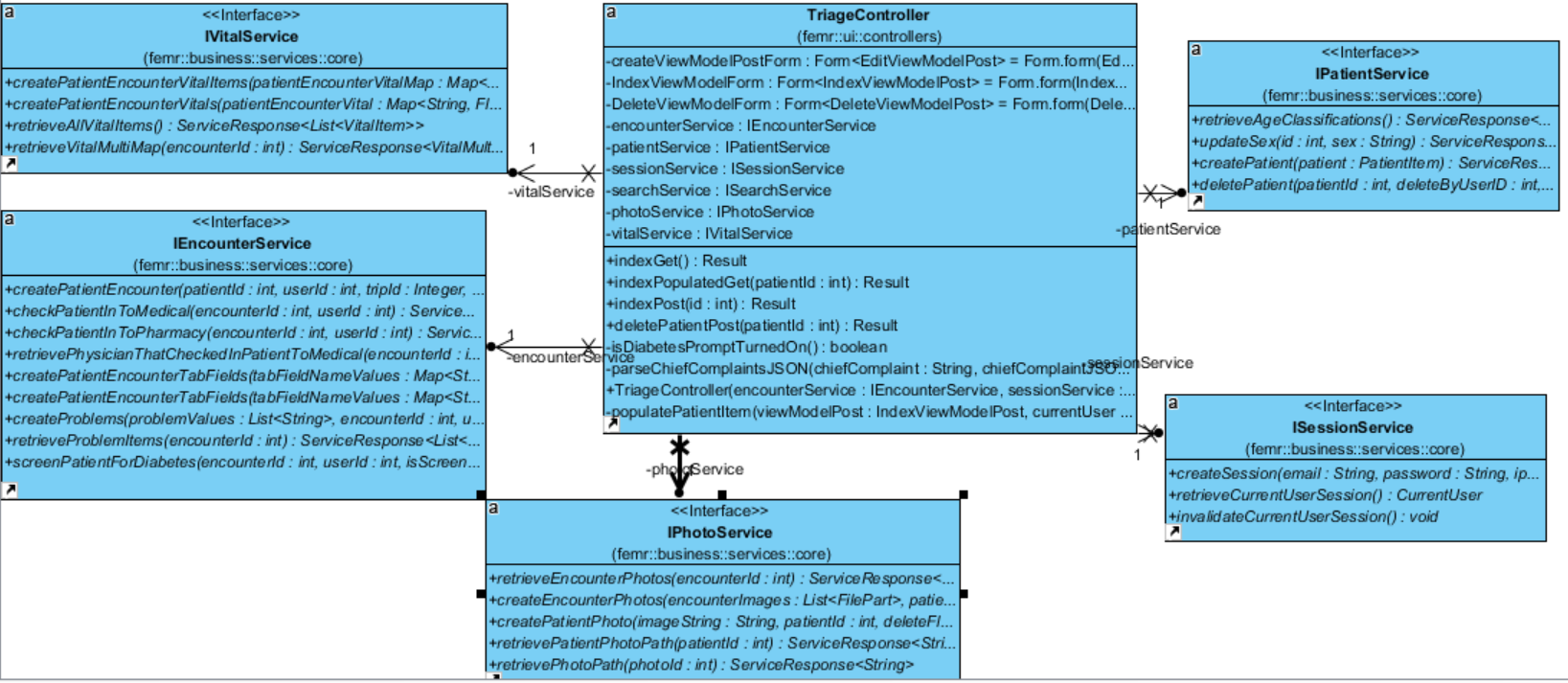
**Entity Relationship Diagram**



**Sequence Diagram**



**Triage Class Diagram**



# Conclusions

For the change concept location was moderately easy because we decided to add radio buttons to denote fake or real birthday. However propagating that change through the view , model and controller layer all the way to the database was challenging because it took time for us to identify how the data is being populated in the front end and how it is retrieved in the application. The impact analysis, actualization was relatively easy because the architecture is relatively easy to understand.

We performed manual verification that the fix is working.

Classes and methods changed:

1. index.scala.html under femr.ui.views.triage
2. femr. business.service.system
3. PatientService.java
4. SearchService.java
5. femr.common

a. IItemModelMapper.java

b. ItemModelMapper.java

1. femr. common.models
2. PatientItem.java
3. femr.data
4. IDataModelMapper.java
5. DataModelMapper.java
6. femr.data.models.core
7. IPatient.java
8. femr.data.models.mysql
9. Patient.java
10. femr.ui.controllers
11. TriageController.java
12. Femr.ui.models.triage
13. indexViewModelPost.java
14. Added database column ‘isAgeReal’ in Patients Table