# 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. Concept Location

**Concept Location**

|  |  |  |
| --- | --- | --- |
| Step # | Description | Rationale |
| 1 | We generated the ER Diagram for the database tables | We wanted to check how the database columns are mapped to the birthday column |
| 2 | We found the table patients\_age\_classification and patient\_encoutners table. Patient\_age\_classification is the master table where the age classification is entered. E.g., (0-1) baby etc. We found that birthdate is null in patient\_encounters table if range is chosen by the user for the birthday of the patient. So we concluded that we needed to focus on the other two options- date of birth or age in years and months | We wanted to check how the birthdate is populated in patient\_encounters table |
| 3 | We went to the triage screen and checked the fields where age is being input. We concluded that given a date of birth or age in years and months, we need to capture if the birthday is real or a guess on that screen itself.  We checked the edit patient screen to see if birthdate can be modified and realized it’s not editable. We decided to have a new field in the triage screen itself to check for fake or real birthday | 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. |
| 4 | We thought of implementing a radio button to determine the birthday type fake or real. | To let the user input if the birthday is real or a guess |
| 5 | For the UI change, we needed to change in the index.scala.html under femr.ui.views.triage | To reflect the change in the UI |
| 6 | 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. 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 used | 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 other 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 | For the UI change, we needed to change in the index.scala.html under femr.ui.views.triage | To reflect the change in the UI |
| 2 | We needed to update the triage model, Femr.ui.models.triage. We needed to create a private field with getters and setters | As we changed the UI, the change needed to trickle down through all the layers of the architecture |
| 3 | We checked the TriageController.java and looked at ways it populates the PatientItem, We populated the patient object with the IsAgeReal obtained from the view model post | As we changed the UI, the change needed to trickle down through all the layers of the architecture |
| 5 | We went to IItemModelMapper.java interface and looked at the method definition CreatePatientItem and updated IsAgeReal argument  Then we looked at the service layer PatientService.java and the method name CreatePatient and added PatientIsAgeReal in the dataModelMapper class.  We went SearchService.java. Under SearchService we updated every reference to patient object by adding the IsAgeReal property  We went to Ipatient.java interface under femr.data.models.core and updated Ipatient.java interface and added the IsAgeReal property there  Then we went to femr.data.models.mysql and then under that we went to Patient.java. Here we created the JPA (Java persistence Annotations) to map the property to the column name. | As we changed the UI, the change needed to trickle down through all the layers of the architecture |
| 5 | To store the value of real or fake , we needed to create a column IsAgeReal in the patients table in the database | To store the value of real/fake birthdays for future reference (with value=0 meaning real birthday and value =1 meaning guessed/fake birth day) |
| 6 | 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

# 

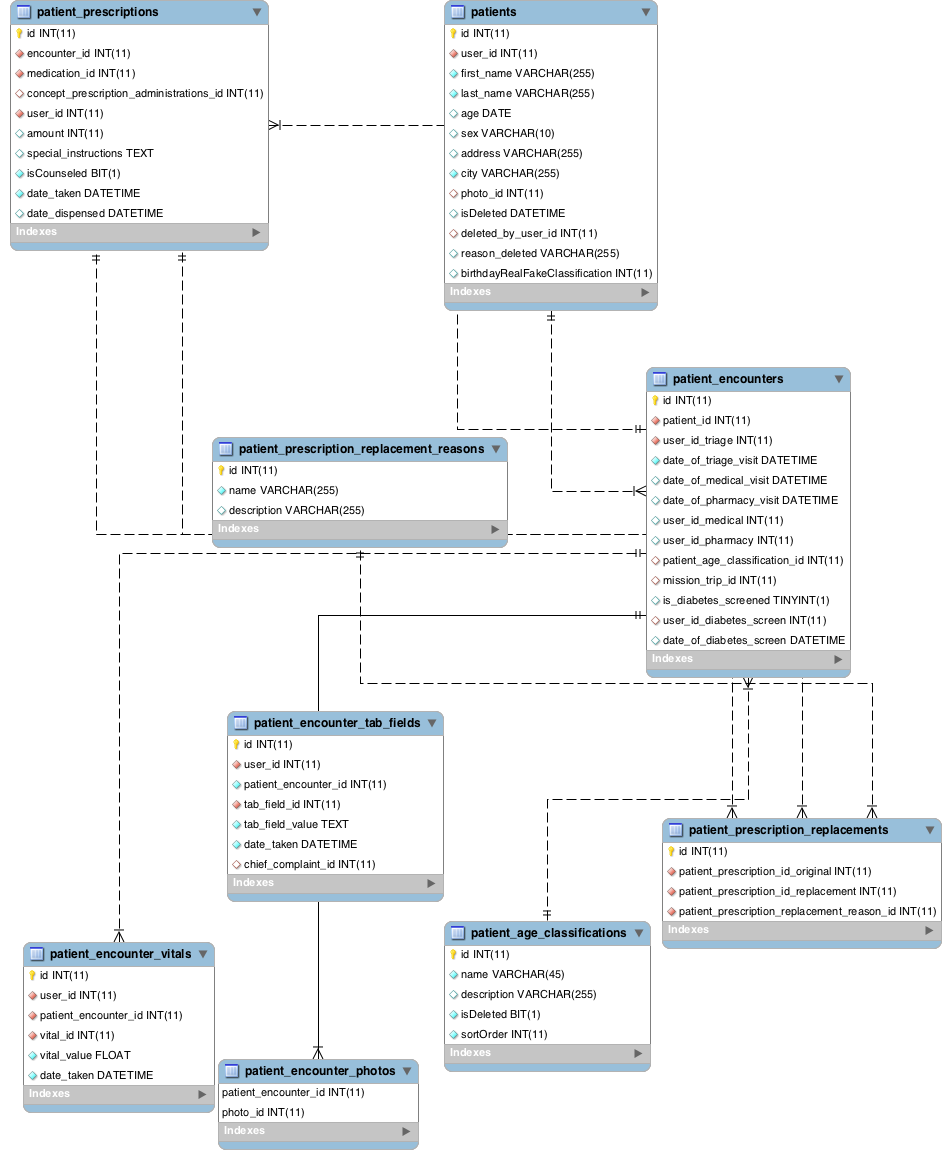
# Timing

Summarize the time spent on each phase.

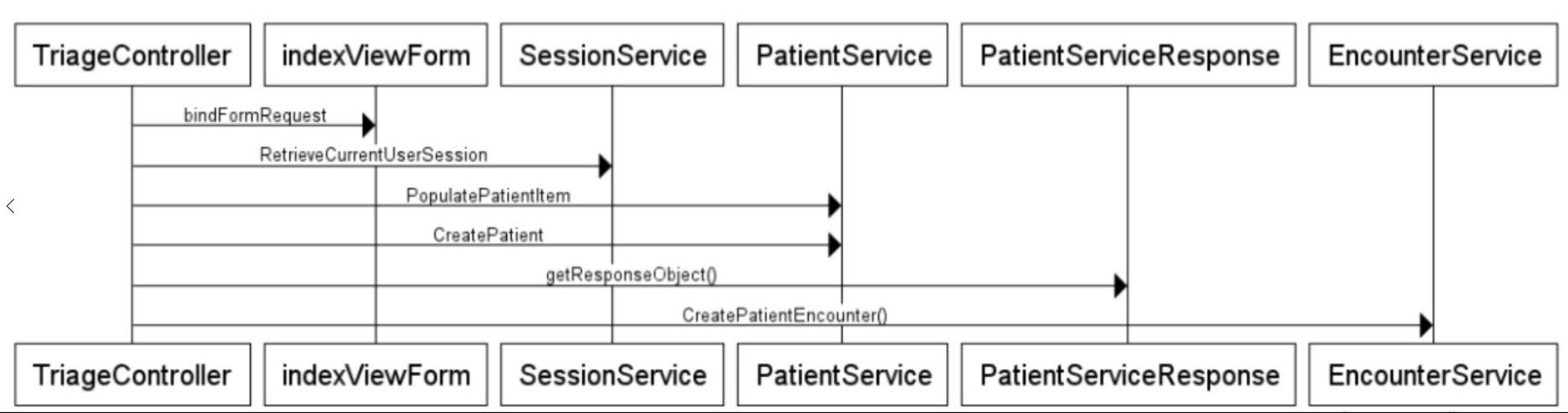
|  |  |
| --- | --- |
| Phase Name | Time (in minutes) |
| Concept location | 45 |
| Impact Analysis | 45 |
| Prefactoring | 0 |
| Actualization | 140 |
| Postfactoring | 0 |
| Verification | 20 |
| Total | 250 |

# Reverse engineering

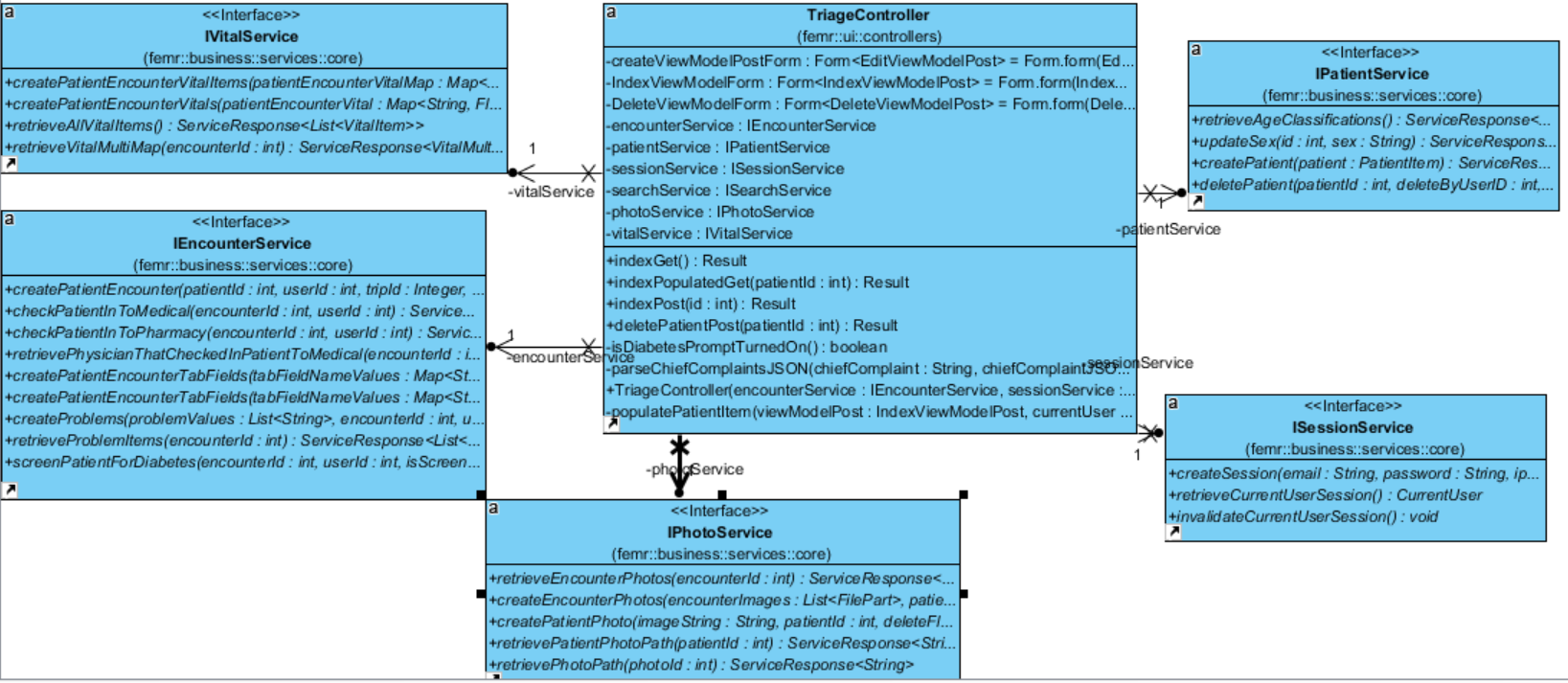
Entity Relationship Diagram



**Sequence Diagram**



**Triage Class Diagram**



# Conclusions

For the change concept location was moderately easy because we decided to add radio buttons to denote fae or real. 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 PatientRealFakeBirthdayClassification in Patients Table